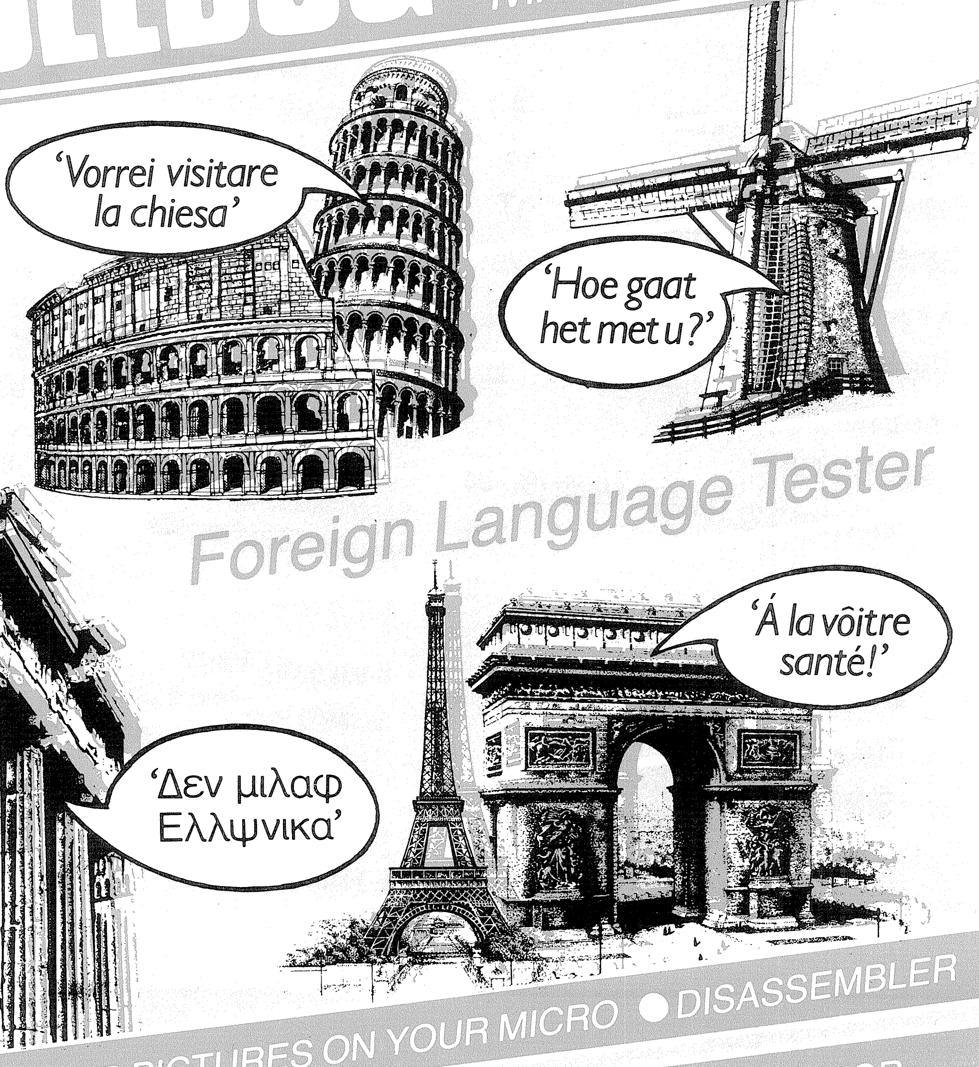
Vol.8 No.2 June 1989

BEEBUG

FOR THE BBC MICRO & MASTER SERIES



Foreign Language Tester

● WEATHER PICTURES ON YOUR MICRO ● DISASSEMBLER

● ACORN'S NEW COMPUTER ● BASIC LINE EDITOR

BEEBUG

Vol.8 No.2 June 1989

FEATURES

Foreign Language Tester 9
Basic Line Editor 13
Genealogy Without Tears 17
A Fast Backup Utility for DFS 20
File Handling for All (Part 12) 22
An Extended Disassembler 28
First Course -
Investigating Teletext Mode (3) 33
512 Forum 36
A Clock Face for the Master 39
Workshop - Spin a Disc (Part 3) 41
The Comms Spot 44
Share Investor (Part 2) 46
The Game of Cribbage 49

REVIEWS

The New BBC Microcomputer 6
Weather Pictures on your Micro 26

REGULAR ITEMS

Editor's Jottings 4
News 5
RISC User 55
Bulletin Boards 56
Postbag 57
Hints and Tips 59
Personal Ads 60
Subscriptions & Back Issues 62
Magazine Disc/Cassette 63

