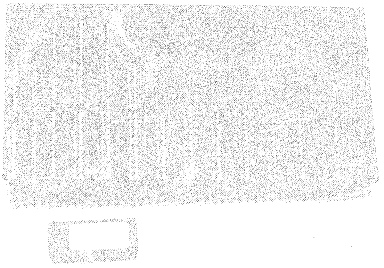
£1.00

BEEBUG FOR THE BBC MICRO

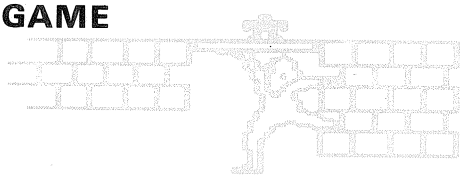
Vol 2 No 9 MARCH 1984

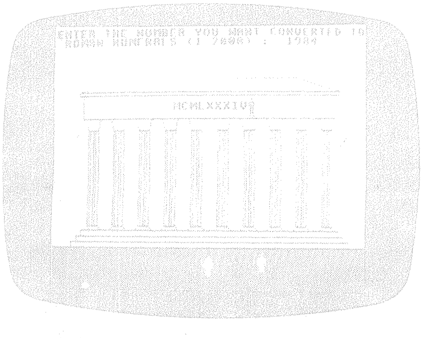
ARIES MEMORY BOARD REVIEWED



MULTIPLE DISC CATALOGUES

MANHOLE GAME





```
ENTER THE NUMBER YOU WANT CONVERTED TO
 ROMAN NUMERALS (1-2000) : 1984

MCMLXXXIV
```

THE STONEMASON

PLUS
- **ASTAAD EXTENDED**
- **TESTING YOUR MICRO**
- **FORTH REVIEWS**
- **ACORN DEVELOPMENTS**
- **COMPACT FUNCTION KEY DEFINITIONS**
- **BACH CANTATA**
- **And much more**

BRITAIN'S LARGEST COMPUTER USER GROUP
MEMBERSHIP EXCEEDS 20,000

EDITORIAL

THIS MONTH'S MAGAZINE

The computer aided design program called ASTAAD that we featured in the December 1983 issue of BEEBUG generated a lot of interest and ideas for further enhancement. This month we show you how to extend the original version to provide a host of new features.

We are also starting a new series of DIY articles which are designed to help you test out your machine when you suspect a fault, and identify the source. This should help you when you take your machine along for repair.

SECOND PROCESSORS

Acorn have still given no firm date for the release of their long awaited second processor systems, though this is still expected to be around Easter time. What is even more disturbing is the report that Acorn already have numerous orders for 65Ø2 second processor systems for use as second level econet systems, and that as a consequence, any other orders for 65Ø2 second processors may have to wait for up to 12 months before supplies become generally available.

TICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BOA

WHAT TO DO WITH YOUR OLD O.S. 1.Ø

If you still have one of the old 1.Ø operating systems, supplied as 2 EPROMS on a carrier board, now is the time to exchange it. BEEBUG are currently able to offer in return either the new O.S. 1.2 ROM, or a £5 voucher which can be used in part payment for any item from BEEBUG or BEEBUGSOFT. This offer only applies to the 1.Ø version and not to the original version Ø.1 which was also supplied initially on a carrier board. If you are unsure which version O.S. is in your machine, then typing '*FXØ <return>' will display this information on the screen. Full details of this offer are at the back of this issue.

ENQUIRIES

As a result of further expansion of staff at BEEBUG we are installing a telephone line to deal with enquiries relating to BEEBUG and BEEBUGSOFT. This is to help with problems relating to subscriptions and orders for backcopies of the magazine, software and hardware. It will not be possible to answer technical queries over the telephone. The number to ring is St Albans 6Ø263 and the line will be manned from 1pm to 4pm each weekday starting from the 1st March.

HINT WINNERS

This month's hint winners are A.Pemberton who wins the £1Ø prize and S.J.Wilkinson who wins the £5 prize. We still need all your good hints and tips for the magazine.

MAGAZINE CASSETTE

This month's magazine cassette contains the full version of our extended version of ASTAAD (called ASTAAD2), the very popular computer aided design program first described in the December 1983 issue of BEEBUG, and the winning program (ROMAN) by Keith Walker in our Roman Numeral Brainteaser competition, which appeared in the November supplement. We have also include the four catalogue versions of the two programs presented in this month's article on Multiple Disc Catalogues, a very useful facility for all disc users.

BEEBUG MAGAZINE

GENERAL CONTENTS

PROGRAMS

HINTS, TIPS & INFO

NEWS

The Chip Shop

On Saturday at 5pm, BBC Radio 4 is presenting 'The Chip Shop', a series of programmes about the world of computing. The Chip Shop is also offering a takeaway service of computer software that will be broadcast over the air. This will use an Esperanto version of Basic developed in the Netherlands and called Basicode. Details of this are available as 'Factsheet 1' from The Chip Shop, BBC, London W12 8QT.

Acornsoft Cassette to Disc Exchange

Acornsoft have launched an exchange service whereby purchasers of their cassette software can, at a later date, exchange this for a disc version at a cost of 50% of the normal price of the disc version. To use the service, simply send your cassette with the correct money to: Disc Replacement Service, Acornsoft Ltd., c/o Vector Marketing, Denington Industrial Estate, Wellingborough, Northants NN8 2RL. The service is available now, though you should check the current Acornsoft catalogue for availability (and price) of 40 or 80 track disc versions.

'Acorn User' changes hands

Addison-Wesley have recently decided that their British subsidiary should no longer continue publishing the successful magazine 'Acorn User'. The magazine will in future be published by Redwood Publishing, a company set up by Acorn.

Interfacing the BBC Micro

A range of fully isolated interfaces for the BBC micro (from £44.98 to £112.48 inc.), available from Minor Miracles, has been designed for home users. Extra tough versions are available for school and industry use. Minor Miracles also sell a digitiser (£36) and a light pen (£15 inc). Further information may be obtained from Minor Miracles, P.O.Box 48, Ipswich, IP4 2AB.

The BBC Micro Speaks Out!

The Microtalker is a new low cost speech synthesiser using allophones, which allows any English word to be pronounced, but at a slightly lower quality speech than the LPC system used by Acorn, is available from R.P.S Electronics called the MICROTALKER, for £39.95 (+80p post + VAT). It plugs into the User Port, and contains a built in amplifier, speaker, and volume control. The MICROTALKER is available from:
R.P.S Electronics, Unit C200, Saltaire Workshops, Ashley Lane, Shipley, West Yorkshire, BD17 7SR.

Business Programs

HCCS have recently produced a range of Integrated Business Software written for the BBC disc system. The full package includes Invoicing and Sales Ledger, Purchase Ledger, Nominal Ledger, Payroll, Stock Control and Order Processing. The cost of each module is £59.95 each, and is available from:
HCCS, 22 Market Square, Biggleswade, Beds, SG18 8AS.

Owl Perch

This is a sturdy monitor support, which stands over the BBC micro, raising the monitor to eye level and protecting the micro. They are cast in metal and painted with a cream enamel paint. The cost is £35 (inclusive) and a partly finished and unpainted version which costs £20 (inclusive) is also available. For more details, contact:
William Broady & Son Ltd., English Street, Hull, HU3 2DU.

BBC Bridge

ICB Enterprises have recently introduced a Bridge program which can produce a random deal, play three hands and includes many other facilities. It costs £20.50 inc. Contact: ICB Enterprises, 33 Burnt Stones Drive, Sheffield, S10 5TT.

EXTENDING ASTAAD (32K)

by David A. Fell

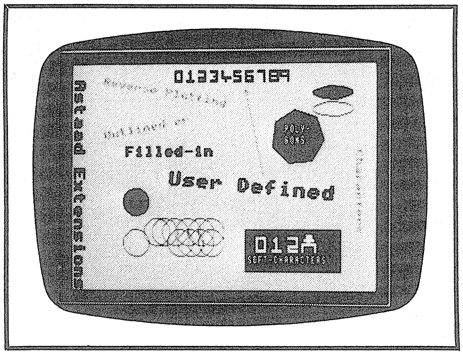
The Computer Aided Design program called ASTAAD that we published in the December 1983 issue of the magazine provoked quite an enthusiastic reponse from readers. Many of you suggested ways in which the original program could be extended and improved. We have implemented six important new features in a modular and comprehensive update of the original program to produce ASTAAD2.

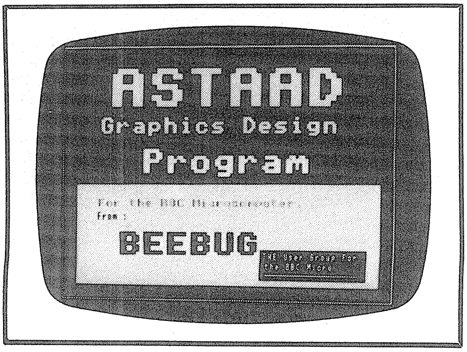
The sections to be added to ASTAAD fall into five areas:

1. Facilities to load and save the current picture to tape/disc, and to call a user-supplied machine code printer screen dump program.

2. The ability to include user-defined characters in 'ASTAAD' text, and the facility to draw 'filled in' polygons and circles. Both of these are 'toggled' (that is to say, they are controlled by a 'switch' that is either on or off).

3. The facility to reverse the plotting colours. This means that the 'area-clear' facility can fill in a rectangle, and then text can be made to appear in the background colour on this area. This extension applies to all functions.




ASTAAD
Graphics Design
Program
For the BBC Microcomputer, from :
BEEBUG
THE User Group for the BBC Micro

4. We have also included a dummy procedure to illustrate how easy it is to incorporate your own features. It shows the correct way to select a text display, and also how to exit back to the main part of ASTAAD2.

5. Finally, we have also included some tidying up of various aspects of the program, and some small extras to make it more attractive. These include the production of a tone whenever TAB or one the function keys is pressed, the inclusion of a border around the drawing area (with extra protection to ensure that text does not stray into the drawing area, and vice versa), and better error handling facilities (see later for more details).

All the new features are controlled by means of the Shift key in combination with one of the red function keys.

»

KEY	ACTION
Shift + fØ	Save Current Screen
Shift + f1	Load new Screen
Shift + f2	*RUN a printer dump program
Shift + f3	Toggle the soft characters
Shift + f4	Toggle the infill switch
Shift + f5	Call to dummy user routine
Shift + f6	Toggle reversal of colours

The other function keys (and the cursor keys) will merely bleep if pressed when shift is held down.

ASTAAD

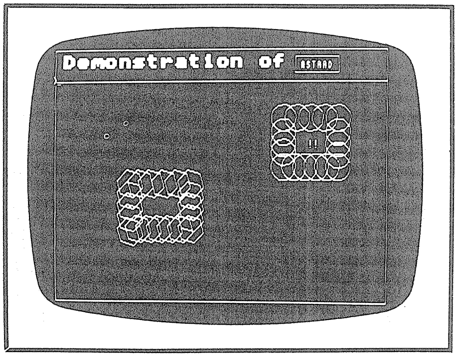
Normally, in ASTAAD2, the screen is kept separated into two windows. The main drawing window is defined as the graphics window, and all the drawing is carried out in this window. When a prompt is needed, the text and graphics cursor are separated (with VDU4), and the text is input in the current text window (the top two lines on the screen).

Error handling has been improved considerably. When an error occurs, a check is first made to see if it is an Escape error. These have been split into two types: Escape and Shift-Escape. Escape is the same as the original ASTAAD, and merely resets a few paramters and sets the cursor to the centre of the screen. This can occasionally leave part of the cursor on the screen, and so this is not the best way of centring the cursor. Holding down Shift whilst pressing Escape causes a normal Escape error to be generated. Whenever any normal error occurs, the error message and line number are displayed at the top of the screen. The user is then prompted whether or not they wish to continue. This allows for the user to exit without having to press the Break key. It also means that if an error occurs, say when trying to find a file on disc, the picture just created is not lost. Thus to exit ASTAAD2, press Shift and Escape, and answer with an 'N'.

PRINTER SCREEN DUMP

The screen dump facility enables the user to dump the image displayed on the screen to their printer. The routine has been tailored so that a machine code printer dump program will be loaded and run when required (equivalent to *RUN). The screen dump must be assembled and saved on cassette or disc prior to using it with ASTAAD2 and you will need to decide at what memory address to do this. Disc users will normally be able to assemble their screen dump program at &AØØ while cassette users should use &DØØ, though you will need to make sure that your assembled screen dump will fit in one page (256 bytes) of memory. Once assembled the screen dump program should be *SAVEd ready for use. The screen dump is selected in ASTAAD2 by pressing Shift-f2 and entering the name of your saved screen dump routine, which is then loaded (from cassette or disc) and run.

Cassette users, particularly, might prefer to load the screen dump routine (use *LOAD) before running ASTAAD2 which will need amending so that the screen dump routine is CALLed when required, thus saving the time involved in loading.



SAVING AND LOADING SCREENS

The screen in Mode Ø occupies memory in one continuous block from &3ØØØ to &7FFF. To save the screen, all that is needed is to supply a filename, and ASTAAD2 will do the rest. To save a picture, a typical sequence of actions might be as given below:

1) Construct your desired image.
2) Hold down Shift and press f0.
3) Enter filename and press Return.

If you are using tape, you will now have to prepare yourself for a fairly long wait! On exit from the save routine, the cursor is left where it was before the save operation was performed.

Loading of a screen is done in a similar manner. To load a screen, press Shift-f1, and enter the filename of the screen to be loaded. Once the screen has been loaded, the cursor is set to the centre of the screen. The previous contents of the screen will be completely replaced by the new screen as it is loaded.

USER DEFINED CHARACTERS

'ASTAAD' is also the name given to the routine brought into action by pressing f0. ASTAAD stands for Any Size Text printed at Any Angle on the Drawing. This facility, however, was limited, in the original version, to the normal keyboard characters. An extension has been provided to allow for the soft (or user defined) characters to be used instead. This allows for new characters to be designed, and then also drawn at any angle, and at any size.

The choice of characters is controlled by Shift-f3. With this in the 'off' state, ASTAAD behaves as it did before. When in the 'on' state, user defined characters are selected from the keyboard with 'A' selecting character 224, 'B' selecting character 225 and so on. The characters are displayed at the top of the screen as selected. This allows for the new string to be seen as it is being built up. The user defined characters must be set up before loading ASTAAD2 with the VDU 23 statement (see User Guide pages 170 and 384). The Shaper program, published in BEEBUG Vol.2 No.6, is an ideal way to design characters that you would like to use in ASTAAD2.

As an example, try the following. Enter Basic (pressing Break will do if you are already in Basic) and type in:

```
VDU23,224,0,255,191,159,159,145,17,17
```

Now load ASTAAD2, and press 'Shift-f3'. (If it now says that the soft characters are off, press 'Shift-f3' a second time.)

Press 'f0' for ASTAAD text
Press 'A' and Return
Enter a size, say 20
Enter an angle, say 10

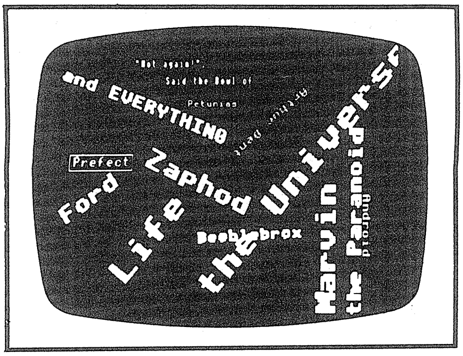
Now watch a primitive elephant being displayed. This illustrates that the soft character is now being read from the memory definition.

FILLED POLYGONS AND CIRCLES

Another new feature, controlled by Shift-f4, is the in-filling of polygons (and thus also circles, as they are drawn using the polygon routine). Normally, when a polygon is drawn in ASTAAD, it appears only as an outline, but ASTAAD2 allows for either an outline, or a solid polygon to be drawn. The selection between the two is made by pressing Shift-f4. For example, if pressed once, then 'in-fill' may be turned off. This means that only the polygon's outline will be drawn. If Shift-f4 is pressed again, then the switch will be altered, and now a solid polygon will be drawn. If the f3 option is used to repeat the last polygon drawn, then whether or not the polygon is filled in is dependent on the current state of the in-fill switch, and not the state of the switch when the initial polygon was drawn. This feature also applies to the circle drawing routine selected by f9.

REVERSING FOREGROUND AND BACKGROUND

When drawing lines, filling in polygons, or any other ASTAAD function, then the foreground colour is normally black, and the background colour white (readers may like trying yellow on blue as opposed to black on white) by including appropriate VDU 19 at the start of the program. There is a final switch in ASTAAD2 that affects the drawing colours. To alter this switch, press Shift-f6. In the 'on' state, the drawing will be drawn normally (black on a white background). In the 'off' state, the drawing will take place in the reverse colours. This allows for the 'blank out an area' function, invoked by f8, to fill in a rectangular area. An example of the use of various functions, including reverse colour, is shown in the illustration below. »



EXTENDING THE ASTAAD PROGRAM

The additions to the program have been divided into three sections, to alleviate excessive typing for the user who only wants, say, the screen load/save functions. All or each should be typed into the computer AFTER loading in the original ASTAAD, which must not have been renumbered in any way, since some of the new lines replace lines from the earlier version. Alternatively, save each of the new sections separately using *SPOOL and then use *EXEC to append each section as required to the original ASTAAD. The first section is a common section, and includes the new error handling routine, and the graphics and text window sections. This can be typed in by itself, and the new version of ASTAAD produced saved, and then used. If, however, either of the two main parts is to be used, then the common section MUST be typed in first, then the appropriate main part. Any lines that occur in Part 2 should replace any coincident ones from either Part 1 or Common, and Part 1 should replace any lines from Common. If both sections are required, then Part 2 should be typed in after Part 1.

This month's magazine cassette contains a complete version of ASTAAD2.