HINTS & TIPS

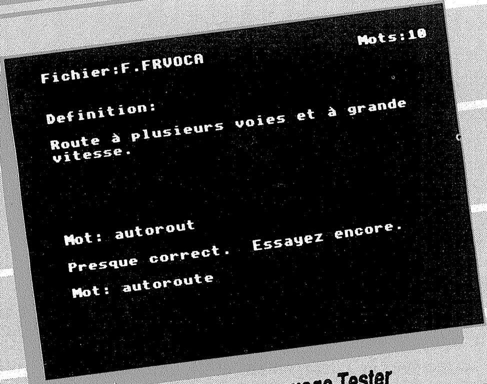
Fonts on the Master
Loading Data from a Library Directory
Using Spellmaster to Handle Sideways RAM
Tape-to-Disc on a Master
Finding the Screen Mode

PROGRAM INFORMATION

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

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

The New Acorn A3000

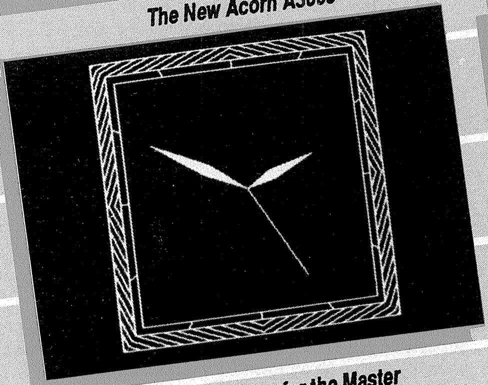


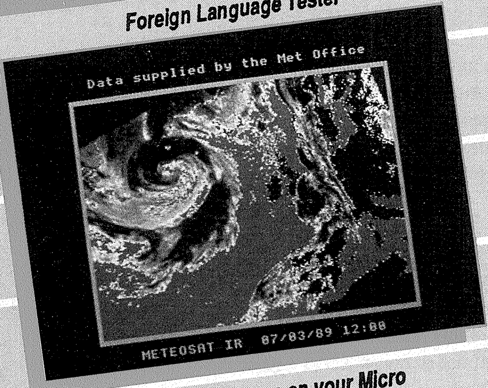
```
Fichier:F.FRVOCA                              Mots:10

Definition:
Route à plusieurs voies et à grande
vitesse.


Mot: autorout
Presque correct.  Essayez encore.
Mot: autoroute
```

Foreign Language Tester

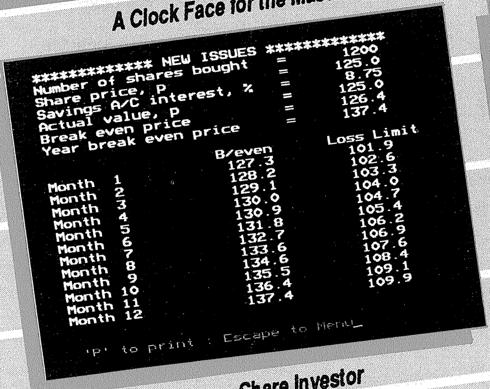

A Clock Face for the Master



```
Data supplied by the Met Office

METEOSAT IR  07/03/89 12:00
```

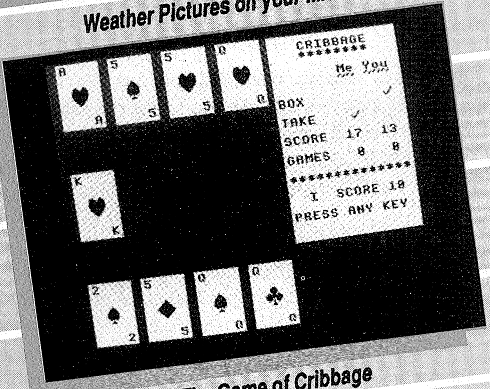
Weather Pictures on your Micro



```
************* NEW ISSUES *************
Number of shares bought      =      1200
Share price, p               =     125.0
Savings A/C interest, %      =      8.75
Actual value, p              =     125.0
Break even price             =     126.4
Year break even price        =     137.4
                        B/even      Loss Limit
Month  1                 127.3        101.9
Month  2                 128.2        102.6
Month  3                 129.1        103.3
Month  4                 130.0        104.0
Month  5                 130.9        104.7
Month  6                 131.8        105.4
Month  7                 132.7        106.2
Month  8                 133.6        106.9
Month  9                 134.6        107.6
Month 10                 135.5        108.4
Month 11                 136.4        109.1
Month 12                 137.4        109.9

   'p' to print : Escape to Menu_
```

Share Investor



```
                          CRIBBAGE
                          ********
                           Me  You
                                ✓
                       BOX
                       TAKE   ✓
                       SCORE  17   13
                       GAMES   0    0
                       **************
                         I   SCORE 10
                       PRESS ANY KEY
```

The Game of Cribbage

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

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

Program will not function on a cassette-based system.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings

ACORN'S NEW COMPUTER

We are pleased to present in this issue a full review of Acorn's latest model in the Archimedes range, the A3000, as we anticipated last month. This fully compatible machine also becomes the latest Acorn computer to receive the endorsement of the BBC, and thus becomes the latest BBC micro.

The new A3000, claimed curiously by Acorn to be part of the Archimedes range, but not an Archimedes, represents a major step by Acorn to address the needs of the home market, and face the competition of Atari and Commodore in particular.

The A3000 can only build on the reputation already established by the Archimedes as one of the world's fastest and most powerful micros, and is likely to prove popular both with the public at large, and with BBC micro and Master users who want to upgrade to the Archimedes range. Acorn rightly deserve our applause.

BBC ACORN USER SHOW

As we have been reporting, BBC micro and Archimedes users can look forward this year to a major show totally dedicated to their machines. We shall be there with a joint BEEBUG/RISC User stand (stand 55), and we hope as many BEEBUG members as possible will visit us there.

BEEBUG has also not been idle in recent months, and we shall be demonstrating for the first time our new word processor/desktop publishing package for the Archimedes, and a compact hand-held scanner, also for the Archimedes, which is due to retail at under £200.

Acorn's new A3000 will also have its first public airing at the show, which promises to be one not to miss for users of Acorn computers. So make sure of the dates and location now - 21st to 23rd July 1989 at Alexandra Palace.

This month's telesoftware password is *dandelion*.

LOOKING AHEAD WITH BEEBUG

The following features are likely to be included in the July issue of BEEBUG:

Applications & Utilities:

FOUR CHANNEL TEMPERATURE PROBE
MATHEMATICAL TRANSFORMATIONS
ADFS DISC SECTOR EDITOR
ABBREVIATION SUBSTITUTION
FOREIGN LANGUAGE TESTER (PART 2)
FONT DESIGNER
MAGIC LANTERN SLIDES

Reviews:

MUSIC 5000 JUNIOR
STRETCH BY 4MATION

Plus News, Postbag, Hints and more.

RISC User

RISC User is the largest circulation magazine devoted entirely to the Acorn Archimedes range. It is available on subscription to all BEEBUG members at a substantially reduced rate (see page 66).

We expect the July/August issue to include:

MULTI-TASK FORMATTER
CROSS REFERENCER
BASIC LOAD & SPOOL
PROGRAM COMPRESSOR
BBC SCREEN CONVERTER
USING 1ST WORD PLUS
PROCEDURE LIBRARY

Reviews:

ARCCOMM FROM BBCSOFT
BASIC COMPILERS
MULTIPLE I/O PODULE FROM BRAINSOFT
ANCESTRY FROM MINERVA

and more.

RISC User is the ideal magazine to keep up to date with the Archimedes scene in every respect, and is particularly useful if you are contemplating the purchase of an Archimedes in the near future.

*The latest details of the contents and distribution of both magazines are contained in the BEEBUG area of Micronet. Just type *BEEBUG# when on-line.*

News News News News News News

ARM 3 IS THE FUTURE

Acorn's research and development department are currently evaluating the ARM 3 processor, the latest development in the RISC based chipset that forms the heart of the Archimedes. The ARM 3 chip uses a 24MHz clock speed which is three times that of the ARM 2 in the Archimedes. Additionally, there is an on-chip 4K instruction cache which is used to hold the last 1000 or so instructions executed. This allows for very fast execution of program loops, and also ensures that the faster chip can be used with existing RAM chips.

Already, one company, Cambridge based Aleph One Ltd., is working on an upgrade board which will allow the ARM 3 chip to be fitted to existing Archimedes machines. The company says that this will offer a speed increase of between three and five times without any additional modifications to the hardware or software. The unit will sell for around £595, but will not be released until quantity supplies of the ARM 3 are available. For more details, contact Aleph One Ltd., The Old Courthouse, Bottisham, Cambridge CB5 9BA, tel. (0223) 811679.

A LONG STRETCH

Stretch from 4-Mation is a new program which allows the Beeb to produce realistic posters, titles, signs, headings and banners. The package is supplied with a set of sixteen different fonts and two sets of pictures, and these can be mixed at will to make up the poster. The image is represented on the screen as it is composed, and text and pictures can be positioned using the keyboard, mouse or a tracker ball. The fonts are printed as outline designs, which means that the border may be of a different colour to the character's interior. *Stretch* will work with a variety of printers including colour devices such as the Integrex 132 and Star LC10.