LISTING OF COMMON

```
   10 REM Program ASTAAD2
   20 REM Version B1.5
   30 REM Author Jim Tonge/David Fell
   70 MODE Ø
   71 VDU24,4;4;1275;948;18,Ø,129,16
   72 VDU24,8;8;1268;94Ø;18,Ø,128,16
   73 VDU28,Ø,1,79,Ø
   75 DIM OS% 4Ø
   77 ON ERROR PROCerror
  12Ø IF key=9 THEN V=5:X1%=X%:Y1%=Y%:k
=1Ø:Z=4:SOUND17,-1Ø,125,1
  13Ø IF key<129 PRINT CHR$(key):X%=X%+
2Ø*(X%<125Ø)*(key<>9):GOTO11Ø ELSE IF k
ey<14Ø OR key>144 SOUND17,-1Ø,125,1
  15Ø ON J% GOTO 18Ø,16Ø,2ØØ,22Ø,23Ø,23
Ø,17Ø,23Ø,19Ø,21Ø,11Ø,26Ø,25Ø,25Ø,25Ø,2
5Ø ELSE 155
  155 ON J%-16 GOTO 221,222,223,224,225
,226,227 ELSE GOTO 25Ø
  221 REM PROCsave:PROCcursor
  222 REM PROCload:PROCcursor
  223 REM PROCdump:PROCcursor
  224 REM PROCtoggle:PROCcursor
  225 REM PROCfill:PROCcursor
  226 REM PROCdummy:PROCcursor
  227 REM PROCcolour:PROCcursor
  25Ø X%=X%-1Ø*(J%=13)*(X%>1Ø)+1Ø*(J%=1
4)*(X%<1268):Y%=Y%-1Ø*(J%=15)*(Y%>1Ø)+1
Ø*(J%=16)*(Y%<94Ø):PLOTZ,X%,Y%
  365 *FX226,145
  366 *FX227
  367 *FX228
  44Ø VDU4:@%=131Ø82:PRINTTAB(6Ø,1)B%+1
Ø-k;:VDU5:j=J%
  48Ø VDU4:INPUT''"Text? "T$''"Size? (2
 to 125): "S%''"Angle(deg.)?"T%:CLS:VDU5
  65Ø VDU4:INPUT''"Angle of arrow? "W%
:CLS:VDU5
  67Ø PLOTy,-a,-b:PLOTØ,a,b:PLOTy,-c,-d
:PLOTØ,c,d
  71Ø VDU4:INPUT''"Length? "L%''"Angle?
(deg.) "N%:CLS:VDU5
  72Ø LX=L%*COS(RAD(N%)):LY=L%*SIN(RAD(
N%)):X2%=X%+LX:Y2%=Y%+LY:PLOTY,X2%,Y2%:
X%=X2%:Y%=Y2%
  76Ø VDU4:INPUT''"Horizontal Axis?"H%'
'"Vertical Axis?"V%''"No.of sides(3Ø=ci
rcle or ellipse)?"G%:CLS:VDU5
  91Ø VDU4:INPUT''"Radius of circle? "R
%:CLS:VDU5
  96Ø VDU24,X1%;Y1%;X%;Y%;16,24,8;8;126
8;94Ø;
 226Ø DEF PROCerror
 227Ø IF ERR=17 AND NOT INKEY-1 VDU4,12
,5:ENDPROC
 228Ø VDU4,31,6,Ø
 229Ø REPORT:PRINT" at line ";ERL;" has
occurred;"''"Continue? (Y/N)";
 23ØØ IF (GET AND &DF)=ASC"Y" VDU12,5:E
NDPROC
 231Ø *FX4
 232Ø *FX225 1
 233Ø VDU22,7:HIMEM=&7CØØ
 234Ø REPORT:PRINT" at line ";ERL'"That
's all folks!!"
 235Ø END
```

LISTING OF PART 1

```
  221 PROCsave:PROCcursor
  222 PROCload:PROCcursor
  223 PROCdump:PROCcursor
  401 IF J%=17 ENDPROC
  402 IF J%=18 k=0:LOCALk:k=10
 2000 DEF PROCsave
 2010 IF J%<>17 ENDPROC ELSE VDU4
 2020 INPUTTAB(6,0)"Name of file to sav
e to: "$OS%
 2030 CLS
 2040 $OS%="SAVE "+$OS%+" 3000,7FFF"
 2050 LOCAL X%,Y%
 2060 X%=OS%:Y%=X%DIV256:CALL&FFF7
 2070 VDU5:ENDPROC
 2080 DEF PROCload
 2090 IF J%<>18 ENDPROC ELSE VDU4
 2100 INPUTTAB(6,0)"Name of file to loa
d from: "$OS%
 2110 $OS%="LOAD "+$OS%+" 3000"
 2120 X%=OS%:Y%=X%DIV256:CALL&FFF7
 2130 CLS
 2140 PROCsetup:k=10:J%=18:PROCruler
 2150 VDU5:ENDPROC
 2160 DEF PROCdump
 2170 IF J%<>19 ENDPROC
 2180 VDU4
 2190 INPUTTAB(6,0)"Name of screen-dump
to RUN: "$OS%
 2200 VDU12,26,28,0,1,79,0
 2210 $OS%="/"+$OS%
 2220 LOCAL X%,Y%
 2230 X%=OS%:Y%=X%DIV256:CALL&FFF7
 2240 VDU24,8;8;1268;940;
 2250 VDU5:ENDPROC
```

LISTING OF PART 2

```
  224 PROCtoggle:PROCcursor
  225 PROCfill:PROCcursor
  226 PROCdummy:PROCcursor
  227 PROCcolour:PROCcursor
  480 VDU4:T$=FNread:INPUT'''"Size? (2 t
o 125):  "S%'''"Angle(deg.)?"T%:CLS:VDU5
  510 IF E% PROCread ELSE A%=&BF00+ASC(
MID$(T$,C%,1))*8
  810 LOCAL I%,K%,M%,N,C,S,B,D,R,T,Y
  815 IF F% Y=85 ELSE Y=5
  860 IF I%>1 THEN PLOTY,K%,M% ELSE MOV
EX%,Y%:MOVE K%,M%
  865 IF I%>1 AND F% PLOT85,X%,Y%
  875 IF F% PLOT85,X%,Y%
 2360 DEF PROCtoggle
 2370 IF J%<>20 ENDPROC ELSE VDU4
 2380 PRINT'''"Toggling characters to ";
 2390 IF E% E%=0:PRINT"off."; ELSE E%=1
60:PRINT"on.";
 2400 LOCALE%
 2410 PROCdelay
 2420 VDU12,5:ENDPROC
 2430 DEF PROCfill
 2440 IF J%<>21 ENDPROC ELSE VDU4
 2450 PRINT'''"Toggling infill to ";
 2460 IF F% F%=FALSE:PRINT"off."; ELSE
F%=TRUE:PRINT"on.";
 2470 LOCALF%
 2480 PROCdelay
 2490 VDU12,5:ENDPROC
 2500 DEF PROCdummy
 2510 IF J%<>22 ENDPROC ELSE VDU4
 2520 REM THIS IS WHERE YOU COULD INSERT
 2530 REM YOUR OWN ROUTINE.
 2540 REM THE FIRST LINE EXITS IF THE CALL
 2550 REM IS NOT FOR YOU, OR SETS UP THE
 2560 REM TEXT WINDOW IF IT IS.
 2570 REM ON EXIT, USE VDU12,5
 2580 REM AND THEN ENDPROC.
 2590 VDU12,5:ENDPROC
 2600 DEF PROCread
 2610 LOCAL X%,Y%
 2620 X%=OS%
 2630 Y%=X% DIV 256
 2640 ?OS%=ASCMID$(T$,C%,1)+E%
 2650 A%=10
 2660 CALL &FFF1
 2670 A%=OS%+1
 2680 ENDPROC
 2690 DEF FNread
 2700 PRINT'''"Text? ";
 2710 T$=""
 2720 LOCAL I%
 2730 REPEAT
 2740 I%=GET
 2750 IF I%=21 PRINTSTRING$(LENT$,CHR$1
27);:T$="":I%=0
 2760 IF I%=127 AND LENT$ VDUI%:I%=0:T$
=LEFT$(T$,LENT$-1) ELSE IF I%=127 VDU7:I%=0
 2770 IF I% AND I%<>13 T$=T$+CHR$I%:VDU
I%+E%
 2780 UNTIL I%=13
 2790 CLS
 2800 =T$
 2810 DEF PROCcolour
 2820 IF J%<>23 ENDPROC ELSE VDU4
 2830 PRINT'''"Toggling Colours to ";
 2840 IF U% PRINT"off.";:U%=FALSE:PROCo
ff ELSE PRINT"on.";:U%=TRUE:PROCon
 2850 PROCdelay
 2860 VDU12,5:ENDPROC
 2870 DEF PROCoff
 2880 GCOL0,129
 2890 GCOL0,0
 2900 ENDPROC
 2910 DEF PROCon
 2920 GCOL0,128
 2930 GCOL0,1
 2940 ENDPROC
 2950 DEF PROCdelay
 2960 LOCAL T%
 2970 T%=TIME
 2980 REPEAT UNTIL TIME>T%+150 OR ADVAL-1
 2990 ENDPROC
```

A REPORT ON ACORN'S LATEST DEVELOPMENTS

by David A. Fell

Following various rumours, announcements and exhibitions, David Fell now reports on Acorn's latest products and future plans.

ELECTRON LATEST

Following the immediate success of the Electron, Acorn are finding it hard to ensure adequate deliveries of the assorted components required for the assembly of this micro, and as a result, Electrons are in short supply (indeed, we understand Acorn have issued 'Out of stock' posters for use by branches of W.H. Smith).

On the plus side, Acorn are currently working on producing a variety of add-ons for the Electron. These are to be connected to the Electron via an expansion box which will link to the Electron using the rear edge connector. With this box connected to an Electron, the user will then be able to use a printer, joysticks and a ROM expansion board (which should allow for Basic and two other sideways ROMs). The expansion box is now expected to be available about Easter.

A cartridge ROM socket is also planned, in principle similar to those available for machines like the Atari VCS. Acorn believe that many Electron owners will not want to fiddle about too much fitting ROMs, as is the case with the BBC Micro, so they hope that a large amount of software will be released in a format suitable for use with the cartridge ROM system. A disc interface may also be prepared for release at a later date.

ECONET

Acorn have recently been encountering a number of problems in their Econet local area network, and this has even caused unfounded rumours that Acorn were going to drop Econet. Acorn report that they have discovered the cause of these problems, and are now going ahead full speed with their Econet system.

IEEE-488 FULL SPECIFICATIONS INTERFACE

This product has now been released, and we hope to report on it in the next issue of BEEBUG.

CAMBRIDGE RING

Under current development, but probably not available for a while, is an interface to link up BBC micros to the advanced local area network (LAN) called the Cambridge Ring. This communications system is based around a 'ring' that links all stations on the network together. Information is transmitted in 'packets' which are sent around the ring at extremely fast speeds. The Cambridge Ring was co-invented at the University of Cambridge by Dr. Andy Hopper who is a non-executive Research Director at Acorn and who provides, on a part-time basis, help with some of Acorn's advanced projects and development programmes.

WINCHESTER DISC DRIVES

Not ready for release, but certainly coming on in development, is Acorn's Winchester disc system (Winchesters are sealed high capacity disc drives using hard rather than floppy discs). This will be in the form of a 16K ROM (containing the Winchester Filing System, known as WFS), that would be inserted into one of the 'sideways' ROM sockets in the Beeb, and the Winchester disc drive(s) which will be attached to the Beeb via the 1MHz bus. The storage capacity is likely to be of the order of 10M bytes, and the system will cost approximately £1500.

SECOND PROCESSORS

When originally planned, it was envisaged that the BBC Micro would have the ability to use 2nd processors. After lengthy delays, the projected 2nd processors are now appearing on the Acorn stands at exhibitions, and some have been available for testing purposes. Much has been said (particularly by Acorn) on making programs 'Tube' compatible, but this has, until now, seemed a somewhat theoretical requirement. Acorn have delayed the release of the 2nd processors (of which there are three planned at present; a 6502, a Z80 and a 16032) to ensure that, when available, there will (hopefully) be no problems with them. The term Acorn are using is 'extensive in-house testing'. However, several other second processor systems, such as that from Torch, have been available for use with the BBC Micro for many months now.

Basically, a Beeb with a 2nd processor is a complex system in which the 'number crunching' (or the language processing) is done by the 2nd processor, and the input and output functions are carried out by the original 'host' processor. The advantage in having two processors is simple; a two processor system can process data significantly faster than a one processor system. Because of the high speeds involved for data transfer via the Tube (in the order of one megabit per second transfer rates), it is essential that the two processors communicate with each other correctly. In designing the Tube, Acorn have encountered a number of problems with regards to the timing of the inter processor communications, but these, it is reported, have now been resolved. Readers may like to read David Graham's article on the implications of the Tube in BEEBUG Vol.1 No.6.

6502 2ND PROCESSOR

The 6502 2nd processor is now virtually ready for release, and its price tag is expected to be about £195 inclusive of VAT. With the 2nd processor fitted, the user should get 42k of memory available from Basic for his (her) program. There is no need for 16k to be set aside for the operating system, as this still resides in the 'host' processor's memory map. However, there is still a need for some extra operating system code to be present because the Tube needs something to control the communications in each processor. Only about 4k of code is needed for this however, and so there is a further 12k of memory available between where Basic normally resides (&8000 onwards), and where it could reside (&B000 onwards).

The Tube version of Acorn's Basic II has been assembled to reside at &B000, allowing the user to take advantage of the 12K of RAM that would otherwise not be accessible. If not already installed, a DFS (disc filing system) ROM will also be supplied, as this contains some code necessary to control data transfers across the Tube. This code resides in the 'host' processor.

Z-80 2ND PROCESSOR

The Z80 2nd processor is also nearly ready for release. As this will be supplied with the CP/M 2.2 Operating System, it opens the door to a tremendous number of proven CP/M applications packages. Because much of a user's existing software will not work on a Beeb with the Z80 2nd processor, with the exception of Tube compatible Basic programs that do not use machine code, Acorn are supplying a large quantity of software bundled in with the package. This is also designed to make the purchase of a Z80 2nd processor offer the user significant gains over the purchase of other alternatives, and thus provide a very attractive package to the first time small business computer user, at whom this system is particularly aimed.

The Acorn Z80 2nd processor system is, like the 6502 2nd processor, expected to be available in the Spring of this year. The price is likely to be between £350 and £400. Major software supplied with the Z80 2nd processor will include a sophisticated word processing system, a database package and a spreadsheet calculator. The user will also receive a Z80 version of BBC Basic, and another version, Microsoft Basic, that is normally used with CP/M based systems. To complement these, the user is also provided with a systems generator which allows the user to design an applications package by simply specifying it in functional terms. Finally, the user gets a full version of Focus CIS COBOL, the established computer language for business applications.

TESTING OUT YOUR MICRO (PART 1) – SIDEWAYS ROMS

by Hugh D. B. Smith

With an increasing number of BBC micros in use, it is almost inevitable that a number of users will experience problems with their machine's hardware or firmware. To help you carry out initial testing of a suspected faulty system, we will be publishing a series of articles and short test programs. We start the series by looking at sideways ROMs.

In this series we aim to guide you through a set of simple test routines to help isolate some of the problems that might arise with your BBC micro. It is important to note that the utilities offered here, and in later articles, are only a guide and can never replace the sophisticated equipment needed to fully test such a complex machine. As a result, you may well be able to isolate and identify a potentially faulty part of your machine before contemplating further action.

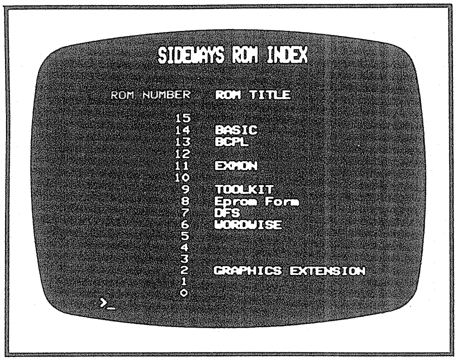
In this first article, we present two programs to test sideways ROMs. For convenience both are not prone to failure but like any type of memory are not infallible. There are two main ways in which a ROM can fail, by failure of the chip itself, or by corruption of one or more bits in the ROM. The latter is more likely in EPROMs (to which this article also applies) and could be caused for a variety of different reasons.

It is the second case that we shall deal with here, since if a ROM breaks down completely, then it is most likely that the machine will not function at all. The usual symptoms of memory bits changing state are errors or bugs in the use of the ROM software, where previously the ROM performed correctly.

To be able to test a ROM, it is necessary to know where it resides inside the machine. In the BBC micro's sideways ROM system, up to 16 ROMs can be used (by using a suitable expansion board). These are numbered from Ø to F (in hexadecimal).

LISTING ROMS

The first program, called simply ROM, will print a list of ROMs present in the machine, and their positions. For a machine with a ROM board these numbers are often printed next to each socket. The four sideways ROM sockets below the keyboard are numbered from 12 to 15 (&C to &F) looking from the keyboard and reading from left to right.


```
SIDEWAYS ROM INDEX

ROM NUMBER  ROM TITLE

    15
    14      BASIC
    13      BCPL
    12
    11      EXMON
    10
     9      TOOLKIT
     8      Eprom Form
     7      DFS
     6      WORDWISE
     5
     4
     3
     2      GRAPHICS EXTENSION
     1
     0
>_
```

Two points should be noted with this. First, if a ROM board is not present the machine employs a system of redundant addressing which means that each ROM will appear in the index up to four times, though only the last four sockets (12 to 15) should be noted. Secondly, the program will only work with ROMs that follow Acorn's specification on initial configuration. If the program appears to crash at a certain ROM title, as happens for example when an Amcom DFS is fitted, it is likely the ROM does not conform and it should be removed from the machine so that ROMs of lower priority may be read.

NOTE. IT IS IMPORTANT TO SWITCH THE MACHINE OFF BEFORE INSERTING OR EXTRACTING ANY ROM.

```
10 REM Program ROM
20 REM Author Hugh D.Brown-Smith
30 REM Version B1.0
```

»

```
   40 REM BEEBUG March 1984
   50 REM Program subject to copyright
   60 :
   70 ON ERROR GOTO 260
   80 PROCassemble
   90 MODE7
  100 ?&71=(PAGE+&700) MOD 256
  110 ?&72=(PAGE+&700) DIV 256
  120 FORT%=0TO15
  130 ?(PAGE+&700+20*T%)=13
  140 NEXT
  150 CALL index
  160 $((15-?&75)*20+PAGE+&700)="BASIC"
  170 PRINT''CHR$141CHR$134SPC5"SIDEWAY
S ROM INDEX"
  180 PRINTCHR$141CHR$134SPC5"SIDEWAYS
ROM INDEX"
  190 PRINT''CHR$129"ROM NUMBER  "CHR$1
33"ROM TITLE"'
  200 R%=0
  210 FORT%=15 TO 0 STEP-1
  220 PRINTCHR$130T%;CHR$131"  ";$(R%*2
0+PAGE+&700)
  230 R%=R%+1
  240 NEXT
  250 END
  260 ON ERROR OFF:MODE7:PRINT'':REPORT
:PRINT" at line ";ERL:END
 1000 DEFPROCassemble
 1010 FORopt=0TO2STEP2
 1020 P%=PAGE+&600
 1030 [OPTopt
 1040 .index
 1050 LDX #15
 1060 STX &70
 1070 .mainloop
 1080 JSR empty
 1090 BMI nextrom
 1100 JSR readtitle
 1110 .nextrom
 1120 CLC
 1130 LDA &71
 1140 ADC #20
 1150 STA &71
 1160 LDA &72
 1170 ADC #00
 1180 STA &72
 1190 DEC &70
 1200 BNE mainloop
 1210 JMP findbasic
 1220 .empty
 1230 LDY &70
 1240 LDA #&80
 1250 STA &F7
 1260 LDA #00
 1270 STA &F6
 1280 JSR &FFB9
 1290 STA &73
 1300 RTS
 1310 .readtitle
 1320 LDA #&80
 1330 STA &77
 1340 LDA #&09
 1350 STA &76
 1360 LDY #0
 1370 STY &74
 1380 .title0
 1390 LDX &77
 1400 STX &F7
 1410 LDX &76
 1420 STX &F6
 1430 LDY &70
 1440 JSR &FFB9
 1450 STA &73
 1460 BEQ lastletter
 1470 CMP #&0D
 1480 BEQ lastletter
 1490 LDY &74
 1500 STA (&71),Y
 1510 INC &76
 1520 INC &74
 1530 CMP #20
 1540 BNE title0
 1550 .lastletter
 1560 LDA #&0D
 1570 LDY &74
 1580 STA (&71),Y
 1590 RTS
 1600 .findbasic
 1610 LDA #&BB
 1620 JSR &FFF4
 1630 STX &75
 1640 RTS
 1650 ]
 1660 NEXT
 1670 ENDPROC
```

ROM TESTER

There are a number of ways to test a ROM, the most common being by a 'Checksum' method. This is simply a case of adding up every byte in a ROM, and comparing the sum with a value which is known to be correct. The problem with such a system is that multiple errors can cancel each other out so producing a correct checksum. A more reliable way is to perform what is known as Cyclic Redundancy Check (CRC). This works through the polynomial shown below for every byte in the ROM, producing a unique value which is almost infallible even for multiple errors.

X^16+X^12+X^5+1

The second program presented here can be used to calculate the CRC of any sideways ROM in the machine. Do note, though that different versions of ROMs will produce different figures and even the smallest modification made by a manufacturer to ROMs of the same version number will result in different values. For this reason it is suggested that on receipt of a new ROM the program is run and the CRC noted. If at a later stage the ROM is suspected of failure, then the ROM should be re-tested and the values compared. Standard Acorn ROM software normally gives values as shown:

Basic I	67FF
Basic II	EC08
DFS 0.90	8C4B

As well as the position of the ROM to be tested, it is also necessary to know its size, either 8k or 16k. This is important since if the wrong value is used then the CRC produced will be incorrect. The size of a ROM is often given in the advertising blurb, but may also be determined by the type number on the chip. In general if the last two digits are 64 then the ROM is 8k, and if the last 3 digits are 128 it is 16k. For example, 2764 and 27128 which are the most common EPROMS used are 8k and 16k respectively. ROMs like BASIC and VIEW are both 16k.

When running this program, the value of PAGE should be set to &1900 or lower if the program is to work correctly. Finally, do note that a CRC which differs from someone else's value does not necessarily mean a faulty component.

```
   10 REM PROGRAM  CRC
   20 REM AUTHOR Hugh D.Brown-Smith
   30 REM Version B1.0
   40 REM BEEBUG March 1984
   50 REM Program subject to copyright
   60 :
   70 ON ERROR GOTO340
   80 MODE7
   90 PRINT''CHR$141;CHR$134SPC5"SIDEWA
YS ROM CRC TEST"
  100 PRINTCHR$141;CHR$134SPC5"SIDEWAYS
 ROM CRC TEST"
  110 PRINT'''''CHR$131"Enter ROM numbe
r (0-F)";
  120 *FX15,1
  130 REPEAT
  140 A=GET
  150 UNTIL (A>47 AND A<58)OR(A>64 AND
A<71)
  160 IF A<58 romnumber%=A-48 ELSE romn
umber%=A-55
  170 PRINTCHR$130"0";~romnumber%
  180 PRINT''CHR$131"Enter ROM size (A=
8k B=16k)";
  190 *FX15,1
  200 REPEAT
  210 B=GET
  220 UNTIL B>64 AND B<67
  230 IFB=65PRINTCHR$130"8k" ELSE PRINT
CHR$130"16k"
  240 size=(B-64)*32
  250 PROCassemble
  260 CALL readrom
  270 PRINT''CHR$134SPC7"CRC value"CHR$
130~(?&71*&100+?&70)
  280 PRINT''CHR$134SPC5"Test another R
OM (Y/N)";
  290 *FX15,1
  300 REPEAT A$=GET$
  310 UNTILA$="Y"OR A$="y"OR A$="N"OR A
$="n"
  320 IFA$="Y"OR A$="y" THEN80 ELSE PRI
NT:END
  330 END
  340 ON ERROR OFF:MODE7:PRINT'':REPORT
:PRINT" at line ";ERL:END
 1000 DEFPROCassemble
 1010 FORQ%=0TO2STEP2
 1020 P%=&1F00
 1030 [OPTQ%
 1040 .readrom
 1050 LDA #romnumber%
 1060 STA &FE30
 1070 STA &F4
 1080 LDX #size
 1090 STX &72
 1100 .calccrc
 1110 LDY #0
 1120 STY &70
 1130 STY &71
 1140 .crc1
 1150 LDA &71
 1160 EOR &8000,Y
 1170 STA &71
 1180 LDX #8
 1190 .crc2
 1200 LDA &71
 1210 ROL A
 1220 BCC crc3
 1230 LDA &71
 1240 EOR #8
 1250 STA &71
 1260 LDA &70
 1270 EOR #&10
 1280 STA &70
 1290 .crc3
 1300 ROL &70
 1310 ROL &71
 1320 DEX
 1330 BNE crc2
 1340 INY
 1350 CPY #&00
 1360 BNE crc1
 1370 INC crc1+4
 1380 DEC &72
 1390 BNE crc1
 1400 LDA #&BB
 1410 LDY #&FF
 1420 JSR &FFF4
 1430 STX &FE30
 1440 STX &F4
 1450 RTS
 1460 ]
 1470 NEXT
 1480 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

COLOUR VIDEO - G.C.Jones

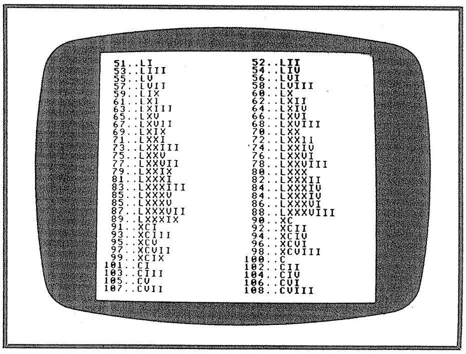
In BEEBUG Vol.1 No.2, we explained how to obtain colour on the composite video output, by soldering a 470pF capacitor in position C58 and joining link S39. On later models of BBC micros, this capacitor is supplied, but link S39 is un-connected. Mr.G.Jones suggests using conductive paint to short the link if you do not feel like soldering in a small space.

Acorn decided not to connect the capacitor (or link), since the picture quality when in colour, is definitely degraded compared with the RGB output.

THE STONEMASON (32K)

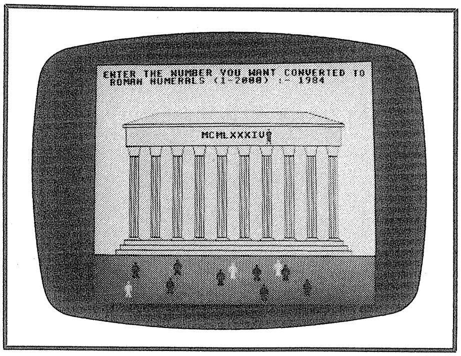
by Brian Knott

The supplement sent out with the November issue of BEEBUG contained a Brainteaser competition which asked entrants to write a program which would convert any given number (in the range 1-2000) into Roman numerals and then list a table of conversions for all the numbers in the specified range. Brian Knott's program won second prize in this competition (see this month's supplement for details of the results), but the graphics and sound that he added to the program were so impressive, that we are printing the program in full. The program is very well written and displays any necessary user instructions on the screen. We think you will appreciate running this program. We certainly did!



```
 51..LI                 52..LII
 53..LIII               54..LIV
 55..LV                 56..LVI
 57..LVII               58..LVIII
 59..LIX                60..LX
 61..LXI                62..LXII
 63..LXIII              64..LXIV
 65..LXV                66..LXVI
 67..LXVII              68..LXVIII
 69..LXIX               70..LXX
 71..LXXI               72..LXXII
 73..LXXIII             74..LXXIV
 75..LXXV               76..LXXVI
 77..LXXVII             78..LXXVIII
 79..LXXIX              80..LXXX
 81..LXXXI              82..LXXXII
 83..LXXXIII            84..LXXXIV
 85..LXXXV              84..LXXXIV
 85..LXXXV              86..LXXXVI
 87..LXXXVII            88..LXXXVIII
 89..LXXXIX             90..XC
 91..XCI                92..XCII
 93..XCIII              94..XCIV
 95..XCV                96..XCVI
 97..XCVII              98..XCVIII
 99..XCIX              100..C
101..CI                102..CII
103..CIII              104..CIV
105..CV                106..CVI
107..CVII              108..CVIII
```

```
   10 REM Program MASON
   20 REM Author  B.L.Knott
   30 REM Version B1.0
   40 REM BEEBUG  March 1984
   50 REM Program subject to copyright
   60 :
  100 ON ERROR GOTO 1990
  110 MODE7:PROCtitle:MODE1
  120 VDU23;8202;0;0;0;
  130 PROCinit
  140 PROCtemple
  150 VDU5:TIME=0:C=0:c%=1:step%=20:D$=
"":d$="MDCLXVI"
  160 FOR F%=0 TO 960STEPstep%
  170 FOR I%=1 TO 11:IF  I%MOD3=0 d%=3
ELSEd%=4
  180 PROCwalk(F%-X%(x%(I%)),Y%(x%(I%))
):NEXT,
  190 *FX15,1
```



```
ENTER THE NUMBER YOU WANT CONVERTED TO
 ROMAN NUMERALS (1-2000) :- 1984
```

```
  200 VDU4,28,0,4,39,0:REPEAT:COLOUR0:C
OLOUR129:INPUT'" ENTER THE NUMBER YOU W
ANT CONVERTED TO  ROMAN NUMERALS (1-20
00) :- "NUM:UNTILNUM>0 AND NUM<2001
  210 PROCroman(NUM)
  220 VDU24,0;0;1279;896;5
  230 PROCshow
  240 VDU4,28,0,4,39,0:COLOUR0:PRINT'"
 Press any key to continue.";:VDU7:Q=GE
T:VDU19,2,6;0;19,1,4;0;
  250 PROClist:VDU30
  260 END
  270 :
 1000 DEF PROCtitle
 1010 VDU23;8202;0;0;0;
 1020 *FX15,1
 1030 PRINTTAB(3,5)CHR$134CHR$157CHR$13
2CHR$141"R O M A N  N U M E R A L S  "
CHR$156TAB(3,6)CHR$134CHR$157CHR$132CHR
$141"R O M A N  N U M E R A L S  "CHR$
156
 1040 PRINTTAB(1,10)CHR$130"This progra
m will convert any number"TAB(0,12)CHR$
130"between 1 and 2000 to its Roman Num
eral"TAB(2,14)CHR$130"equivalent, and w
ill then list all"
 1050 PRINTTAB(4,16)CHR$130"Roman Numer
als from 1 to 2000."TAB(13,20)CHR$131"B
rian L.Knott"TAB(6,23)CHR$134"Press any
 key to continue."
 1060 A=GET:ENDPROC
 1070 :
 1080 DEF PROCinit
 1090 VDU19,3,2;0;19,1,6;0;
 1100 ENVELOPE1,0,0,0,0,0,0,0,126,-20,0
,-20,126,0
 1110 DIM X%(11),Y%(11),x%(11)
 1120 FOR I%=1 TO 11:READx%(I%),X%(I%),
Y%(I%):NEXT
```

»

```
 1130 DATA1,32,100,10,128,160,6,160,180
,8,224,80,4,256,160,11,358,170,2,416,14
0,5,608,180,3,640,105,9,800,170,7,832,9
0
 1140 VDU23,225,24,24,0,60,126,255,255,
189
 1150 VDU23,226,189,189,60,60,60,36,36,
102
 1160 VDU23,227,24,24,0,60,60,126,127,2
53
 1170 VDU23,228,152,60,60,60,60,54,98,6
7
 1180 VDU23,229,24,24,0,60,60,60,60,60
 1190 VDU23,231,24,24,0,60,126,126,126,
60
 1200 VDU23,232,60,60,60,60,60,36,36,10
2
 1210 VDU23,230,24,60,60,60,60,24,24,28
 1220 VDU23,255,255,255,255,255,255,255
,255,255
 1230 ENDPROC
 1240 :
 1250 DEF PROCtemple
 1260 VDU24,0;200;1279;1023;:GCOL0,129:
CLG:VDU24,0;0;1279;199;:GCOL0,131:CLG:V
DU24,100;200;1169;219;:GCOL0,130:CLG
 1270 VDU24,120;220;1149;239;:CLG:VDU24
,140;240;1129;259;:CLG:VDU26:GCOL0,0
 1280 MOVE 0,200:DRAW1280,200:MOVE100,2
00:DRAW100,220:DRAW1170,220:DRAW1170,20
0:MOVE120,220:DRAW120,240:DRAW1150,240:
DRAW1150,220:MOVE140,240:DRAW140,260:DR
AW1130,260:DRAW1130,240
 1290 VDU24,165;264;1104;639;:GCOL0,130
:CLG:VDU26
 1300 FOR I%=0 TO 9:X%=165+(I%*100):MOV
EX%-10,260:DRAWX%,275:DRAWX%+40,275:DRA
WX%+55,260:MOVEX%,275:DRAWX%,600
 1310 FOR J%=1 TO 4:MOVEX%+(J%*10),275:
DRAWX%+(J%*10),600:NEXT
 1320 MOVEX%,600:DRAWX%-10,640:DRAWX%+5
0,640:DRAWX%+40,600:DRAWX%,600
 1330 NEXT
 1340 VDU24,145;640;1124;719;:GCOL0,130
:CLG:VDU26
 1350 MOVE145,640:DRAW1125,640:DRAW1125
,720:DRAW145,720:DRAW145,640:VDU24,130;
720;1139;734;:GCOL0,130:CLG:VDU26
 1360 MOVE130,720:DRAW1140,720:DRAW1140
,735:DRAW130,735:DRAW130,720:MOVE130,73
5:DRAW300,780:DRAW970,780:DRAW1140,735:
GCOL0,129:GCOL0,2:FOR Y%=736 TO 779STEP
4:PLOT&4D,500,Y%:NEXT
 1370 ENDPROC
 1380 :
 1390 DEF PROCwalk(a%,b%)
 1400 GCOLd%,c%:MOVEa%,b%:VDU227+C:MOVE
a%,b%-32:VDU228+C:C=(C+2)MOD4:MOVEa%+st
ep%,b%:VDU227+C:MOVEa%+step%,b%-32:VDU2
28+C
 1410 IF F%=960 GCOL0,(d%-1)MOD3:VDU8,2
26,8,11,225
 1420 ENDPROC
 1430 :
 1440 DEF PROCroman(N%)
 1450 IF  N%>=1000 S%=1:T%=1000 ELSE IF
  N%>=100 S%=3:T%=100 ELSEIF  N%>=10 S%
=5:T%=10 ELSES%=7:T%=1
 1460 REPEATPROCdo(N% DIV T%)
 1470 N%=N% MOD T%:S%=S%+2:T%=T%DIV10
 1480 UNTILS%=9 OR N%=0
 1490 ENDPROC
 1500 :
 1510 DEF PROCdo(n%)
 1520 IF  n%=0 ENDPROC
 1530 IF  n%=4 OR n%=9 D$=D$+MID$(d$,S%
,1)+MID$(d$,S%-1+(n%=9),1):ENDPROC
 1540 IF  n%>=5 D$=D$+MID$(d$,S%-1,1):n
%=n%-5:IF  n%=0 ENDPROC
 1550 FOR I%=1 TO n%:D$=D$+MID$(d$,S%,1
):NEXT
 1560 ENDPROC
 1570 :
 1580 DEF PROCshow
 1590 c%=2:d%=3:start%=252:step%=32
 1600 IF  LEN(D$)MOD2=1 start%=start%-1
6
 1610 MOVEstart%,708:GCOL0,0:VDU255,8,1
0,255
 1620 FOR I=1 TO 1400:NEXT:VDU230,8,11,
229
 1630 FOR I=1 TO 1400:NEXT:GCOL0,2:VDU8
,8,255,8,10,255
 1640 FOR I=1 TO 1400:NEXT
 1650 FOR F%=0 TO 32*((21-LEN(D$))DIV2)
STEPstep%:PROCwalk(start%+32+F%,708):FO
R H%=1 TO 1200:NEXT,
 1660 FOR H=1 TO 1200:NEXT
 1670 GCOL0,2:VDU8,255,8,11,255,8:GCOL0
,0:VDU231,8,10,232,8,11
 1680 PROCspell
 1690 GCOL0,2:VDU255,8,10,255,8:GCOL0,0
:VDU228,8,11,227
 1700 C=0
 1710 Z%=start%+64+(32*(((21-LEN(D$))DI
V2)+LEN(D$)))
 1720 FOR G%=Z%  TO  956STEPstep%:PROCw
alk(G%,708):FOR H%=1 TO 1200:NEXT,
 1730 GCOL0,0:VDU11,255,8,10,255
 1740 FOR H%=1 TO 1200:NEXT:GCOL0,2:VDU
8,8,255,8,11,255
 1750 FOR H%=1 TO 1400:NEXT:VDU255,8,10
,255
 1760 ENDPROC
 1770 :
 1780 DEF PROCspell
 1790 GCOL0,0:VDU231,8,10,232,8,11
 1800 FOR G%=0 TO (LEN(D$)-1):FOR H=1 T
O RND(5)+10:SOUND&100,1,6,RND(4)+1:SOUN
D&101,1,255,1:NEXT
```

»