Stretch is supplied on two discs along with a comprehensive full-colour manual, a key strip, and a planning sheet, all in a smart looking presentation wallet. The package costs £26.45, and is available from 4-Mation Educational Resources, Linden Lea, Rock Park, Barnstaple, Devon EX32 9AQ, tel. (0271) 45566.

BARGAIN GRAPHICS FROM NIDD VALLEY

Nidd Valley Micro Products are offering their *Illustrator* and *Colourbox* packages together with a mouse for £59.90 (inc. VAT). *Illustrator* is a drawing and painting package offering nine fonts, sixty-four shading patterns and sixteen brush styles. *Colourbox* takes the images from *Illustrator* and allows colour to be added to them. Shading is used to obtain twenty-two colours, and printer dumps are provided for Epson compatibles, the Integrex 132, and the Star LC10 colour. More details from Nidd Valley Micro Products Ltd., Hammerain House, Hookstome Avenue, Harrogate, North Yorkshire HG2 8ER, tel. (0423) 870145.

HUNTING FOR TEXT

Themewise is a package designed to simplify the searching of text files for particular words or phrases. The package was developed by the Sociology department of Nottingham University, and was designed to simplify the analysis of questionnaires and surveys. *Themewise* will work with text files in any format (including View and Wordwise), and could be used to form the basis of a powerful bibliography system. As well as the text searching facilities, *Themewise* also includes a file management utility and a section compiler, this being used to create a new file consisting of specified fields picked out of other files. Documentation is supplied in the form of text files on disc. *Themewise* is suitable for all BBC computers, and is available for £7 (inclusive) from Dr. D Lawrence, Department of Sociology, University of Nottingham, Nottingham NG7 2RD.

PRESTEL GROWS

Micronet users who want a break from reading about computers can take advantage of four new on-screen magazines launched by Dialcom - the owners of Prestel. The magazines are *Newsday* containing the latest news and in-depth stories, *Look!* for leisure and the arts, *SportsEye* for competitors and spectators alike, and *Games City* which provides a wealth of quizzes as well as a gateway into the multi-user game Shades. To browse the new magazines simply log on and go directly to page 1.

The New BBC Microcomputer

Mike Williams reviews Acorn's recently launched A3000, which takes its place as the new BBC microcomputer.

Over the past few months, rumours have persistently appeared regarding a new Archimedes from Acorn. As we predicted in the last issue of BEEBUG, the new machine has now been publicly announced. Called the A3000, the new computer is said by Acorn to be part of the Archimedes range, but is not an Archimedes! In addition, the new machine has the endorsement of the BBC, and now becomes the new BBC Microcomputer.

INTRODUCTION

The A3000 marks a radical change from the Archimedes as we have come to know it, reverting to the single box approach of the BBC micro and Master 128, and used so successfully by the Atari ST and others. The moulded plastic case is in the standard Acorn crackle-finish cream. The dimensions of 48.5cm by 32cm by 6.5cm make it about 1cm wider than the Master 128, but the depth from front to back, and the height are less. The new machine has the same 'PC-enhanced' style of keyboard as other Archimedes (built into the front sloping fascia), but it is of different construction, and has a much lighter and more positive feel to it.

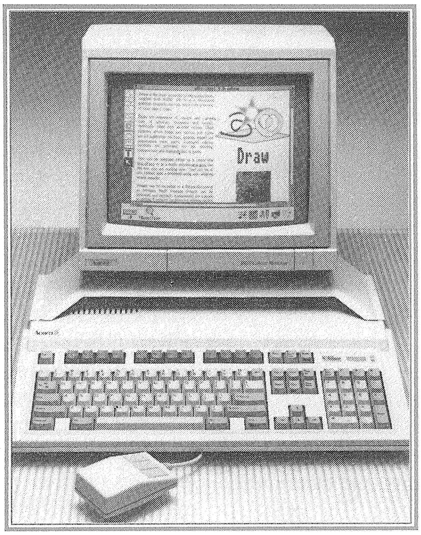


Figure 1. The new Acorn A3000 computer

A recessed on/off switch is fitted at the left-hand side, while a standard 3.5" disc drive is built into the right-hand side of the new machine, to the rear of the keyboard. The Reset switch is located beneath this and is also recessed.

The disc drive feels comfortable to access for inserting and removing discs, though the eject button requires a little more effort than I might have wished, and my fingers were often slightly blocked by the edge of the casing. However, I found no confusion in use between the eject button and the Reset switch.

In use, the disc drive is noisier compared to the disc drives fitted to the Archimedes 300 series, but this may be the consequence of the use of a plastic rather than a metal case. One useful touch is the provision of a repeat 'telltale' light on the front panel to indicate when the drive is in operation.