```
 1810 FOR H=1 TO 2000:NEXT
 1820 GCOL0,2:VDU255,8,10,255:GCOL0,0:V
DU232,8,11,231:PLOT0,-64,-12:PRINTMID$(
D$,G%+1,1);:PLOT0,0,12
 1830 FOR H=1 TO 1400:NEXT,
 1840 ENDPROC
 1850 :
 1860 DEF PROClist
 1870 @%=4:VDU26:COLOUR1:COLOUR130:CLS
 1880 PRINTTAB(5,6)"ALL THE NUMBERS FRO
M 1 TO 2000"TAB(10,8)"WILL NOW BE PRINT
ED"TAB(11,10)"AS ROMAN NUMERALS"
 1890 GCOL0,1:MOVE160,300:DRAW1100,300:
DRAW1100,544:DRAW160,544:DRAW160,300
 1900 PRINTTAB(10,16)"TO STOP THE DISPL
AY"TAB(13,18)"PRESS ANY KEY"TAB(7,20)"T
O CONTINUE PRESS ANY KEY"
 1910 PRINTTAB(4,28)"Press SPACE BAR to
start display.":Q=GET
 1920 VDU28,0,30,39,1:CLS
 1930 COLOUR0
 1940 FOR  J%=1 TO 2000:D$="":PRINTJ%;"
..";:PROCroman(J%):PRINTD$;:IF  POS<20
PRINTSPC(20-POS); ELSEPRINT
 1950 IF  INKEY(0)<>-1 A=GET
 1960 NEXT:VDU15
 1970 ENDPROC
 1980 :
 1990 ON ERROR OFF
 2000 MODE 7:IF ERR=17 END
 2010 REPORT:PRINT" at line ";ERL
 2020 END
```

THE ARIES-B20 MEMORY EXPANSION

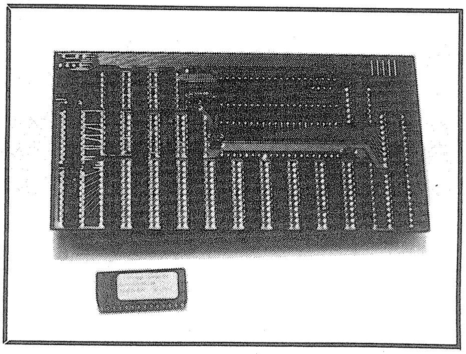
Reviewed by Philip Le Grand

Although the BBC micro is noted for its excellent graphics, as much as 20K of user memory can be used in supporting the graphics, leaving little room for programs and data. Now Computer Consultants Ltd. have produced a 20K memory expansion board for the Beeb. Is this the answer?

Product : ARIES-B20 memory board.
Price : £99.95 (inc. VAT & post)
Supplier: Cambridge Computer Consultants Ltd., FREEPOST, Cambridge, CB1 1BR.

The problem with the BBC micro, is that it shares its memory with the video circuitry, and by the time the operating system, Basic, DFS and any other ROM software has taken its own workspace from the available RAM, there is very little remaining for the user's programs. At first glance, there seems to be no immediate cure for the situation. However, the latest offering from Cambridge Computer Consultants Ltd. seems to answer the problem.

The RAM board arrives ready to be installed in the cpu socket, the cpu (6502 central processing unit) being placed in an equivalent socket on the ARIES board. An EPROM is also supplied, for fitting in a sideways ROM socket which automatically selects the extra RAM. As soon as the computer is switched on, the board becomes active, as it does if you perform a hard break, displaying the message "BBC computer 52K".



The double-sided pcb of the Aries-B20 is very sturdy and its size has been kept to a minimum. The ICs are evenly distributed around the mounting pillar so that there is virtually no strain imposed on the cpu socket. Although the board is simple to fit, you need to be careful about removing the cpu from the Beeb, and lining up the pins in the vacant socket. Once pushed home, the board is very stable. The pins on the mounting pillar are of the type which will not enlarge the holes of the cpu socket, useful if you ever want to use your machine without the board. With the ARIES-B2Ø installed, it is possible to use one of the double density disc controllers reviewed in BEEBUG Vol.2 No.8 issue as well, but none of the ROM extension boards reviewed in BEEBUG Vol.2 No.6. The manual supplied with the package is very clear with plenty of diagrams explaining the installation and use of the board in adequate detail.

The 2ØK of extra RAM operates in parallel with screen memory from &3ØØØ to &7FFF enabling much larger programs to run, including those which are too long to be used with conventional disc-based systems. Thus in any screen mode, a minimum of 25k of memory (28k with a cassette system) is always free for programs and data. The system automatically switches between the screen memory and the new user memory, in a way that is completely hidden from the user.

It is possible to turn the ARIES board off by issuing the command *XOFF. This facility enables you to use EPROM software which cannot make use of the extra RAM, (e.g. WORDWISE, TOOLKIT), though software such as VIEW, BCPL and EXMON can take advantage of this extra memory. The board can be switched on again using *XON.

The switching between the two RAM areas does slow the BBC micro down by about 1%, but it is very unlikely to be noticed by the user, and only affects the screen accesses (e.g. PRINTing and PLOTting). The controlling software requires a page of workspace, and PAGE is automatically set to &FØØ (cassette) or &1AØØ (disc). This is obviously a very small loss compared to the 2ØK gained.

Another way of using the extra memory is for data storage. Cambridge Computer Consultants. have been given the exclusive right by Acorn to use *FX111 to switch between the two RAM areas. This may be used from Basic or assembler to store and access two large areas of data in parallel.

The ARIES-B2Ø board certainly provides an attractive solution to the lack of sufficient memory on the Beeb. The user will need to judge whether he (she) is likely to make sufficient use of the extra memory to justify the cost. On the other hand, it will almost certainly be a cheaper way of gaining extra memory than with any 2nd processor system.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

TURNING OFF ROMS - A.Pemberton

Any sideways ROM socket can be turned off by software control, to prevent it interfering with the running of a particular program. A table at &Ø2A1 - &Ø2BØ holds information about each ROM socket. If no ROM is present, the corresponding location holds the value zero. Thus typing ?&2BØ=Ø will ignore the ROM in socket 15. *HELP will show which ROMs are still recognised by the operating system. This can be extremely useful when ROMs conflict with each other, as in the case of filenames for example. Note also that !&2AD=Ø will 'empty' all four sockets.

LOWER CASE BASIC - A.Armstrong

If you redefine the two characters @ and } by running the following program, you will find all upper case letters will now be printed in lower case, yet they still work as upper case.

```
10 MODE4
20 VDU 23,64,60,60,60,60,60,60,60,,60
30 VDU 23,125,0,0,255,255,255,255,0,0
```

BACH'S CANTATA NO. 147

Programmed by Derek Hufton

The following program provides an interesting implementation of the well known piece of music "Jesu Joy of Man's Desiring" from cantata No.147 by J.S.Bach. The program itself is very short, and will repeatedly cycle through the composition using one of four envelopes, which is selected at random each time the music restarts. While the music is playing, pressing any key will cause the program to terminate and the music to stop.

You will need to take care typing the program in if your micro is to reproduce this music accurately, but the results are, indeed, well worth while in this case.

```
   1Ø REM Program BACHJ
   2Ø REM Author  D.Hufton
   3Ø REM Version B1.1
   4Ø REM BEEBUG  March 1984
   5Ø REM Program subject to copyright
   6Ø :
  1ØØ MODE 7
  11Ø ON ERROR GOTO 32Ø
  12Ø ENVELOPE1,1,1,-1,Ø,1,1,-1,127,-2,
-1,-15,9Ø,5Ø
  13Ø ENVELOPE2,1,Ø,Ø,Ø,Ø,Ø,Ø,6Ø,Ø,Ø,-6
Ø,9Ø,9Ø
  14Ø ENVELOPE3,3,1,-1,1,1,2,1,75,Ø,Ø,-
75,9Ø,9Ø
  15Ø ENVELOPE4,1,Ø,Ø,Ø,Ø,Ø,Ø,127,-1,-1
,-1,9Ø,Ø
  16Ø :
  17Ø FORA=3TO4:PRINTTAB(6,A)CHR$141;CH
R$129;CHR$157;CHR$131;"Cantata No.147
";CHR$156:NEXT
  18Ø PRINTTAB(2,12)CHR$133;"""Jesu Joy
 of Man's Desiring"""
  19Ø PRINTTAB(15,6)CHR$129;"by"
  2ØØ PRINTTAB(12,8)CHR$13Ø;"J.S.BACH"
  21Ø PRINTTAB(1Ø,2Ø)CHR$13Ø;"Programme
d by"
  22Ø PRINTTAB(1Ø,22)CHR$134;"Derek  Hu
fton"
  23Ø VDU23;11,Ø;Ø;Ø;Ø
  24Ø :
  25Ø RESTORE1ØØØ:X%=Ø:e=RND(4)
  26Ø REPEAT:I%=INKEY(Ø)
  27Ø READc,n,d:SOUNDc,e,n,d:X%=X%+1
  28Ø IFX%=93e=RND(4):X%=Ø:RESTORE1ØØØ
  29Ø UNTILI%>Ø
  3ØØ END
  31Ø :
  32Ø ON ERROR OFF
  33Ø MODE 7:IF ERR=17 END
  34Ø REPORT:PRINT" at line ";ERL
  35Ø END
  36Ø :
 1ØØØ DATA&1Ø1,81,4,&1Ø2,81,12,1,129,4,
1,137,4,&1Ø1,145,4,&1Ø2,129,12,1,157,4,
1,149,4
 1Ø1Ø DATA&1Ø1,149,4,&1Ø2,117,12,1,165,
4,1,157,4,&1Ø1,157,4,&1Ø2,97,12,1,177,4
,1,173,4
 1Ø2Ø DATA&1Ø1,177,4,&1Ø2,117,12,1,157,
4,1,145,4,&1Ø1,129,4,&1Ø2,69,12,1,137,4
,1,145,4
 1Ø3Ø DATA&1Ø1,149,4,&1Ø2,89,12,1,157,4
,1,165,4,&1Ø1,157,4,&1Ø2,97,12,1,149,4,
1,145,4
 1Ø4Ø DATA&1Ø1,137,4,&1Ø2,1Ø1,12,1,165,
4,1,157,4,&1Ø1,125,4,&1Ø2,89,12,1,129,4
,1,137,4
 1Ø5Ø DATA&1Ø1,1Ø9,4,&1Ø2,77,12,1,125,4
,1,137,4,&1Ø1,149,4,&1Ø2,1Ø9,12,1,145,4
,1,137,4
 1Ø6Ø DATA&1Ø1,145,4,&1Ø2,81,12,1,129,4
,1,137,4,&1Ø1,145,4,&1Ø2,117,12,1,157,4
,1,149,4
 1Ø7Ø DATA&1Ø1,149,4,&1Ø2,1Ø1,12,1,165,
4,1,157,4,&1Ø1,157,4,&1Ø2,97,12,1,177,4
,1,173,4
 1Ø8Ø DATA&1Ø1,177,4,&1Ø2,117,12,1,157,
4,1,145,4,&1Ø1,129,4,&1Ø2,1Ø9,12,1,137,
4,1,145,4
 1Ø9Ø DATA&1Ø1,117,4,&1Ø2,1Ø1,12,1,157,
4,1,149,4,&1Ø1,145,4,&1Ø2,1Ø5,12,1,137,
4,1,129,4,&1Ø1,1Ø9,4,&1Ø2,1Ø9,12,1,129,
4,1,125,4
 11ØØ DATA&1Ø1,129,4,&1Ø2,81,24,1,145,4
,1,157,4,1,177,4,1,157,4,1,145,4,&1Ø1,1
29,24,&1Ø2,1Ø9,24
```

MACHINE CODE GRAPHICS (Part 2)

by Peter Clease

We continue our series on machine code graphics by looking at the definitions of multi-coloured characters in Mode 5, which is one of the modes used most frequently in commercially produced games.

MODE 5

Mode 5 is a popular mode for graphics games because it offers a nice balance between the amount of memory used, and the number of colours available. It offers the programmer four colours, but uses only 1Øk of memory.

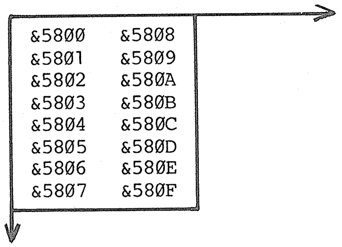
In Mode 5 each byte of memory is used to code four pixels. This is because with four colours possible, each pixel requires two bits to specify the colour of that pixel, and thus with 8 bits to a byte, each byte can represent the colours of four pixels. The table below illustrates the relation between the logical colours, and the two bits allocated to each pixel. (A logical colour may be redefined to a different physical colour by use of the VDU 19 statement.)

Binary	Decimal	Logical Colour
ØØ	Ø	Black
Ø1	1	Red
1Ø	2	Yellow
11	3	White

Table 1. Binary codes for Mode 5 colours.

Since a screen character consists of a matrix of 8 by 8 pixels, one byte in Mode 5 represents half a character line, and two bytes a full character width line.

Below is the byte map of the top left character block on the screen for Mode 5:



&58ØØ	&58Ø8
&58Ø1	&58Ø9
&58Ø2	&58ØA
&58Ø3	&58ØB
&58Ø4	&58ØC
&58Ø5	&58ØD
&58Ø6	&58ØE
&58Ø7	&58ØF

Figure 1. Screen memory addresses for the first character in Mode 5.

To illustrate how to form a four pixel byte to poke into screen memory, we will look at a simple example. Suppose we want to make the first four pixels of the screen black, red, yellow and white respectively, to give a 'striped' appearance.

Using table 1 above, we can see that the four pixel's codes are ØØ, Ø1, 1Ø, 11 respectively for black, red, yellow and white. The way that these 8 bits are combined to form the correct pattern is not quite the way that might initially be expected. Figure 2 illustrates the interleaving process that is required.

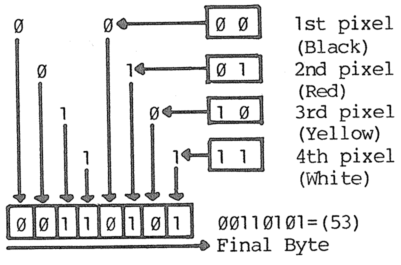


Figure 2. Coding of pixel colours in Mode 5.

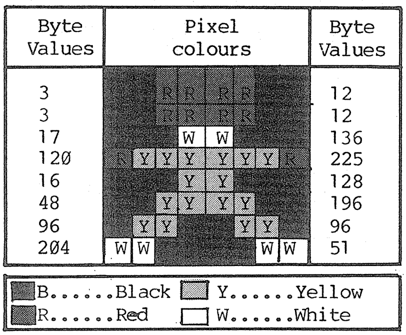
»

The basic method of interleaving the bits for each pixel to form the required byte for Mode 5 (and for the other graphical modes) is as follows:

Take the first (or highest) bit of each pixel, and arrange these from left to right at the start of the byte. If the mode is one which uses more than one bit per pixel, then repeat the process for each bit in turn, working from left to right.

Although this may seem strange at first, it is really quite logical, and once understood, it is usually easy to follow through the process of interleaving bits when designing characters. If it helps, until you have acquired the knack of building up a byte easily, it may be quickest to use the method illustrated in figures 2 and 4.

We will now take the interleaving technique further, and produce a four coloured picture of a man and place it at the top of the screen. Figure 3 shows the man that we are going to build (the letters indicate the colours of each pixel, and the numbers alongside are the coded values to be calculated, ready for poking to the screen).



Byte Values	Pixel colours	Byte Values
3	R R R R	12
3	R R R R	12
17	W W	136
120	R Y Y Y Y Y Y R	225
16	Y Y	128
48	Y Y Y Y	196
96	Y Y Y Y	96
204	W W W W	51

▩ B......Black ▨ Y......Yellow
▩ R......Red ☐ W......White

Figure 3. Designing and coding a four colour character in Mode 5.

Having designed the man, we need to convert each set of four horizontal pixels into one byte's-worth of data to insert into screen memory. This is done in the same way that we converted the earlier example.

The first byte representing the man will contain two black pixels followed by two red pixels. Figure 4 illustrates these four pixels being arranged into a single byte:

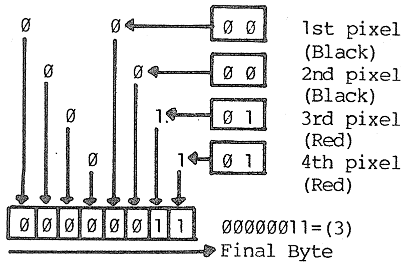


Figure 4. Coding the first byte of the 'man' character.

The remaining fifteen bytes have been coded for you, but it would be a good exercise to code these yourself, and then to check them against the correct data in figure 3.

Now that we have produced the data for poking to the screen, we need a machine code program that will actually do this for us. Program 5, listed below, will accomplish this. Type it in, and try it. The DATA statements at lines 17Ø and 18Ø are the parts of the program that actually determines the shape and colour of the character displayed on the screen. Thus, just changing these lines to a different set of 16 bytes will give a different shape.

```
 1Ø REM PROGRAM 5
 2Ø REM VERSION B1.1
 3Ø REM AUTHOR PETER CLEASE
 4Ø REM BEEBUG MARCH 1984
 5Ø REM PROGRAM SUBJECT TO COPYRIGHT.
 6Ø :
1ØØ PROCassemble
11Ø FOR I=Ø TO 15
12Ø READ I?data
13Ø NEXT
14Ø DELAY=INKEY 2ØØ
15Ø MODE 5
16Ø VDU 23;11,Ø;Ø;Ø;Ø
17Ø CALL CODE
18Ø PRINT'''
19Ø END
2ØØ :
21Ø DATA 3,3,17,12Ø,16,48,96,2Ø4
```

»

```
 220 DATA 12,12,136,225,128,192,96,51
 230 :
1000 DEF PROCassemble
1010 DIM CODE 100
1020 DIM data 15
1030 FOR PASS =0 TO 3 STEP 3
1040 P%=CODE
1050 [
1060 OPT PASS
1070 LDY #0
1080 .loop
1090 LDA data,Y
1100 STA &5800,Y
1110 INY
1120 CPY #16
1130 BNE loop
1140 RTS
1150 ]
1160 NEXT
1170 ENDPROC
```

Program 5 is very similar to the one last month that plotted the space-invader in Mode 4. This time sixteen, instead of eight, numbers must be stored in screen memory to produce the character, because twice as many bits are needed per pixel.

It is a fairly simple matter to design your own characters by working out the byte values to be poked, using the method given above. Remember that program 5 works its way down each column and so the data must be in that order. If you have decided to work out your own data, replace the data at lines 210 and 220, and then run the program. Programs, such as SPRITES from BEEBUGSOFT, are also available which will allow you to design your own characters more easily and which also includes routines to display and move characters on the screen. By way of example, here are some further sets of data that you can try at lines 210 and 220 in place of the original 'man'.

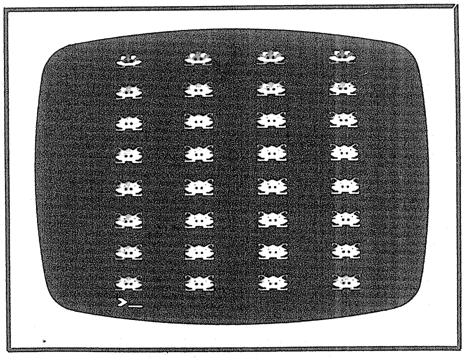
Space Invader.

```
210 DATA 12,2,7,45,7,3,6,12
220 DATA 3,4,14,75,14,12,6,3
```

Strange Fruit.

```
210 DATA 16,0,18,7,15,15,15,6
220 DATA 32,112,132,14,111,111,15,6
```

Once you can define single characters, you can build up larger shapes on the screen by combining several characters together. As an example, we will build up a shape consisting of a block of 4 characters, which will thus be a matrix of 16 by 16 pixels. As a result, there will be a total of 64 bytes to be coded and placed in appropriate Data statements. The task is virtually the same as before, but obviously rather more lengthy to carry out. The following program (Program 6) is similar to Program 5, but places a four character shape on the screen. Again, there are two procedures, PROCassemble, to assemble the machine code, and PROCdraw, to place the shape on the screen. The data in our program will produce a figure that is similar to a space invader, but you should be able to design your own shapes as well. The main program section also includes a loop which will display this colourful shape in a series of different positions on the screen.



```
 10 REM PROGRAM 6
 20 REM VERSION B1.1
 30 REM BEEBUG MARCH 1984
 40 REM AUTHOR DAVID FELL
 50 REM PROGRAM SUBJECT TO COPYRIGHT.
 60 :
100 PROCassemble
110 FOR I=0 TO 63
120 READ data?I
130 NEXT
140 DELAY=INKEY 150
150 MODE 5
160 FOR I=&5800 TO &7C00 STEP 1280
170 PROCdraw(I):PROCdraw(I+80)
180 PROCdraw(I+160):PROCdraw(I+240)
190 NEXT
200 PRINTTAB(0,31);
210 END
220 :
230 DATA 14,11,9,1,3,7,15,3
240 DATA 0,1,11,15,207,139,15,15
```

```
 250 DATA Ø,8,13,15,63,46,15,15
 260 DATA 7,13,9,8,12,14,15,12
 270 DATA 15,120,60,7,3,8,15,15
 280 DATA 13,13,135,240,15,15,12,12
 290 DATA 11,11,30,240,15,15,3,3
 300 DATA 15,225,195,14,12,1,15,15
 310 :
1000 DEF PROCassemble
1010 DIM CODE 200
1020 DIM data 63
1030 FOR PASS= 0 TO 3 STEP 3
1040 P%=CODE
1050 [
1060 OPT PASS
1070 LDY #0
1080 .loop1
1090 LDA data,Y
1100 STA (&80),Y
1110 INY
1120 CPY #32
1130 BNE loop1
1140 CLC
1150 LDA &80
1160 ADC #&20
1170 STA &80
1180 LDA &81
1190 ADC #1
1200 STA &81
1210 .loop2
1220 LDA data,Y
1230 STA (&80),Y
1240 INY
1250 CPY #64
1260 BNE loop2
1270 RTS
1280 ]
1290 NEXT
1300 ENDPROC
1310 :
1320 DEF PROCdraw(P%)
1330 !&80=P%
1340 CALL CODE
1350 ENDPROC
```

TECHNICAL NOTES

The method involved in plotting the four characters is very similar to the method used for plotting a single character, except that for the second character (numbering the top two as 1 and 2, left to right, and the bottom two as 3 and 4) we merely carry on adding one to the contents of the Y register. This is because the address in memory at which the second character has to go is directly after that at which the first character went. For characters 2 and 3, there is a gap in the memory addresses. The program adds an offset to the value stored at &80 and &81. This value is calculated as &140 (the actual difference between the two) minus the contents of the Y register (&20 at this point). This leads to &120 being added. By only adding this amount on, and using the Y register to provide the rest of the offset it means the we can also store our character data in a single block, because the Y register will loop from 0 to 63 in steps of one, and not skip over any values.

Note that the two programs that accompany this month's article store their machine code in areas of memory allocated from Basic via the DIM statement. This is the normal method of designating an area of memory in which to store large pieces of machine code when generated from Basic. The data for the characters is also stored in an area DIMmed from Basic for ease. The amount of memory required for the character data is known precisely at the time of program entry, so only that amount of memory is requested. With the machine code section, however, it is not practical to quickly calculate how many bytes of memory will be required, and so a figure that is generous is used. Note also the use of the READ statement. This is a little known and flexible way of reading data directly into memory.

Users of Basic II may be interested to know of a new group of commands that allow data to be stored in memory more easily than the method used here, especially when developing machine code programs. These are EQUB, EQUW, EQUD, and EQUS (see also BEEBUG Vol.2 No.4). The command of particular interest is EQUB, as this takes a list of single byte values, much as VDU normally does, and inserts them into memory at the current 'program counter'. This allows both the data and machine code to be allocated memory together. It is recommended that the data resides after the machine code.

That is all we have space for this month. In the next article we will look at how to cope with the choice of 16 colours available in Mode 2, and the other graphics modes.

PRINTING YOUR OWN FUNCTION KEY LABELS (16K)

by Tim Powys-Lybbe

The ten user-defined function keys are one of the most useful features of the BBC micro. The short program that we describe here enables you to print your own function key labels on an Epson or similar printer.

This program uses the facilities on the Epson to print function key labels using condensed and emphasized (darker) characters. The program should also work with the Shinwa CP80 printer, and any other printer which is compatible with the Epson. It should be possible to adapt the program to work with many other dot matrix printers. The program as listed will print the key labels for the extended version of ASTAAD and the labels for Colin Lindsay's keyset program (both in this issue). Now those readers with an Epson printer needn't cut up their copies of BEEBUG for the titles.

The program prints nine lines of text. You can put whatever you want in those nine lines by means of the Data statements in lines 4000 onwards. Each Data line corresponds to a line of print and must consist of 11 data items, or else the titles will be printed wrongly. Each line has an initial title of up to 7 characters, which can be used to label the keystrip as a whole, followed by the 10 keys making 11 items in all.

The titles may be any length you want, but will be printed alternately 12 and 13 characters in length. Titles shorter than 12 characters will be centralised above their key, so for clarity you will get better results with short descriptions on multiple lines rather than long titles on a single line. Note that you do not need quotation marks around your titles in the Data statements, while completely blank sections require nothing at all between the commas as in lines 4050 and 4060.

The rows of numbers in the 'Initialise' procedure send control codes to the printer for narrow, enhanced printing and to make the lines closer together. The numbers in the 'Reset' procedure reset the printer back to normal. If you have a different type of printer you may well be able to change these values so that the program will work for your own printer. If you do get the printer tied up in knots then either turn it off and on again or type (for an Epson):

```
  VDU 2,1,27,1,64,3
```

which will reset the printer.

It is possible to have more than 9 lines of print, though only 9 will fit under the perspex strip. To print more, alter line 2010 to:

```
  FOR J=0 TO lines: FOR I=0 to 10
```

where the value 'lines' is the number you want. Then add in the additional data statements after line 4080.

The lines printed are 132 characters long, which forces a carriage return and line feed on the Epson printer. This is why it is vital to include the semi-colon in line 2080, to prevent another line feed from the micro.

```
  10 REM Program FNKEY
  20 REM Version B1.1
  30 REM Author T.F.Powys-Lybbe
  40 REM BEEBUG March 1984
  50 REM Program subject to Copyright
  60 :
  70 MODE7
  80 ON ERROR GOTO 1000
  90 PROCInitialise
 100 PROCTitles
 110 PROCReset
 120 END
 130 :
 140 ON ERROR OFF:MODE 7:REPORT:PRINT"
 at line ";ERL:END
 150 :
1000 DEF PROCInitialise
1010 VDU2:C$=STRING$(7,CHR$32)
1020 PRINT"Put this guide under the pe
rspex screen above the function keys."
1030 VDU13,13,13,13,1,27,1,51,1,24,1,1
5,1,27,1,71
1040 ENDPROC
1050 :
```

»

```
 2ØØØ DEF PROCTitles
 2Ø1Ø FOR J=Ø TO 8:FOR I=Ø TO 1Ø
 2Ø2Ø READ A$
 2Ø3Ø IF I=Ø B$=LEFT$(A$+C$,7):GOTO 2Ø8
Ø
 2Ø4Ø IF I/2=INT(I/2) A$=LEFT$(A$,13) E
LSE A$=LEFT$(A$,12)
 2Ø5Ø A$=STRING$(6-INT(LEN(A$)/2),CHR$3
2)+A$+C$
 2Ø6Ø IF I/2=INT(I/2) A$=LEFT$(A$,13) E
LSE A$=LEFT$(A$,12)
 2Ø7Ø B$=B$+A$:
 2Ø8Ø NEXT I:PRINTB$;
 2Ø9Ø NEXT J
 21ØØ ENDPROC
 211Ø :
 3ØØØ DEF PROCReset
 3Ø1Ø VDU1,18,1,27,1,72,1,27,1,5Ø,13,13
,13,13,13,13,3
 3Ø2Ø ENDPROC
 3Ø3Ø :
 4ØØØ DATA,SCREEN,SCREEN,SCREEN,SOFT CH
ARS,INFILL,DUMMY,REVERSE,,,
 4Ø1Ø DATA,SAVE,LOAD,DUMP,ON/OFF,ON/OFF
,ROUTINE,COLOURS,,,
 4Ø2Ø DATAASTAAD+,,,,,,,,,,
 4Ø3Ø DATA,ASTAAD,DRAW,DRAW,REPEAT,MOVE
,DRAW,LINE,DELETE,DELETE,DRAW
 4Ø4Ø DATA,TEXT,ARROW,POLYGON,POLYGON,,
,,LINE,AREA,CIRCLE
 4Ø5Ø DATA,,,,,,,,,
 4Ø6Ø DATA,,,,,,,,,,
 4Ø7Ø DATAREDKEY,CLEAR TEXT,FREE,ERROR,
CAT,CHAIN"",INSPECT,PAGED,PRINTER,PRINT
ER,BIN TO HEX
 4Ø8Ø DATA,BLUE RUN,MEMORY,LINE,,,MEMOR
Y,LIST,ON,OFF,DEC TO HEX
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

DISC LOCATION - K.Simpson

After accessing a disc, either read or write, location &FØ5 DIV 8 contains the number of files on the disc.

INCREMENTING COUNTERS - R.Rottier

In BEEBUG Vol.2 No.7, we included a hint about COUNT errors. However, the accompanying machine code program, as pointed out by R.Rottier, could be replaced by the following single command:

```
INC &1E
```

*COMMANDS - A.Pemberton

When using the structure IF...THEN *<command>, the THEN is not optional.

WHICH PAGE? - J.Brayshaw

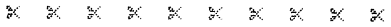
When in terminal mode, using the Teletext adapter, pressing the Copy key shows which page is currently being displayed.

OUT-DENTING IN WORDWISE - S.J.Wilkinson

The text indenting command in WORDWISE (TI) can accept negative values, giving a piece of text jutting out from the main body of text.

SPECIAL SYMBOLS FROM WORDWISE - R.W.Lyne & M.A.Batey

Here are a couple of helpful symbols you can print out in your documents on an EPSON FX type printer. This method sets up a user defined character in each case. The first comes from Mr.R.W.Lyne, for a 'cut here' scissors symbol, use the following codes:

OC27,75,8,Ø,198,17Ø,1Ø8,4Ø,16,4Ø,68,13Ø. 

The second symbol is a telephone, from Mr.M.A.Batey, using the following codes:

OC27,75,9,Ø,112,243,231,255,255,255,231,243,112

MULTIPLE DISC CATALOGUES

by C. C. Chan

We present an extremely useful disc utility that enables you to double, or even quadruple, the number of files that can be stored on a disc using the Acorn DFS. This utility, which can be adapted to both 40 and 80 track units, is simple to use, and will allow much more efficient use of your disc storage.

One of the main snags with the Acorn DFS, as those using it quickly find out, is the 31 files limit on the disc catalogue. This often means that programs and data have to be moved onto another disc and copied back and forth all the time. This can be especially frustrating if you are trying to keep track of a series of documents, different generations of data files, programs, etc. Worst of all, there is a severe wastage of disc space. For an 80 track disc, most of the files would have to be at least 6K in size to be space-efficient. Many files will fall below this size, and most 80 track discs rarely get much above half-full. Similar problems are also likely to arise with 100k discs, though less disc storage will be wasted.

One way around this problem is to create another catalogue on each disc, thereby immediately enabling each disc to hold up to 61 files (one file is required for system purposes). This is done by modifying the existing catalogue to point to only half of the disc, and then creating another catalogue for the other half. This second catalogue is tucked away securely at the end of the disc, inaccessible in normal disc usage, except by addressing the relevant sectors directly. The files and areas held by the first catalogue will be included in the second catalogue as a system file called '*.%sysfl%' which cannot be deleted by the '*DELETE' command. As both catalogues are in the normal DFS format, all the DFS commands can still be used, regardless of which catalogue is currently active.

Two programs are needed, one to set up and initialise the dual catalogues (CAT2), and one to swap between catalogues (SWAP2). The program CAT2 is used just once to create the second catalogue on disc, and will ask you which drive (0, 1, 2 or 3) is to be used. In general, you should do this on a blank disc, although it is possible to create a dual catalogue on a disc already in use, provided it is not more than half full. Check carefully before running CAT2 and make certain that a *COMPACT is done just before the run. The program CAT2 can be saved on any convenient disc, for use when required.

After creating the second catalogue, which divides the disc into two parts, you can switch between the two sections, by running the SWAP2 program. For convenience, you could save a copy of this program in each catalogue that is to be swapped, though this isn't essential. The program will ask for the number of the drive on which the two catalogues are to be swapped, and after swapping over the two catalogues, the files in the new catalogue are compacted to update and tidy up the DFS buffers held in memory. It may be useful to define a function key to swap over catalogues, for example:

```
   *KEY0CH."SWAP2"|M1|M
```

This sets up function key f0 to swap catalogues.

The programs listed here are intended for use with a 40-track disc unit, and some minor but very important changes will be necessary if they are to be used with 80-track units. These changes are listed at the end of this article.

PROGRAM NOTES

A quick word about how the programs work. As you will see from the listings, they hinge around the procedure 'sectoraccess'. The purpose of this procedure is to read or write a specified disc sector to or from a specified memory address. An error check is incorporated to ensure that any disc errors are brought to the attention of the user. You might like to use this procedure in other disc programs of your own.

The 'sectoraccess' procedure takes 1Ø bytes from address &CCØ onwards as the parameter block for a call to OSWORD with A set to &7F (this call is handled by the DFS sideways ROM). The initialiser program (CAT2) reads in the original catalogue, modifies it to point to only half the disc, and then writes it back to its original location on the disc. It then modifies the copy of the disc catalogue in memory to skip past the file areas used by the first catalogue, reduces the total number of disc sectors available by 2 to safeguard 2 sectors at the middle of the disc, and then writes the second catalogue to the last 2 sectors of the disc.

EXTENDED USE

Experienced users may be interested to note that the procedure 'sectoraccess' has several other possible uses. For example, you may 'mark' a disc with certain passwords and attributes in particular sectors which other programs can then check each time they run. This would make it more difficult to copy programs to other discs and making it virtually impossible to dump a program onto tape, thereby adding some measure of software protection. It may also be used to interrogate specific sectors on disc and dump the catalogues for viewing or for security purposes, etc.

By modifying the initialiser program, it is also possible to create catalogues of unequal sizes on a disc, if you wish to keep all the large files in one catalogue and the smaller ones in the other.

These split catalogues have been in use for some time now and no problems have been experienced whatsoever. While it is not a comparable alternative to getting another disc drive, it does provide a true upgrade option for users frustrated by the limited number of files allowed by the Acorn DFS. One of the main points you should bear in mind however, is that each catalogue is effectively the whole disc when you use the '*BACKUP' DFS command. Therefore, you will need to do two backups for each disc with a split catalogue, one for each catalogue.

It is in fact possible to create as many extra catalogues as required on a disc; the initialiser can be modified to create more catalogues at the end of disc storage space, and the swapper modified to select whichever catalogue is required. Two programs that initialise and swap 4 catalogues on an 8Ø track disc are included on this month's magazine cassette. Note that the 4-catalogue swapper requires a spare byte held in the 2nd header sector which means that you must NOT re-title the disc after creating the 4 catalogues. This problem does not arise when using the 2 catalogue swap program.

```
   1Ø REM PROGRAM CAT2 (4Ø TRACK)
   2Ø REM VERSION B1.2
   3Ø REM BEEBUG MARCH 1984
   4Ø REM AUTHOR C.C. CHAN
   5Ø REM PROGRAM SUBJECT TO COPYRIGHT.
   6Ø :
  1ØØ ON ERROR GOTO 2ØØØ
  11Ø @%=1Ø:MODE7
  12Ø PRINTCHR$131;"Dual Catalogue Disc
 Utility."
  13Ø PRINTCHR$131;STRING$(28,"=")
  14Ø PRINT'CHR$13Ø"4Ø track version."
  15Ø DIM buffer% 512
  16Ø PRINT'CHR$134;:INPUT"Create dual
catalogues on drive? "drive%
  17Ø PROCsectoraccess(drive%,Ø,Ø,buffe
r%,"read")
  18Ø PROCsectoraccess(drive%,Ø,1,buffe
r%+256,"read")
  19Ø W%=?(buffer%+262)AND&FØ:?(buffer%
+262)=W%:?(buffer%+263)=&C8
  2ØØ PROCsectoraccess(drive%,Ø,1,buffe
r%+256,"write")
  21Ø W%=?(buffer%+262)AND&FØ:?(buffer%
+262)=W% OR 1:?(buffer%+263)=&8E
  22Ø $(buffer%+8)="%sysfl%":?(buffer%+
15)=&2A:?(buffer%+271)=2:?(buffer%+27Ø)
=Ø:?(buffer%+268)=&FF:?(buffer%+269)=&C
5:?(buffer%+261)=8
  23Ø PROCsectoraccess(drive%,79,8,buff
er%,"write")
  24Ø PROCsectoraccess(drive%,79,9,buff
er%+256,"write")
  25Ø PRINT'CHR$133;"Dual Catalogues Cr
eated O.K."''CHR$131;
  26Ø $&AØØ="COM."+STR$(drive%):X%=Ø:Y%
=&A:CALL&FFF7
  27Ø END
  28Ø :
```

»

```
 1000 DEFPROCsectoraccess(unit%,track%,
sector%,address%,action$)
 1010 LOCALresult%
 1020 ?&CC0=unit%:?&CC5=3:?&CC9=&21
 1030 !&CC1=address%:?&CC7=track%:?&CC8
=sector%
 1040 IFaction$="write"THEN?&CC6=&4B:EL
SE?&CC6=&53
 1050 X%=&C0:Y%=&C:A%=&7F:result%=USR(&
FFF1)
 1060 IF?&CCA THENPRINTCHR$130;">> Fail
ed:sectoraccess;code=";~?&CCA;",results
=";result%:STOP
 1070 ENDPROC
 1080 :
 2000 ON ERROR OFF:MODE 7
 2010 IF ERR<>17 REPORT:PRINTERL
 2020 END
```

```
   10 REM PROGRAM SWAP2 (40 TRACK)
   20 REM VERSION B1.2
   30 REM BEEBUG MARCH 1984
   40 REM AUTHOR C.C. CHAN
   50 REM PROGRAM SUBJECT TO COPYRIGHT.
   60 :
  100 ON ERROR GOTO 2000
  110 MODE7:@%=10
  120 DIM area% 512, buffer% 512
  130 number%=39
  140 PRINTCHR$131"Dual Catalogue Swap
Utility."
  150 PRINTCHR$131;STRING$(28,"=")
  160 PRINT'CHR$130;number%+1;" track v
ersion."
  170 PRINT'CHR$134;:INPUT"Swap catalog
ues on drive? "drive%
  180 PROCsectoraccess(drive%,0,0,area%
,"read")
  190 PROCsectoraccess(drive%,0,1,area%
+256,"read")
  200 PROCsectoraccess(drive%,number%,8
,buffer%,"read")
  210 PROCsectoraccess(drive%,number%,9
,buffer%+256,"read")
  220 PROCsectoraccess(drive%,number%,8
,area%,"write")
  230 PROCsectoraccess(drive%,number%,9
,area%+256,"write")
  240 PROCsectoraccess(drive%,0,0,buffe
r%,"write")
  250 PROCsectoraccess(drive%,0,1,buffe
r%+256,"write")
  260 PRINT'CHR$133;"Catalogues swapped
 O.K."''CHR$130;
  270 $&A00="COM."+STR$drive%:X%=0:Y%=&
A:CALL&FFF7
  280 END
  290 :
 1000 DEFPROCsectoraccess(unit%,track%,
sector%,address%,action$)
 1010 LOCALresult%
 1020 ?&CC0=unit%:?&CC5=3:?&CC9=&21
 1030 !&CC1=address%:?&CC7=track%:?&CC8
=sector%
 1040 IFaction$="write"THEN?&CC6=&4B:EL
SE?&CC6=&53
 1050 X%=&C0:Y%=&C:A%=&7F:result%=USR(&
FFF1)
 1060 IF?&CCA THENPRINTCHR$130;">> Fail
ed:sectoraccess;code=";~?&CCA;",results
=";result%:STOP
 1070 ENDPROC
 1080 :
 2000 ON ERROR OFF:MODE 7
 2010 IF ERR<>17 REPORT:PRINTERL
 2020 END
```

MODIFICATIONS FOR AN 80 TRACK UNIT

The following changes are required to convert the listed utilities to work on an 80-track unit.