All the connectors (bar one) are fitted at the rear of the machine and include analogue RGB, mono composite video, Econet, printer port, serial port, stereo sound jack and a podule expansion socket (see figure 2). However, the serial and Econet interfaces are both upgrades to the basic machine. An Archimedes mouse is included with the system, and this plugs into a socket recessed into the underside of the machine. This is clearly difficult to access, but nevertheless has the advantage that the mouse can readily be positioned to either the left or right of the keyboard (or wherever it is most convenient in use).

The A3000 is designed for use with either an analogue RGB monitor, or one which will accept a mono composite video input. A stand, as in the illustration, will be available to raise any monitor above the casing, and help ventilation. Certainly the power supply does generate some heat (as expected), but on the review machine this was nothing like early models of the old BBC micro which was nearly hot enough to fry an egg, and certainly fried a few components in its time.

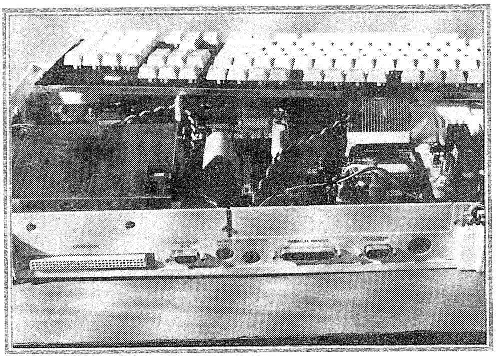


Figure 2. Rear of machine showing sockets as fitted

HARDWARE

Lifting the lid is accomplished by loosening just two screws, and the lid immediately unclips. This reveals the power supply, circuit board and disc drive (see figure 3). The power supply is quite unprotected, leaving dangerous voltages exposed, which means (officially at least) that any upgrades are dealer only. The A3000 uses the same ARM chip-set and disc controller as the latest 400 series (themselves about 10% faster than the original Archimedes 305 and 310), and the performance of the new machine is thus just as good. There is also a new custom keyboard controller chip fitted. The A3000 is also fitted with the same four RISC OS ROMs as other Archimedes, which should remove any doubts about software compatibility.

General circuit layout is very similar to existing Archimedes models, but takes advantage of 'surface mounted' components to save space. These are minute, and look little more than mere blobs of solder. The 1MByte of memory is in the form of eight vertically mounted chips (also used in the latest 400/1 series), which thus use less space on the circuit board than the more traditional horizontal ones. A row of pins is waiting for the plug-in 1MByte upgrade that will double the memory to 2MBytes, the maximum that the machine is designed to accommodate.

The board also provides for internal expansion. Econet uses the existing plug-in board, while the serial interface requires just two chips to be fitted. Acorn claims that it is using a completely bug-free version of the controller chip to avoid the problems encountered in earlier Archimedes with the serial interface.

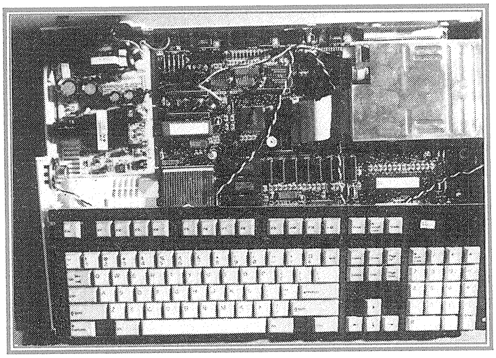


Figure 3. Plan view of the inside of the A3000

There is also a new-style mini-podule expansion capability. The only suitable podule currently announced is a combined User Port/MIDI interface from Acorn, though other manufacturers will no doubt step in. Only one such podule may be fitted anyway, and none of the existing podules will fit. However, there is also an external expansion socket fitted at the rear of the machine, and this will accept single card podules, including a hard disc interface if required. A podule expansion box offering up to four podule connections can also be mounted here. Provision has been included in the moulding for add-ons to be securely attached to the A3000, but this will obviously require a certain amount of re-designing on the part of manufacturers.

The A3000 System

Hardware
- Single box design
- 1MByte RAM (expandable to 2MBytes)
- Internal 3.5" 800K floppy disc
- Internal stereo speakers
- NiCad battery back-up
- Mouse
- Side-mounted on/off switch
- Integral power supply

Connectors
- Centronics compatible printer
- Analogue RGB
- Mono composite video
- Stereo sound
- Econet (option)
- Expansion connector (for podules)
- User port/MIDI (internal expansion)
- Serial (internal expansion)
- User port for Concept Keyboard

Price
- £649 ex. VAT (£746.35 inc. VAT)
- Monitor extra.

The new machine also sports stereo loudspeakers (using two of the same miniature speakers as previously fitted). This is hardly hi-fi sound quality, but the stereo effect is noticeable. For example, using *STEREO 1 -127 caused the bleep to sound only through the left-hand speaker, while *STEREO 1 127 changed it to appear from the right-hand speaker. I also experimented with some of the tunes supplied with the Maestro music package on one of the two Applications discs to achieve some interesting results. For anything approaching good quality stereo sound you will still need to use the output from the rear-mounted miniature stereo jack plug.

The A3000 is also fitted with NiCad battery back-up, the batteries being constantly recharged while the machine is switched on. This seems an ideal arrangement.

SOFTWARE

As already stated, the A3000 is supplied with the same RISC OS chips as the Archimedes, and thus provides the same multi-tasking WIMP environment. The machine thus includes Archimedes BBC Basic, while standard RISC OS applications such as Edit, Paint and Draw are supplied on two Applications discs, identical (as far as I can judge) to those supplied with RISC OS on other Archimedes, apart from the disc labels.

Two 6502 emulators are included (6502Host and 65Tube), and the PC emulator will run just as well on the A3000 as other Archimedes, though it is not supplied with the system.

Although the existing ROM set accounts for 0.5MBytes of memory, the design of the A3000 permits ROM software up to a massive 2MBytes as with other Archimedes.

The new machine officially takes over the title of 'BBC microcomputer', the revamped 400 series machines on which we reported last month being Acorn-badged only.

MANUALS

The A3000 is supplied with a Welcome Guide and a 470 page User Guide. These are quite similar to those currently being supplied with the Archimedes 400 series, but have clearly been rewritten specifically for the A3000. One point which I am sure will upset many potential purchasers is that for the first time with its machines, Acorn has provided no information at all on BBC Basic. Indeed, not since the days of the original BBC micro has Acorn supplied a complete guide to BBC Basic with its machines. A3000 purchasers who want to do any programming, and I would have thought that would have been most of them, will need to buy the Acorn BBC Basic Guide at a cost of £19.95.

Apart from that, the A3000 manuals are well produced and attractive to look at. However, I still find the larger page size (20cm by 21cm) a little unwieldy for such a large manual as the User Guide. A ring binder with hard covers would have been nice, but no doubt too expensive for Acorn to contemplate.

COMMENT

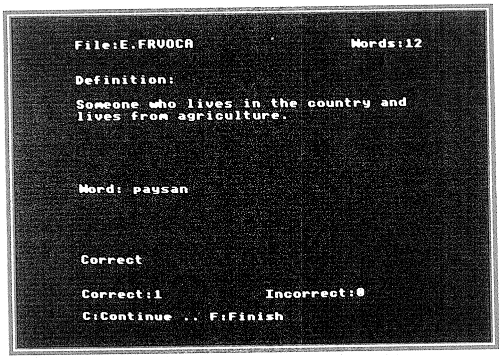
It is good to see Acorn capitalising on the power and flexibility of the Archimedes by packaging it up into an Atari style machine. It is

Continued on page 40

Foreign Language Tester

Thinking of going on a foreign holiday?
Eddy Hunt has just the program to brush up your foreign languages before you go.

In BEEBUG Vol.7 No.4 I described how both screen displays and printouts could be customised for work with foreign languages by defining appropriate foreign character sets. In this article, and a further one next month, I propose to show how this facility can be combined with View and a couple of programs to create a simple system for testing your vocabulary in a foreign language of your choice.



```
File:E.FRVOCA                       Words:12

Definition:

Someone who lives in the country and
lives from agriculture.



Word: paysan



Correct

Correct:1              Incorrect:0
C:Continue .. F:Finish
```

Testing French vocabulary

When learning a foreign language, it is important to build up a vocabulary by maintaining a list of words as they are learnt. This list should be revised at regular intervals. For any one person, the list of words will be a personal one, reflecting both the level and interests of that individual. It is not useful to spend time revising words which have already been learned, and any schedule of revision should concentrate on new words and those words which have caused difficulty, whilst allowing less frequent revision of words which have already been acquired.

Using a computer can help in a number of ways. The presentation can be randomized, and biased to more frequent presentation of 'problem' words, a record of progress can be maintained and spelling is checked automatically, although it is useful to be prompted where the spelling is not quite right and offered a second chance. I have incorporated these features into the programs VocTest (listed here) and VocProg (to be published in the following issue). The system permits words and definitions to be entered using the View word processor (ASCII files generated with other word processors should work too), employing where necessary a foreign character set as described previously.

The actual use of the programs will depend very much on individual requirements. People learning a language for holiday purposes will probably want definitions and screen instructions to be in English, while more advanced users would benefit from both instructions and definitions appearing in the target language. It may also be useful to students of *English as a Foreign Language*, where, of course, the program for re-defining the keyboard can be omitted. In the discussion and examples which follow I will assume that you are learning French through the medium of English.