SWAP2

Change the 39 at line 130 to a 79. This is the number of tracks that SWAP2 recognises.

CAT2

Change all the PROCsectoraccess calls with the second parameter of 39 to 79.

Line 190 determines the number of sectors to be managed by the catalogue. Change the whole line to read:

```
190 W%=?(buffer%+262)AND&F0:?(buffer%+2
62)=W% OR 1:?(buffer%+263)=&90
```

Line 210 specifies the total space to be held on the new catalogue. Change it to read:

```
210 W%=?(buffer%+262)AND&F0:?(buffer%+2
62)=W% OR 3:?(buffer%+263)=&1E
```

Line 220 of the initialiser assigns the area to be covered by '*.%sysfl%' which should be enough to cover all the file areas of the first catalogue. Change it to read:

```
220 $(buffer%+8)="%sysfl%":?(buffer%+15
)=&2A:?(buffer%+271)=2:?(buffer%+270)=&
10:?(buffer%+268)=&FF:?(buffer%+269)=&8
D:?(buffer%+261)=8
```

OF INTEREST

There are several errors in the disc manual: OSFIND (&FFCE) returns the channel in A, not Y; the format of the 2nd sector of the disc catalogue is incorrect and has the middle order and low order bits of the file's load address, exec address and length transposed.

WHICH FORTH FOR THE BBC MICRO?

A comparative review by John Yale

In last month's issue of BEEBUG, John Yale gave a short introduction to the computer language FORTH. This month, John provides a comparative review of the three implementations of FORTH now readily available for the BBC micro.

FORTH is a comparatively new computer language that has rapidly achieved a cult following among many programming enthusiasts. It differs from languages like Basic in that writing FORTH programs involves extending the language itself to meet the application, rather than using the language as provided to describe a solution (as happens with Basic). A measure of the interest in FORTH is the availability of at least one popular micro that offers FORTH as its standard language. Versions of FORTH are also available for many other micros and there are now three implementations of FORTH available for the BBC micro.

Acornsoft FORTH and r q FORTH (supplied by Level 9 Computing) are cassette or disc FORTHs based on the Fig (FORTH Interest Group) FORTH model modified to meet the FORTH-79 standard.

JWB FORTH supplied by HCCS Associates differs from the other two in that it is supplied in an 8k byte EPROM for fitting in the computer. JWB FORTH is based on the Fig FORTH model but without the FORTH-79 modifications. HCCS explain that as the majority of applications are supplied by Fig this should make things easier for new users. However, this will only be the case if you already have a quantity of Fig software, as priority is now given to publishing FORTH-79 programs.

Full details of prices, suppliers and recommended books are given at the end of this review.

EDITOR

FORTH source code is normally stored in 1k byte blocks or screens. Acornsoft FORTH and JWB FORTH both provide the standard Fig line and string editor for editing these screens. Although this is a powerful editor when mastered, it is difficult for the beginner to use. Use of the cursor and Copy keys does however allow editing in a similar manner to Basic.

Acornsoft FORTH allows two screens in memory at one time and JWB FORTH allows one. As each screen is typed in, compiled and tested it must be saved to tape before the next screen can be entered. This does cause problems when a screen in the middle of an application needs changing.

In contrast r q FORTH adopts a different approach to block storage and editing. Firstly a selectable number of blocks is kept in memory for instant recall at any time, saving to tape only being required at the end of a session (or more frequently for security). The only disadvantage to this scheme is that it uses a large amount of memory which might be required for graphics. However, as the number of blocks stored is configurable, the user may reduce this to say two blocks and revert to the scheme used by the other systems under review. Secondly r q FORTH uses a screen size of 512 bytes organised as 16 lines of 32 characters instead of 64 characters as used by the other systems. This allows a complete screen to fit onto a Mode 7 display but unfortunately it does not comply with the FORTH-79 standard which requires 1024 byte blocks. Editing is performed by moving the cursor to the required place on the screen and typing away. A set of control keys is also provided to delete text or move it around the screen via a holding buffer. This editor is much easier to use than the Fig one, and will be found to be more powerful than the Basic editor supplied as part of the BBC computer.

BENCHMARKS

The December 1982 issue of Personal Computer World has a set of Benchmarks for FORTH implementations. In general JWB FORTH is slightly slower in execution, though the differences between the three systems are not significant.

CASSETTE SYSTEM

All three systems can save blocks on

cassette. As mentioned above, Acornsoft FORTH allows two screens in memory at one time and JWB FORTH allows only one. Thus as each screen is entered and debugged, it must be saved to tape using the built in block save facility. r q FORTH uses the standard MOS *SAVE command (accessed from within FORTH) to save all the block buffers in one cassette file. In practice this system is much easier to use.

DISC SYSTEM

When FORTH is used with discs all the screens are kept on the disc, only being brought into memory as required for editing or compiling. This system which is known as 'virtual memory' is transparent to the user, it seeming as if all the blocks are continuously available (except for the disc activity as blocks are swapped in and out).

Acornsoft disc FORTH is supplied on a protected disc, but this has little effect and it is possible to produce a backup copy of your FORTH system as it is easily transferred from memory to a new disc by *SAVEing from PAGE to HERE (this also works with the cassette version). I found the disc access time of 3 - 4.5 seconds per block rather long but this does depend on the DFS in use. Blocks are stored nine to a file which allows 90 blocks on a 100k disc.

A disc version of r q FORTH is supplied on the back of the cassette. Block reading and writing is much faster than with Acornsoft FORTH as the disc filing system is not used. Instead sectors are accessed directly via OSWORD calls, giving block read times in the range 0.7 to 1.6 seconds.

The JWB FORTH EPROM will also work with the BBC disc filing system but as each block is stored as a separate file this means a maximum of 31 blocks on a disc due to the DFS file limit. Thus two thirds of the disc is wasted. This system also does not support the important FORTH word BLOCK.

ROM SYSTEM

JWB FORTH is the only one of the three packages currently being supplied as a ROM system (actually in EPROM). Fitting instructions are supplied with the EPROM which is selected by *FORTH.

Acornsoft FORTH should be available in ROM later this year.

Level 9 Computing have no plans at present for a ROM version of r q FORTH.

DOCUMENTATION

Acornsoft FORTH is not supplied with any documentation and requires separate purchase of the book 'FORTH on the BBC Microcomputer'. This is suitable for beginners and advanced users alike and is highly recommended.

r q FORTH is supplied with a 72 page A5 manual containing a complete glossary of all the words in the system, configuration details, source code of the editor etc. This manual contains all the information the experienced FORTH user requires to use the system, but it is not a good introduction for the beginner. Also supplied by Level 9 is an A4 summary card giving brief details of all the words in the system, configuration, editing, saving etc.

JWB FORTH is supplied with a 37 page A4 manual containing a glossary, sections on Graphics and Sound, System description and some demonstration programs. The review copy glossary was rather difficult to read as the first few letters of each line were lost in the binding. This manual is again not an introductory text and HCCS will be introducing a separate book 'Welcome FORTH' in the near future.

A new book 'The Complete FORTH' by Alan Winfield will provide a suitable introduction to any of the FORTH systems reviewed here, particularly the FORTH-79 standard, as this is the dialect used throughout the book. All the basic FORTH techniques are covered in the book's 130 pages, including the definition of new defining words with DOES>. Some of the chapters have exercises, and answers are given at the back of the book. Two complete FORTH programs are also given as examples.

'Starting FORTH', the standard work on FORTH covers more ground with more detail and is highly recommended, but at less than half the price 'The Complete FORTH' is very good value.

ASSEMBLER

Acornsoft FORTH provides an assembler package in source code form on tape or disc which may be added to the system by compiling it and saving a copy of the extended system.

»

The other two systems do not provide an assembler as part of the standard system, but allow access to the machine operating system via CALL (r q FORTH) or *FX and OSWORD (JWB FORTH). Of these, CALL is the more powerful, allowing access not only to OSBYTE and OSWORD but also to file routines such as OSFILE etc. Level 9 Computing also provide a separate r q FORTH Toolkit which includes an assembler.

COMPATIBILITY

Whilst the standard FORTH words are the same in both systems designed to the FORTH-79 standard, the situation is not so good where new words have been defined to use some special feature of the BBC micro. An example is the word used to send the rest of the command line to the Operating System, equivalent to the Basic use of '*'. Thus we have OS' (Acorn), *MOS (r q) and MONITOR (JWB). This doesn't need too much conversion but consider a word with three parameters to provide the equivalent of, say, PLOT. This could be K X Y PLOT, X Y K PLOT or even Y X K PLOT and there are good reasons for all three choices. I would suggest that in such cases as this, all authors should keep to the same order as in the equivalent Basic command, i.e. K X Y PLOT. This is not always the most convenient for FORTH, but at least it would be standard.

None of the different cassette or disc systems reviewed are compatible with each other. However, all three can compile Wordwise files from disc if Return is used at the end of each line.

ERROR MESSAGES

Acornsoft and JWB FORTH use error numbers whereas r q FORTH uses English messages for all errors (eg. 'no such block' rather than '# Msg no 6').

When an error occurs in the loading of a source block it is not always apparent at what point in the block the error has occurred. Typing 'WHERE' in Acornsoft FORTH displays the screen number and the line in error with an arrow pointing at the point in the line where the error was detected. Typing 'WHERE' in r q FORTH enters the screen editor with the cursor at the point in the block where the error occurred.

UPGRADES

Level 9 are also now supplying a FORTH toolkit containing an assembler, turtle graphics package, decompiler (allows you to see the definition of any word in the dictionary), double precision extension and various other extras.

CONCLUSIONS

Of the systems reviewed, r q FORTH stands out as the best system for cassette-based use, and the r q FORTH Toolkit is very good value at £1Ø.

For a disc-based system the choice is between Acorn and r q FORTH which both give good performance.

A ROM-based FORTH has obvious advantages, particularly for cassette use. The HCCS ROM is available now but has several drawbacks, not being to the FORTH-79 standard, having a poor cassette system and with only restricted disc use.

SUPPLIERS AND PRICES

(all prices include VAT)

Acornsoft Ltd, c/o Vector Marketing Ltd, Denington Industrial Estate, Wellingborough.
FORTH cassette £16.85
FORTH on the BBC Microcomputer £7.5Ø

Level 9 Computing, 229 Hughenden Road, High Wycombe, Bucks. HP13 5PG.
r q FORTH £15.ØØ
r q FORTH Toolkit £1Ø
Starting FORTH (book) £14.35

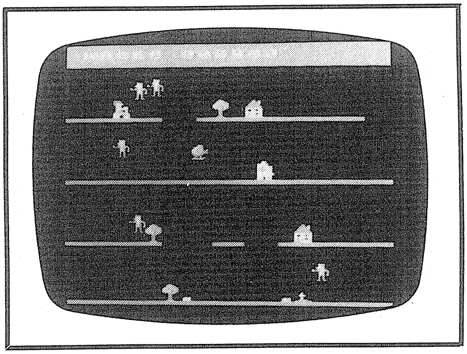
HCCS Associates, 533 Durham Road, Low Fell, Gateshead, Tyne and Wear NE9 5EY.
JWB FORTH EPROM £34.72
Welcome FORTH (book) £6.75

John Wiley & Sons Ltd. (Publisher)
The Complete FORTH, A.J.Winfield £6.95
ISBN Ø-9Ø51Ø4-22-6

The above may also be available from computer shops, specialist book shops and major chain stores such as W.H.Smith.

THE LATEST SOFTWARE REVIEWED

Title	: Daredevil Dennis
Supplier:	Visions Software
Price	: £7.95
Reviewer:	Alan R. Webster
Rating	: ***



Dennis is a rather active stuntman, whose aim in life is to make as much money from the movies as possible. Each stunt he performs earns him big money, but there are several risks that have to be taken.

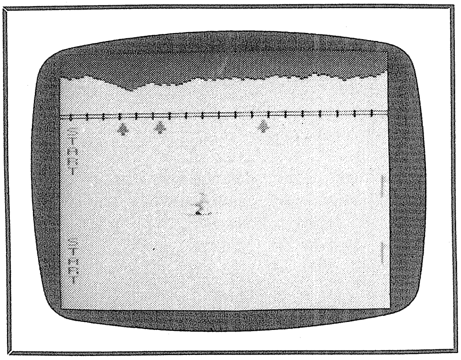
There are three different scenes. The first requires Dennis to ride a motorbike, leaping over houses and engaging in other dangerous pursuits. Among the stunts in the second scene, he has to jump a light-house with a wetbike, and in the third, his stunts are all performed on skis.

The game has smooth, fast graphics and uses over thirty multicoloured 'sprite' characters. It has an interrupt driven sound routine, so that you have constant music while loading. There are also six skill levels to choose from, and each level has six intermediate levels, making thirty-six in all.

The only reason that this game gets three stars instead of four is that a few minor programming bugs still remain, occasionaly spoiling your enjoyment of the game.

Title	: Slalom
Supplier:	R H Software
Price	: £7.95
Reviewer:	Mike Williams
Rating	: ****

Maybe your imagination has been fired recently by watching the skiing in the winter Olympics on television. Perhaps you would like to have a go yourself. Well now you can, by playing Slalom, one of the best new games on the market. From the starting point, you weave your way through the gates and down the course to the finish. The conditions are often icy and the occasional large snowball provides an additional hazard that will often be your undoing. And your reward for completing the course? You start from the top again, but this time the gates are closer together, the twists and turns tighter, and the course is now really difficult.



Crashes are quite spectacular as the skier tumbles down the slope all arms and legs. This game offers superbly smooth and fast graphics, with a high score table, but maybe there is just one little niggle - even at its easiest, Slalom is perhaps just a little too hard.

Titles	: The Pen and the Dark
	My Secret file
Supplier:	Mosaic Publishing
	Distributed by John Wiley
Prices	: £9.95 each
Reviewer:	Mike Williams
Ratings	: *** and ***

Both of the titles reviewed here are described by the publisher as 'Bookware' in that they both combine together as a single package a book and a computer program (supplied on cassette). The Pen and the Dark is an adventure game based upon a science fiction short story which is part of the package, while 'My secret file' provides the means of storing lots of personal data on a computer, and is based upon the Puffin book of the same name, again included as part of the package.

THE PEN AND THE DARK

As a science fiction short story this is indeed very short, and although it is based on an intriguing idea, the story lacks substance. The adventure game, which is broadly based on the book, seemed much more interesting once you accept the role playing that it entails. In the computer game you act out the part of the book's main character. The edition of the book included in the package has clearly been specially produced for use with the adventure game, and contains full instructions for loading and running the game, including the facility of saving the game at any stage for reloading later. In general, I suspect that this package will appeal far more to teenagers than to adults, and this is particularly true of the story. The adventure game is quite extensive, and being written in Basic, you can search through the coding for addituonal clues if you get really desperate.

MY SECRET FILE

My Secret File was originally published as a Puffin Book. The book is written for young teenagers and provides the space, and lots of ideas, for the youngster to build up an extensive personal data-base about himself, his family and friends. In this respect the book is admirable, provided that it is treated in the light hearted way intended.

I was less happy about the computer version provided. The book is the original Puffin and thus the additional instructions for using the program cassette are supplied as a separate leaflet. In some cases the information supplied is inaccurate or incomplete, and the section on saving and reloading your data file was particularly poor (as was the program's screen display at this point). Regrettably, the program more or less mirrors the book, providing the means to store and subsequently redisplay the information recorded, but there is no attempt to exploit the computer's capacity for information retrieval or for reordering of the information stored. Although any data stored in the computer is easily changed I wonder how many children would continue to do this after the initial novelty has worn off.

With more thought and development this particular package could have provided a really exciting bridge between home, education and the serious world of computing. Unfortunately, the opportunity has been lost by tying everything to the limitations of the original book format. None the less, this is a good combination, apart from the less than adequate program instructions provided.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

SHEPPARD SCALE - Andrew Civil

The following program plays a succession of chords on the Sheppard tone scale. This appears to rise endlessly and is prone to drive you insane!

```
10 REPEAT FOR P=0 TO 11
20 SOUND 1,-P,P*4,8
30 SOUND 2,-11,P*4+48,8
40 SOUND 3,P-11,P*4+96,8
50 NEXT:UNTIL FALSE
```

COMPACT FUNCTION KEY DEFINITIONS (16K)

by Colin Lindsay

Only a limited amount of memory is available for storing function key definitions. In many applications the use of this space is at a premium. In this article, Colin Lindsay shows you how to compact your function key definitions and hence maximise the use of this space.

When defining any of the function keys, there are in fact three ways in which Basic keywords can be represented; as the full keyword, in its abbreviated form or as a token. Following the '*' command, Basic does not automatically tokenise Basic keywords, which are therefore stored as strings of ASCII characters.

Substantial savings, however, can be made by using Basic tokens rather than keywords in "red key" definitions, and it is possible to do this if the correct format is used. The appropriate token value for certain keywords can be built up from a few symbols whose total value is taken by the computer to be the value for the keyword itself. This accumulated value is stored in memory using only a single byte for most keywords. This process allows function key definitions to be compacted even further, maximising the use of the limited amount of memory allocated for this purpose.

As an example, the keyword 'RENUMBER' (token value &CC in hex) can be built up from '|!' (ASCII value &80) and 'L' (ASCII value &4C). The computer adds these values and interprets the symbols as having an ASCII value of &CC. Hence, a key definition of

```
     10 *KEY0 |!L|M
```

is equivalent to

```
     10 *KEY0 RENUMBER|M
```

or

```
     10 *KEY0 REN.|M
```

and occupies only 2 bytes in the first case compared to 9 bytes and 5 bytes for the second and third examples (because 'RENUMBER' and 'REN.2' following a * command are not tokenised but stored as ASCII character strings).

Using this technique, programming many or all of the function keys will save valuable space in the 256 byte buffer, though at the expense of rendering much of the code unintelligible!

The following list includes all the keywords that can be "shortened" to a single byte in key definitions. To use this method type |! followed by the character shown after the keyword, in the key definition. Remember that the character "|" is produced by the key next to the cursor left key.

AND	(AND=&80 in hex so type nothing)				
AUTO	F	BPUT	u	CALL	v
CHAIN	W	CHR$	=	CLEAR	X
CLG	Z	CLOSE	Y	CLS	[
COLOUR	{	DATA	\	DELETE	G
DIM	^	DRAW	_(i.e. underline)		
END	`	ENDPROC	a	ENVELOPE	b
EOF	E	EVAL	(ie space-bar)		
EXP	!	EXT	"	FALSE	#
FN	$	FOR	c	GCOL	f
GET	%	GET$	>	GOSUB	d
GOTO	e	HIMEM(L)	S	IF	g
INKEY	&	INKEY$	?	INPUT	h
INSTR(	'	INT	(	LEFT$(	@
LEN	)	LET	i	LIST	I
LN	*	LOAD	H	LOCAL	j
LOG	+	LOMEM(L)	R	MID$(	A
MODE	k	MOVE	l	NEW	J
NEXT	m	NOT	,	OLD	K
ON	n	OPENOUT	.	PAGE(L)	P
OPENIN	-(ie negative operand)				
PI	/	PLOT	p	POINT(	0
POS	1	PRINT	q	PROC	r
RAD	2	READ	O	REM	P
REPEAT	u	RENUMBER	L	REPORT	v
RESTORE	w	RETURN	x	RIGHT$(	B
RND	3	RUN	y	SAVE	M
SGN	4	SIN	5	SOUND	T
SQR	6	STOP	z	STR$	C
STRING$(	D	TAN	7	TIME	Q
TO	8	TRACE	|	TRUE	9
UNTIL	}	USR	:	VAL	;
VDU	o	WIDTH	~		

»

Note that the characters given for HIMEM, LOMEM and PAGE apply only when assigning values TO these system variables (contrary to the information in the User Guide page 483 which gives the tokens for these variables reversed).

The following program fills the red-key buffer using the token method and shows just how powerful the user-defined keys can become. It has been used for program editing (in mode 6) and is included mainly to demonstrate the packing technique.

In the above form the keys can be loaded in 256 bytes (just!). Using abbreviated keywords the same definitions would occupy almost 25% more space and using full-length keywords would take up 40% more space and, of course, wouldn't fit!

You will need to take care when typing in the program. No error trapping has been included, as errors in function key definitions will only be revealed when a key is first pressed after running this program. Do not be tempted to add any extra spaces in the key definitions or the buffer area will overflow and the error mesage 'Bad key' will be displayed. The key functions are identified by REM statements preceding each definition. For those with a suitable printer, a key strip for this program is included as one of the examples in the article on printing function key labels elsewhere in this issue.

```
   10 REM Program REDKEY
   20 REM Version B1.1
   30 REM Author Colin Lindsay
   40 REM BEEBUG March 1984
   50 REM Program subject to copyright
   60 :
  100 REM Flush red-key buffer
  110 *FX18
  120 REM MODE6/BLUE BACKGROUND/RUN
  130 *K.0|!k6:|!o19;4;0;:|!y|M
  140 REM MEMORY LEFT
  150 *K.1|!^P%-1:|!qH.-P%|M
  160 REM AUTO LIST OF LINE IN ERROR
  170 *K.2C$="L."+|!CERL+|!=13:A%=138:X
%=0:|!cL=1|!8|!)C$:Y%=ASC(|!AC$,L,1)):|
!V&FFF4:|!m|M
  180 REM CATALOGUE
  190 *K.3*.|M
  200 REM CHAIN""
  210 *K.4|!W""|M
  220 REM INSPECT MEMORY
  230 *K.5|!h"From",M$:M=|! M$:|!cX=M|!
8 M+99:|!q~?X" "|!=?X:|!m|M
  240 REM PAGED LIST
  250 *K.6|L|!I|N|M
  260 REM PRINTER ON
  270 *K.7|!o2|M
  280 REM PRINTER OFF
  290 *K.8|!o3|M
  300  REM CONVERT BINARY TO HEX, THEN
DECIMAL TO HEX
  310 *K.9T=0:|!h"No=",A$:L=|!)A$:|!cX=
L|!81S.-1:T=T+(2^(L-X))*|!;(|!AA$,X,1))
:|!m:|!q~T:|!h,Z:|!q~Z|M
  320 REM OLD/MODE6/RED/PAGED LIST
  330 *K.10|!K|M|!k6|M|!o19;1;0|M|N|!I|M
  340 PRINT'"Function keys now defined."
  350 END
```

POINTS ARISING

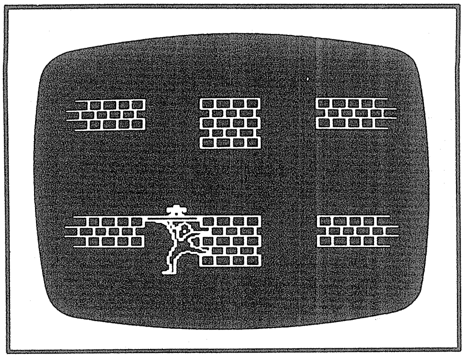
MACHINE CODE GRAPHICS

Unfortunately a number of minor inaccuracies occured in the first article in this series, which appeared in the Jan/Feb issue of BEEBUG. The byte map for mode 2 on page 15 showed the first character block on line two as 'block 41'. Mode 2 is a 20 character mode and this should have been marked 'block 21'. The two line example on page 16 specified Mode 0 when Mode 4 should have been used. Lastly, the value in line 50 of the program on page 17 should have been &5800 and not &5820 as printed. We are sorry for these errors and have double checked this month to ensure there are no inaccuracies in this month's article.

THE MANHOLE GAME (32K)

by E. Christie

The Manhole Game combines very good graphics and sound in a fast moving game to test your reactions. You are the guardian of the Catacombs below the streets of Cheltenham. Armed with only a single manhole cover it is your job to prevent the populace from dropping through four holes in the road. To do this you need to keep your fingers delicately poised above the E, D, I and J keys, and press the right one at the right time. Before the game starts you may also select a degree of difficulty. Level 3 is not for the feint of heart.



PROGRAM NOTES

When you play the game you may be surprised at the speed with which the screen responds to your key press. This is achieved through a particularly useful technique. The underground man is in fact drawn four times at the start of the game in four different logical. colours. But these are all set to physical colour black. You therefore cannot see them. The PROCKEY procedure then sets one of these to colour green according to whichever key has been pressed. Because the object has already been drawn it appears almost instantaneously. Pressing another key extinguishes that position and lights another. This is all performed using the VDU19 call. See the User Guide page 382 for further details.

```
   1Ø REM Program MANHOLE
   2Ø REM Version B1.6
   3Ø REM Author E.Christie
   4Ø REM BEEBUG March 1984
   5Ø REM Program subject to Copyright
   6Ø :
  1ØØ VDU23,224,24,24,255,189,189,36,36
,1Ø2:C$=CHR$255:D$=CHR$13Ø
  11Ø ON ERROR GOTO 238Ø
  12Ø :
  13Ø REPEAT
  14Ø MODE 7
  15Ø PROCstart
  16Ø MODE2:VDU23,1,Ø;Ø;Ø;Ø;:PROCfour
  17Ø T=INKEY(1ØØ)
  18Ø PROCrandom
  19Ø REPEAT
  2ØØ PROCtwo
  21Ø PROCkey
  22Ø IF D>=Ø THEN D=D-Ø.2
  23Ø UNTIL miss>9
  24Ø PROCend
  25Ø UNTIL FALSE
  26Ø END
  27Ø :
 1ØØØ DEF PROCkey
 1Ø1Ø B$=INKEY$(2)
 1Ø2Ø IF B$<>"" THEN A$=B$ ELSE 11ØØ
 1Ø3Ø IF B$<>"E" AND B$<>"I" AND B$<>"D
" AND B$<>"J" THEN 11ØØ
 1Ø4Ø FORI=2TO5:VDU19,I,Ø,Ø,Ø,Ø:NEXT
 1Ø5Ø IF B$="E" VDU19,2,2,Ø,Ø,Ø
 1Ø6Ø IF B$="I" VDU19,3,2,Ø,Ø,Ø
 1Ø7Ø IF B$="D" VDU19,4,2,Ø,Ø,Ø
 1Ø8Ø IF B$="J" VDU19,5,2,Ø,Ø,Ø
 1Ø9Ø *FX 15,Ø
 11ØØ ENDPROC
 111Ø :
 112Ø DEF PROCtwo
 113Ø GCOLØ,6:VDU5:MOVE X,Y+5
 114Ø PRINTCHR$224
 115Ø FORK=1TOD:NEXT
 116Ø IF Y=79Ø AND X>3ØØ AND X<42Ø AND
A$<>"E" THEN 132Ø
 117Ø IF Y=79Ø AND X>3ØØ AND X<42Ø AND
A$="E" THEN 126Ø
 118Ø IF Y=79Ø AND X>7ØØ AND X<82Ø AND
A$<>"I" THEN 132Ø
 119Ø IF Y=79Ø AND X>7ØØ AND X<82Ø AND
A$="I" THEN 126Ø
 12ØØ IF Y=43Ø AND X>3ØØ AND X<42Ø AND
A$<>"D" THEN 132Ø
 121Ø IF Y=43Ø AND X>3ØØ AND X<42Ø AND
A$="D" THEN 126Ø
```

»

```
 122Ø IF Y=43Ø AND X>7ØØ AND X<82Ø AND
A$<>"J" THEN 132Ø
 123Ø IF Y=43Ø AND X>7ØØ AND X<82Ø AND
A$="J" THEN 126Ø
 124Ø SOUNDØ,-15,3Ø,1:SOUNDØ,-15,7Ø,1
 125Ø GOTO128Ø
 126Ø score=score+1
 127Ø SOUND1,-15,5Ø,3
 128Ø GCOLØ,Ø:MOVE X,Y+5 :PRINTCHR$224
 129Ø IF DI=1 THEN X=X+6Ø ELSE X=X-6Ø
 13ØØ IF X>115Ø OR X<5Ø THEN PROCrandom
 131Ø GOTO141Ø
 132Ø SOUND1,-15,1ØØ,1:SOUND1,-15,15Ø,1
 133Ø SOUND1,-15,8Ø,1:SOUND1,-15,5Ø,1
 134Ø SOUND1,-15,1ØØ,1
 135Ø miss=miss+1:IF miss>9 THEN 141Ø
 136Ø GCOLØ,Ø:MOVEX,Y+5:PRINTCHR$224
 137Ø GCOLØ,6:MOVEX,Y:PRINT"*"
 138Ø FORI=1TO3ØØ:NEXT
 139Ø GCOLØ,Ø:MOVEX,Y:PRINT"*"
 14ØØ PROCrandom
 141Ø ENDPROC
 142Ø :
 143Ø DEF PROCrandom
 144Ø X=15Ø:Y=43Ø:DI=1
 145Ø R%=RND(4)
 146Ø IF R%=1 X=15Ø:Y=79Ø:DI=1:GOTO 149
Ø
 147Ø IF R%=2 X=1Ø5Ø:Y=79Ø:DI=Ø:GOTO 14
9Ø
 148Ø IF R%=3 X=1Ø5Ø:Y=43Ø:DI=Ø
 149Ø ENDPROC
 15ØØ :
 151Ø DEF PROCend
 152Ø GCOLØ,12:MOVE39Ø,55Ø:PRINT"GAME E
NDS"
 153Ø GCOLØ,11:MOVE39Ø,13Ø:PRINT"Score:
";score
 154Ø FOR I=1 TO 8
 155Ø SOUND1,-15,1ØØ,2:SOUND3,-15,11Ø,2
:SOUND1,-15,15Ø,2
 156Ø SOUND3,-15,13Ø,2:SOUND1,-15,5Ø,2:
SOUND3,-15,6Ø,2
 157Ø SOUND1,-15,8Ø,2:SOUND3,-15,9Ø,2:S
OUND1,-15,5Ø,2
 158Ø SOUND3,-15,7Ø,2
 159Ø NEXT I
 16ØØ TIME=Ø:REPEAT UNTIL TIME=5ØØ
 161Ø CLG
 162Ø ENDPROC
 163Ø :
 164Ø DEF PROCfour
 165Ø CLS:GCOLØ,1:H=Ø
 166Ø VDU19,1,Ø,Ø,Ø,Ø
 167Ø FOR J=61Ø TO 73Ø STEP 3Ø
 168Ø IF J<66Ø THEN 174Ø
 169Ø FOR I=1 TO 5
 17ØØ READ X,X1
 171Ø PROCbox
 172Ø PROCbox2
 173Ø NEXT I
 174Ø FOR K=1 TO 4
 175Ø READ X2
 176Ø PROCbox1
 177Ø NEXT K
 178Ø NEXT J
 179Ø IF H=Ø RESTORE:H=-36Ø:GOTO167Ø
 18ØØ GCOLØ,2:W=35Ø:Z=75Ø:X1=Ø:Y1=Ø:RES
TORE222Ø
 181Ø VDU19,2,Ø,Ø,Ø,Ø
 182Ø MOVE W,Z
 183Ø FORJ=1TO3Ø:READX,Y:DRAWX+X1,Y+Y1:
NEXT
 184Ø DRAW49Ø+X1,75Ø+Y1:DRAW49Ø+X1,76Ø+
Y1:DRAW31Ø+X1,76Ø+Y1:DRAW31Ø+X1,75Ø+Y1:
DRAW4ØØ+X1,75Ø+Y1
 185Ø MOVE465+X1,715+Y1:DRAW46Ø+X1,735+
Y1:DRAW445+X1,74Ø+Y1:DRAW43Ø+X1,736+Y1:
DRAW42Ø+X1,72Ø+Y1
 186Ø MOVE43Ø+X1,725+Y1:DRAW43Ø+X1,722+
Y1
 187Ø MOVE445+X1,717+Y1:DRAW445+X1,713+
Y1
 188Ø IF X1=Ø AND Y1=Ø PROCseta:GOTO182
Ø
 189Ø IFX1=4ØØAND Y1=Ø PROCsetb:GOTO182
Ø
 19ØØ IFX1=ØAND Y1=-36Ø PROCsetc:GOTO18
2Ø
 191Ø VDU19,1,1,Ø,Ø,Ø:VDU19,5,2,Ø,Ø,Ø
 192Ø ENDPROC
 193Ø :
 194Ø DEF PROCseta
 195Ø W=75Ø:Z=75Ø:X1=4ØØ:Y1=Ø:GCOLØ,3:V
DU19,3,Ø,Ø,Ø,Ø
 196Ø RESTORE222Ø
 197Ø ENDPROC
 198Ø :
 199Ø DEF PROCsetb
 2ØØØ W=35Ø:Z=39Ø:X1=Ø:Y1=-36Ø:GCOLØ,4:
VDU19,4,Ø,Ø,Ø,Ø
 2Ø1Ø RESTORE222Ø
 2Ø2Ø ENDPROC
 2Ø3Ø :
 2Ø4Ø DEF PROCsetc
 2Ø5Ø W=75Ø:Z=39Ø:X1=4ØØ:Y1=-36Ø:GCOLØ,
5:VDU19,5,Ø,Ø,Ø,Ø
 2Ø6Ø RESTORE222Ø
 2Ø7Ø ENDPROC
 2Ø8Ø :
 2Ø9Ø DEF PROCbox
 21ØØ MOVEX,(J+H):DRAWX+5Ø,(J+H):DRAWX+
5Ø,J+H+3Ø:DRAWX,J+H+3Ø
 211Ø ENDPROC
 212Ø :
 213Ø DEF PROCbox1
 214Ø MOVEX2,J+H:DRAWX2+5Ø,J+H:DRAWX2+5
Ø,J+H+3Ø:DRAWX2,J+H+3Ø:DRAWX2,J+H
 215Ø ENDPROC
 216Ø :
```

```
 2170 DEF PROCbox2
 2180 MOVEX1,J+H:DRAWX1-50,J+H:DRAWX1-5
0,J+H+30:DRAWX1,J+H+30
 2190 ENDPROC
 2200 DATA500,550,600,650,530,580,630,6
30
 2210 DATA50,950,100,1000,150,1050,200,
1100,250,1150,500,550,600,650,20,980,70
,1030,120,1080,170,1130,220,1180,530,58
0,630,630,50,950,100,1000,150,1050,200,
1100,250,1150,500,550,600,650
 2220 DATA 350,745,380,740,400,705,410,
675,390,635,395,595,370,590,370,585,410
,585,415,625,435,650,455,655,500,645,52
5,645,525,655,500,660,470,670,460,695,5
00,705,525,705,525,713,465,715,455,705,
435,690,425,690,420,700,420,720,405,725
 2230 DATA 395,740,400,750
 2240 :
 2250 DEF PROCstart
 2260 CLS:RESTORE
 2270 miss=0:score=0:M1=0:A$="J"
 2280 VDU23,1,0;0;0;0;
 2290 FOR I=0 TO 1:PRINTTAB(9,I+5)CHR$1
41CHR$134"THE MANHOLE GAME":NEXT I
 2300 PRINTTAB(2,10)CHR$130"Range of di
fficulty? (1,2 or 3)"
 2310 *FX15,0
 2320 REPEAT:D=GET:UNTIL INSTR("123",CH
R$D)
 2330 D=100*(51-D):IF D=0 THEN D=1
 2340 CLS:PRINTTAB(13,10)"GENTLY !!!!!!
!!"
 2350 TIME=0:REPEAT UNTIL TIME=200
 2360 ENDPROC
 2370 :
 2380 REM ERROR EXIT
 2390 ON ERROR OFF
 2400 MODE7:IF ERR<>17 THEN REPORT:PRIN
T" at line ";ERL
 2410 END
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

SOUND SUPPRESSION

Sound output can be prevented by the command *FX210,n where n is any non zero value. If n equals zero, then the sound channel is reinstated.

*SAVE EXTENSION - M.Sykes

When using *SAVE to save a file, the usual method is to type:
*SAVE <filename> <start address> <+length> <execution address>
This can be expanded to include the load address of the file when loaded back into the computer. The load address is normally taken to be the start address of the file, but it can be appended to the end of the normal set of numbers. i.e. *SAVE <filename> <start address> <length> <execution address> <reload address>.

If the load address is typed in as FFFF then the computer cannot load the file back into memory unless the *LOAD <filename> <load address> command is used, offering an easy method of software protection, but do make a list of the necessary load addresses.

DIRECT ENTRY OF BASIC TOKENS (O.S. 1.2) - L.A.Leetham, Sheila Snowden, Peter Potter

When typing in program or in immediate mode it is possible to enter some keywords with the shift or control function keys e.g. PRINT ~(Ctrl-f0). Ctrl-f0 generates the ASCII code &90 which is entered into Basic input buffer and is interpreted as a token, the token for PAGE being &90. This technique is particularly useful in saving space when defining user keys. Examination of the list of keywords and tokens (see BEEBUG reference card) will show all the keywords that can be generated in this way. Some of the more useful ones are:-

Shift f5 - ERROR	Ctrl f1 - TIME
Shift f8 - STEP	Ctrl f2 - LOMEM
Ctrl f0 - PAGE	Ctrl f3 - HIMEM

Note that some of the characters generated by these keys are also teletext control codes and that these will be generated when the program is typed in. However they are turned into keywords when the program is listed.

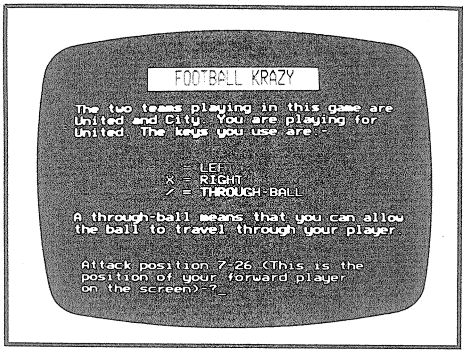
FOOTBALL KRAZY (32K)

by Alan Dickinson

At the height of the football season we present here, for your entertainment, the best game of football in town! In each game, you get 9Ø seconds of fast moving, high scoring action, where you pit your wits and skill against those of the computer.

Although this comparatively short program is written in Basic, the game is fast and furious as a result of the good structuring techniques that have been used. This also enables you to modify the program quite easily, and change, for example, the playing time and the names of the teams.

In the game presented here, you are designated as the 'United' team with the yellow chequered shirts, playing against the computer, which controls the 'City' team in the pale blue striped shirts.



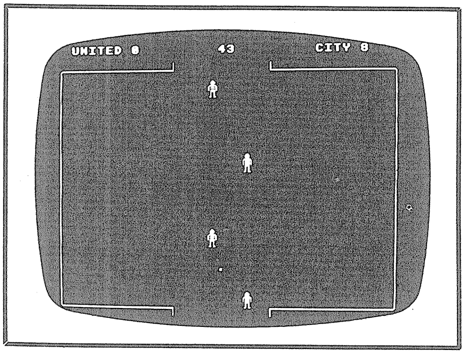
```
FOOTBALL KRAZY

The two teams playing in this game are
United and City. You are playing for
United. The keys you use are:-

          Z = LEFT
          X = RIGHT
          / = THROUGH-BALL

A through-ball means that you can allow
the ball to travel through your player.