SETTING UP THE SYSTEM

You will need to set up two text files as shown in figures 1 and 2. The first file should be called E.INSTRUC, and contains the on-screen instructions in English. The second file contains the vocabulary to be tested and should be given a name like E.FRVOCA. This name must end with 'A', and both files must be placed in a directory other than the default directory (directory 'E' for English in our example).

The name of the instruction file is fixed. The vocabulary file can be given any name, apart from the last letter, which should be the upper case letter 'A'. The directory letter can be used to indicate the language used. Thus, using this system, a file called E.FRVOCA as above would contain French words with explanations in

English. A file called F.FRVOCA would contain definitions in French. ADFS users will, of course, need to create appropriate directories in advance.

The instructions file should be created using View (or other word processor) so that it looks as in Figure 1. To enter the vocabulary file, the screen character definitions should be first altered to the French character set, by typing:

```
CH."FRENCH"
```

where FRENCH is the program from Vol.7 No.4 with the French character definitions included (repeated on this month's magazine disc/tape).

```
File
Definition
Word
Words
Correct
Incorrect
Continue
Finish
Almost correct. Try again.
```

Figure 1. Contents of file E.INSTRUC

The vocabulary file can then be created, again using View (or other word processor). The syntax required for this file is:

<word defined>=<definition><blank line>

where the definition can take a free format extending over more than one line. There is a limit of 250 words per file, while the complete entry for each word must be no greater than 250 characters. There is an overall memory restriction of about 9000 characters for a 32K machine. If more space is required, then a number of separate files can be employed.

```
cathédrale=large church.

librairie=bookshop.

ordinateur=computer.

verre=glass.

paysan=Someone who lives in the
country and makes their living
from agriculture.

étudiant=student.

autoroute=motorway.
```

Figure 2. Part of a vocabulary file

TESTING YOUR VOCABULARY

To test your knowledge of the foreign vocabulary, you will need to run the program listed here with the screen still set up for the French character set. This program should be typed in and a copy saved before you proceed further. When prompted for a file, type:

```
E.FRVOCA
```

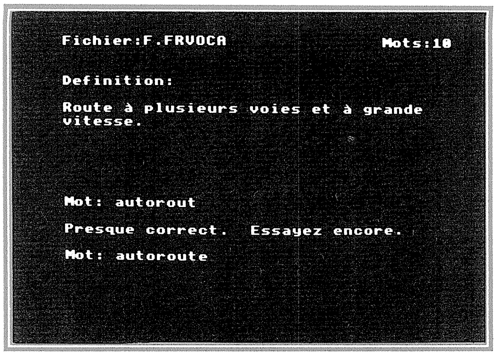
(including the directory letter).

The program then reads both the instructions and the vocabulary, and then presents one word or phrase at a time to test you. When typing in your response, you will need to make sure that accented characters are correct, and that you have made correct use of upper and lower case letters as required. Some languages (e.g. German) make specific use of upper case letters so the program is sensitive to case. You should remember this when creating any vocabulary file.

If the program detects that your answer is nearly right it will give you a second opportunity to answer. If you do not know the answer at all then just press Return. The program will supply the correct answer if you fail to find it. After each word or phrase has been dealt with, pressing 'C' will continue to the next. Words or phrases are presented in random order, those that you answer correctly being presented less frequently, and those that you do not answer correctly being presented more often.

The program also keeps a (hopefully reducing) count of the number of words you have yet to recognise correctly. Once all the words have been learnt, or after pressing 'F' to leave the program, you will find that a new file E.FRVOCZ has been created. This file contains your progress for this and subsequent sessions and will be used by the VocProg program to be described next month.

If you notice any mistakes to the vocabulary during the running of the program, then corrections can be made to the vocabulary file using View. If you wish to add more words, then you should do so at the *end* of the file. Of course the value of the program will be only as good as the words in the vocabulary file. Care should be taken when entering data, and any spellings which don't appear correct should be checked with a dictionary.



```
Fichier:F.FRVOCA                    Mots:10

Definition:

Route à plusieurs voies et à grande
vitesse.



Mot: autorout

Presque correct.  Essayez encore.

Mot: autoroute
```

Instructions and vocabulary in French

Due to the restrictions of the character sets, use of the program for Greek and Turkish must be only in the target language, unless only the lower case letters are redefined, with the definitions in English in capitals.

Au revoir.