 Attack position 7-26 (This is the
 position of your forward player
 on the screen)-?_
```



When the program runs, it initially asks you to enter the position in which you want your forward man to play (the goalie always stays on the bottom row). If you enter a value outside of the range asked for, then your man will be assigned to position 7. Position 26 is in the row next to your goalie, and 7 is in the row next to the computer's goalie. You are then asked for a value for the computer's defence tactic; this governs how the computer will respond in the game. The greater the number entered, the harder the game played. Try, for example, entering 26 in response to the position request, and 1ØØ to the defence tactic. In this arrangement, you will find it virtually impossible to beat the computer as it will dribble, tackle, pass and score with unnerving accuracy, and run rings around your players (Nobody in the office won, or even scored in this mode!). If you try entering negative numbers to 'defence tactic', you will find that you can rapidly bring your opponents players to a standstill!

To move your players, use the 'Z' key to go to the left, 'X' to move right while at any time, pressing '/' will allow an oncoming ball to pass through your player. A word of warning though, when a goal is scored, the computer moves its men back to the centre of the pitch, but leaves your men where they were when the goal was scored. You should, therefore, move your men back quickly, or the computer will instantly score another goal!

If you do wish to change the playing time, alter the value of Z% in line 4ØØ, and if you'd prefer different team names, then they should be changed in lines 62Ø, 63Ø, 1Ø7Ø, 1Ø8Ø, and 121Ø to 125Ø. (Don't forget to alter the relevant TAB values to take into account any change in length of the names you use.)

Now you can sit back and enjoy the skill of the cup final every night of the week!

```
   10 REM Program: FOOTER
   20 REM Version: B1.5
   30 REM Author A.DICKINSON
   40 REM BEEBUG March 1984
   50 REM Program subject to Copyright
   60 :
   70 ON ERROR IF ERR<>17 THEN ON ERROR
 OFF:MODE 7:REPORT:PRINT" at line ";ERL
:END
   80 REPEAT
   90 MODE 7
  100 PROCinit
  110 PROCtitle
  120 MODE 1
  130 VDU23,1,0;0;0;0;0
  140 PROCstart
  150 PROCtune
  160 PROCplay
  170 PROCfinalscore
  180 REPEAT
  190 A$=GET$
  200 UNTIL A$="Y" OR A$="N"
  210 UNTIL A$="N"
  220 *FX15,1
  230 MODE 7
  240 END
  250 :
  260 DEF PROCinit
  270 VDU23,224,0,0,0,24,24,0,0,0
  280 VDU23,225,0,24,60,60,24,126,173,1
81
  290 VDU23,226,173,181,60,36,36,36,36,
102
  300 VDU23,227,0,24,60,60,24,126,129,1
89
  310 VDU23,228,129,189,60,36,36,36,36,
102
  320 VDU23,229,0,0,0,0,0,0,0,255
  330 VDU23,230,255,0,0,0,0,0,0,0
  340 VDU23,231,1,1,1,1,1,1,1,1
  350 VDU23,232,128,128,128,128,128,128
,128,128
  360 VDU23,233,255,1,1,1,1,1,1,1
  370 VDU23,234,255,128,128,128,128,128
,128,128
  380 VDU23,235,1,1,1,1,1,1,1,255
  390 VDU23,236,128,128,128,128,128,128
,128,255
  400 Z%=9000:REM GAME LENGTH
  410 B$=CHR$17+CHR$7+CHR$224
  420 P$=CHR$32+CHR$226+CHR$32+CHR$11+C
HR$8+CHR$8+CHR$8+CHR$32+CHR$225+CHR$32
  430 Q$=CHR$32+CHR$228+CHR$32+CHR$11+C
HR$8+CHR$8+CHR$8+CHR$32+CHR$227+CHR$32
  440 ENDPROC
  450 :
  460 DEF PROCtitle
  470 CLS:FORP=2TO3:PRINTTAB(8,P)CHR$13
1;CHR$157;CHR$141;CHR$129"FOOTBALL KRAZ
Y   ";CHR$156:NEXT
  475 PRINTTAB(0,5)"The two teams playi
ng in this game are  United and City. Y
ou are playing for    United. The keys
you use are:-"
  480 PRINTTAB(10,10)CHR$129"Z = LEFT";
  490 PRINTTAB(10,11)CHR$130"X = RIGHT"
;
  500 PRINTTAB(10,12)CHR$131"/ = THROUG
H-BALL";
  501 PRINTTAB(0,14)"A through-ball mea
ns that you can allow the ball to trave
l through your player."
  510 PRINTTAB(0,18)CHR$133"Attack posi
tion 7-26 (This is the";TAB(0,19)CHR$13
3;"position of your forward player"TAB(
0,20)CHR$133;"on the screen)-";
  520 INPUT A%
  530 IF A%<7 OR A%>26 A%=13
  540 PRINTTAB(0,22)CHR$134"Defence tac
tic ";
  550 INPUT T%
  560 ENDPROC
  570 :
  580 DEF PROCstart
  590 VDU19,0,4,0,0,0:VDU19,1,6,0,0,0
  600 S$=STRING$(3,CHR$32)
  610 CLS:SP%=0:SQ%=0
  620 PRINTTAB(1,1)"UNITED 0";
  630 PRINTTAB(28,1)"CITY   0";
  640 PRINTTAB(1,3)STRING$(38,CHR$229);
  650 PRINTTAB(1,30)STRING$(38,CHR$230)
;
  660 PRINTTAB(14,3)SPC11
  670 PRINTTAB(14,30)SPC11
  680 PRINTTAB(13,3)CHR$235;TAB(25,3)CH
R$236;TAB(13,30)CHR$233;TAB(25,30)CHR$2
34
  690 FOR Y%=4 TO 29
  700 PRINTTAB(0,Y%)CHR$231
  710 PRINTTAB(39,Y%)CHR$232
  720 NEXT
  730 P%=18
  740 COLOUR2
  750 PRINTTAB(P%,A%)P$;TAB(P%,28)P$;
  760 Q%=18:B%=18+RND(4)
  770 COLOUR1
  780 PRINTTAB(Q%,5)Q$;TAB(Q%,B%)Q$;
  790 ENDPROC
  800 :
  810 DEF PROCtune
  820 RESTORE
  830 REPEAT
  840 READ V%,D%:SOUND 1,-15*V%,72,D%
  850 UNTIL D%=0
  860 DATA 1,3,0,3,1,3,0,3
  870 DATA 1,2,0,1,1,2,0,1,1,4,0,2
  880 DATA 1,2,0,1,1,2,0,1,1,2,0,1,1,4,
0,2
  890 DATA 1,2,0,1,1,4,0,0
  900 ENDPROC
```

»

```
  910 :
  920 DEF PROCplay
  930 TIME=0
  940 REPEAT
  950 X%=20:D%=1-RND(2):Y%=16
  960 IF RND(10)>5 E%=1 ELSE E%=-1
  970 PRINTTAB(Q%,5)S$;TAB(Q%,B%)S$;TAB
(Q%,4)S$;TAB(Q%,B%-1)S$;
  980 Q%=18:PRINTTAB(X%,Y%)B$;
  990 PROCutd:PROCcity
 1000 REPEAT
 1010 PRINTTAB(19,1);TIME DIV 100;
 1020 PROCutd:PROCcity:PROCball
 1030 UNTIL Y%<2 OR Y%>30 OR TIME>Z%+99
 1040 IF Y%<2 SP%=SP%+1
 1050 IF Y%>30 SQ%=SQ%+1
 1060 PRINTTAB(X%,Y%)" ";
 1070 PRINTTAB(1,1)"UNITED ";SP%
 1080 PRINTTAB(28,1)"CITY   ";SQ%;
 1090 FOR I%=0 TO 100 STEP 7
 1100 IF Y%<2 SOUND 1,-15,I%,2 ELSE SOU
ND 1,-15,110-I%,2
 1110 NEXT
 1120 UNTIL TIME>Z%+99
 1130 ENDPROC
 1140 :
 1150 DEF PROCfinalscore
 1160 PRINTTAB(15,16)" FULL TIME ";
 1170 PROCtune
 1180 REPEAT UNTIL TIME>Z%+500
 1190 CLS
 1200 PRINTTAB(2,4)"Result:"
 1210 PRINTTAB(4,6)"United ";SP%;
 1220 PRINTTAB(4,8)"City   ";SQ%;
 1230 IF SP%>SQ% PRINTTAB(4,11)"UNITED
WIN THE MATCH";
 1240 IF SP%=SQ% PRINTTAB(4,11)"IT'S A
DRAW ! ! !";
 1250 IF SP%<SQ% PRINTTAB(4,11)"CITY WI
N THE MATCH !!!";
 1260 PRINTTAB(2,14)"Again ?";
 1270 *FX15,1
 1280 ENDPROC
 1290 :
 1300 DEF PROCutd
 1310 COLOUR2
 1320 IF INKEY(-98) AND P%>1 P%=P%-1:PR
INTTAB(P%,A%)P$;SPC1;TAB(P%,28)P$;SPC1;
:ENDPROC
 1330 IF INKEY(-67) AND P%<36 PRINTTAB(
P%,A%)SPC1;P$;TAB(P%,28)SPC1;P$;:P%=P%+
1:ENDPROC
 1340 PRINTTAB(P%,A%)P$;TAB(P%,28)P$;
 1350 ENDPROC
 1360 :
 1370 DEF PROCcity
 1380 COLOUR1
 1390 IF X%<Q%+1 AND Q%>3 AND Y%<13+T%
Q%=Q%-1:PRINTTAB(Q%,5)Q$;SPC1;TAB(Q%,B%
)Q$;SPC1;:ENDPROC
 1400 IF X%>Q%+1 AND Q%<34 AND Y%<13+T%
PRINTTAB(Q%,5)SPC1;Q$;TAB(Q%,B%)SPC1;Q
$;:Q%=Q%+1:ENDPROC
 1410 PRINTTAB(Q%,5)Q$;TAB(Q%,B%)Q$;
 1420 ENDPROC
 1430 :
 1440 DEF PROCball
 1450 N%=X%+D%
 1460 M%=Y%+E%
 1470 IF N%<1 OR N%>38 N%=X%:D%=-D%:SOU
ND 1,-15,100,1
 1480 IF M%=30 OR M%=3 IF N%<14 OR N%>2
3 M%=Y%:E%=-E%:SOUND 1,-15,120,1
 1490 IF M%=5 OR M%=B% PROCQman
 1500 IF M%=A% OR M%=28 PROCPman
 1510 PRINTTAB(X%,Y%)SPC1;TAB(N%,M%)B$;
 1520 X%=N%:Y%=M%
 1530 ENDPROC
 1540 :
 1550 DEF PROCPman
 1560 IF N%<P% OR N%>P%+2 ENDPROC
 1570 SOUND 1,-15,60,1
 1580 IF INKEY(-67) D%=-1
 1590 IF INKEY(-98) D%=1
 1600 IF NOT INKEY(-105) E%=-E%
 1610 ENDPROC
 1620 :
 1630 DEF PROCQman
 1640 IF N%<Q% OR N%>Q%+2 ENDPROC
 1650 SOUND 1,-15,52,1
 1660 IF RND(10+T%)>3 E%=1 ELSE E%=-E%
 1670 D%=0
 1680 R%=RND(10)
 1690 IF R%>7 D%=-1
 1700 IF R%<4 D%=1
 1710 IF T%<3 OR R%<4 ENDPROC
 1720 IF X%>25 D%=-1
 1730 IF X%<15 D%=1
 1740 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

DISABLING THE ESCAPE KEY & DESTROYING PROGRAMS

```
*FX 200,0 Re-enables the escape key.
*FX 200,1 Will disable the escape key.
*FX 200,2 enables the escape key, but pressing Break will clear the memory,
          preventing the program being retrieved with OLD.
*FX 200,3 disables the escape key, and clear the memory on Break.
```

IF YOU WRITE TO US

BACK ISSUES (Members only)

All back issues are kept in print (from April 1982). Send 90p per issue PLUS an A5 SAE to the subscriptions address. This offer is 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 & Software Address

BEEBUG
PO BOX 109
High Wycombe
Bucks

SUBSCRIPTIONS

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

MEMBERSHIP COSTS:
U.K.
£5.40 for 6 months (5 issues)
£9.90 for 1 year (10 issues)

Eire and Europe
Membership £16 for 1 year.
Middle East £19
Americas and Africa £21
Elsewhere £23
Payments in Sterling preferred.

PROGRAMS AND ARTICLES

All programs and articles used are paid for at around £25 per page, but please give us warning of anything substantial that you intend to write. 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", "Minitext Editor" or other means. 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.

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

Editorial Address

BEEBUG
PO Box 50
St Albans
Herts

BEEBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Editor: Mike Williams.
Technical Editor: Philip Le Grand. Production Editor: Phyllida Vanstone.
Technical Assistants: Alan Webster and David Fell.
Managing Editor: Lee Calcraft.
Thanks are due to Sheridan Williams, Adrian Calcraft, John Yale, Robert Barnes, Michael Beasley, Colin Lindsay, Hugh Brown-Smith and Tim Powys-Lybbe for assistance with this issue.

BEEBUG NEW ROM OFFER

1.2 OPERATING SYSTEM

A special arrangement has been agreed between Acorn and BEEBUG whereby BEEBUG members may obtain the 1.2 operating system in ROM at the price of £5.85 including VAT and post and packing.
The ROM will be supplied with fitting instructions to enable members to install it in their machine.
If the computer does not subsequently operate correctly, members may take their machine to an Acorn dealer for the upgrade to be tested, which will be done at a charge of £6.00 plus VAT. This charge will be waived if the ROM is found to have been defective. If the computer has been damaged during the installation process, the dealer will make a repair charge.

FREE

NEW ROMS FOR OLD
EXCHANGE YOUR 1.0 FOR THE 1.2

We can now exchange your old 1.0 operating system for the new 1.2, free of charge. To take advantage of this offer, please send your 1.0 (supplied on eprom with a carrier board), in good condition to the address below.

£5 FOR YOUR OLD 1.0

If you have the 1.0 operating system and have already bought a 1.2, we will exchange the 1.0 (supplied on eprom with a carrier board) for a £5 voucher. This voucher may be used against any purchase from BEEBUGSOFT.

ADDRESS FOR 1.2 OS:-
ROM Offer, BEEBUG, PO Box 109, High Wycombe, Bucks, HP11 2TD.

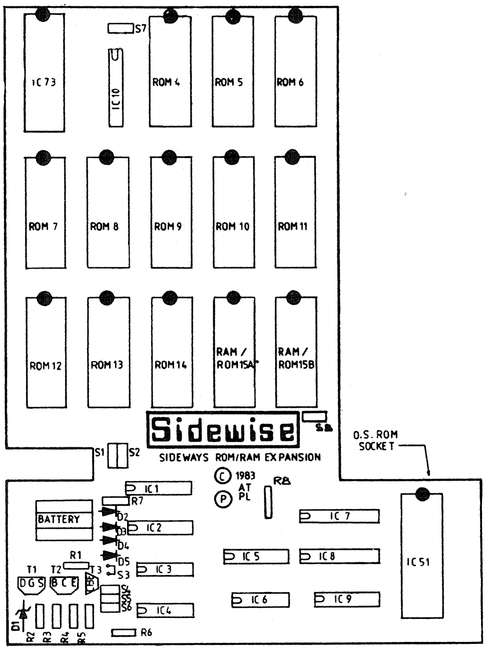
BEEBUGSOFT

ATPL'S SIDEWAYS ROM EXPANSION BOARD

SPECIAL PRICE TO MEMBERS £39.00 inc.

Save £5.70 on normal price of £44.70

* Simply plugs into the BBC Micro
* No soldering necessary
* Increases the sideways ROM capacity to 16
* Fully buffered - allows all sockets to be used
* Complete with full and detailed instruction booklet.
* Accepts 16K RAM in special sockets
* Battery back up facility for RAM
* As used at BEEBUG
* Reviewed in BEEBUG vol.2 number 6



HOW TO ORDER

Please send your order with a cheque / postal order made payable to BEEBUG, and enclose your membership number. We are unable to supply the board to overseas members.

The address for SIDEWAYS is:
BEEBUGSOFT, PO Box 1Ø9, High Wycombe, Bucks. HP11 2TD.

MAGAZINE CASSETTE OFFER

To save wear and tear on fingers and brain, we offer, each month, a cassette of the programs featured in the latest edition of BEEBUG. The first program on each tape is a menu program, detailing the tape's contents, and allowing the selection of individual programs. The tapes are produced to a high technical standard by the process used for the BEEBUGSOFT range of titles. Ordering information, and details of currently available cassettes are given below.

All previous magazine cassettes (from Vol.1 No.10) are available.

This month's cassette (Vol.2 No.9) includes: the Manhole game, Machine Code Graphics example programs, ASTAAD2, the full new version, Stonemason, Football game, Multiple Disc Catalogues (4 programs in all), program to print function key labels,

compact function key definer, programs to test your sideways ROMs, program version of Bach's Cantata No.147, plus the winning program from the Brainteaser Roman Numeral competition.

All magazine cassettes cost £3.00 each. For ordering information see BEEBUGSOFT advertisement at the back of this month's magazine supplement.

MAGAZINE CASSETTE SUBSCRIPTION

We are able to offer members subscription to our magazine cassettes. Subscriptions will be for a period of one year and are for ten consecutive issues of the cassette. If required, subsriptions may be backdated as far as Vol.1 No.10, which was the first issue available on cassette. This offer is available to members only, so when applying for subscription please write to the address below, quoting your membership number and the issue from which you would like your subscription to start.

CASSETTE SUBSCRIPTION ADDRESS:

Please send a sterling cheque with order, together with your membership number and the date from which the subscription is to run, to:
PO Box 109, High Wycombe, Bucks,

CASSETTE SUBSCRIPTION PRICE:
UK £33 inc VAT and p&p
OVERSEAS (inc Eire) £39 inc p&p
(no VAT payable).

BEEBUG BINDER OFFER

BEEBUG MAGAZINE BINDER OFFER

A hard-backed binder for BEEBUG magazine is available. These binders are dark blue in colour with 'BEEBUG' in gold lettering on the spine. They allow you to store the whole of one volume of the magazine as a single reference book. Individual issues may be easily added or removed, thus providing ideal storage for the current volume as well.

BINDER PRICE
U.K. £3.90 inc p&p, and VAT.
Europe £4.90 inc p&p
(no VAT payable).
Elsewhere £5.90 inc p&p
(no VAT payable).

Make cheques payable to BEEBUG.
Send to Binder Offer, BEEBUG, PO Box 109, High Wycombe, Bucks.
Please allow 28 days for delivery on U.K. orders.

Printed in England by Staples Printers St Albans Limited at The Priory Press. ISSN 0263-7561