The magazine disc/tape also contains the French-only versions of the instruction and vocabulary files as well as the French-English versions, and the programs to define the additional French characters needed. We have also included the character definitions for German, Greek and Turkish repeated from Vol.7 No.4. Next month, in addition to the program and files to be described in the article, we will also include character definitions for Italian, Portuguese and the Scandinavian group of languages.

```
   10 REM Program VocTest
   20 REM Version B1.0
   30 REM Author  Eddy Hunt
   40 REM BEEBUG  June 1989
   50 REM Program subject to copyright
   60 :
  100 ON ERROR GOTO 420
  110 DIM st$(250),ft$(2)
  120 ft$(0)="REVISION"
  130 ft$(1)="CURRENT":ft$(2)="NEW"
  140 wd%=40:NF%=0
  150 f$="FILE"
  160 MODE7
  170 VDU 23,1,0;0;0;0;
  180 PRINTTAB(11,2)CHR$131;"Vocabulary
Tester"
  190 fn$=FNins(f$,5)
  200 PROCiread
  210 PRINT'ft$(-NF%-NR%);" ";f$''"Loadi
ng"
  220 PROCread
  230 CLOSE#C%
  240 VDU7:MODE6
  250 VDU 23,1,0;0;0;0;
  260 nw%=n%-1:ca%=0:er%=0
  270 L%=LEN(ix$)-3
  280 IF L%<nw% THEN ix$=ix$+STRING$(nw%
-L%,"M")
  290 rd$=""
  300 FOR i%=1 TO nw%:rd$=rd$+CHR$(i%):N
EXT
  310 wv$=LEFT$(ix$,3):ix$=MID$(ix$,4)
  320 IF NOT NR% THEN ix$=STRING$(nw%,"M
"):fn$=fn$+"(R)"
  330 h$=f$+":"+fn$+STRING$(34-LEN(f$)-L
EN(fn$)-LEN(wpl$)," ")+wpl$+":"
  340 REPEAT
  350 PROCtest
  360 PROCspc
  370 UNTIL EX% OR nw%=0
  380 IF NR% THEN PROCiwr
  390 MODE7
  400 END
  410 :
  420 MODE7:CLOSE#0
  430 REPORT:PRINT" at line ";ERL
  440 END
  450 :
 1000 DEF PROCread
 1010 EL%=0
 1020 if$=LEFT$(fn$,2)+"INSTRUC"
 1030 PROCof(if$)
 1040 f$=FNstin:m$=FNstin:w$=FNstin
 1050 wpl$=FNstin:rt$=FNstin:wr$=FNstin
 1060 ct$=FNstin:stop$=FNstin:tr$=FNstin
 1070 ct%=ASC(ct$) AND &DF
 1080 stop%=ASC(stop$) AND &DF
 1090 CLOSE#C%
 1100 PROCof(fn$)
 1110 n%=0
 1120 REPEAT
 1130 n%=n%+1:PRINT".";
 1140 x$=""
 1150 x%=BGET#C%:IF EOF#C% GOTO 1190
 1160 x$=x$+FNchr(x%):IF NOT ES% GOTO 11
50
 1170 st$(n%)=x$
 1180 IF LEN(x$)<4 GOTO 1140
 1190 UNTIL EOF#C%
 1200 ENDPROC
 1210 :
 1220 DEF PROCtest
 1230 CLS:PRINT'h$;nw%
 1240 r%=FNrnd
 1250 tx$=st$(r%)
 1260 p%=INSTR(tx$,"=")
 1270 ans$=LEFT$(tx$,p%-1)
 1280 tx$=MID$(tx$,p%+1)
 1290 PRINTTAB(0,4)m$;":"
 1300 PROCtxout(5):tr%=0
 1310 inp$=FNins(w$,13+4*tr%)
 1320 CA%=inp$=ans$
 1330 IF CA% PRINTTAB(0,19)rt$:ca%=ca%+1
:PROCdel(rp%):GOTO 1360
 1340 IF tr%>0 OR FNmv<0.65 THEN PRINTTA
B(0,19)wr$:PRINT rt$;": ";ans$:er%=er%+1
 :GOTO 1360
 1350 tr%=tr%+1:PRINTTAB(0,15)tr$:GOTO 1
310
 1360 PRINTTAB(0,22)rt$;":";ca%:PRINTTAB
(20,22)wr$;":";er%
 1370 ix$=FNincs
 1380 ENDPROC
 1390 :
 1400 DEF PROCiread
 1410 ixf$=LEFT$(fn$,LEN(fn$)-1)+"Z"
 1420 C%=OPENUP(ixf$)
 1430 v$=RIGHT$(fn$,1)
 1440 IF C%=0 THEN ix$=CHR$0+CHR$0+v$:NR
%=-1:NF%=-1:ENDPROC
 1450 INPUT#C%,ix$:CLOSE#C%
 1460 NR%=MID$(ix$,3,1)=v$
 1470 ENDPROC
 1480 :
 1490 DEF PROCiwr
 1500 C%=OPENOUT(ixf$)
 1510 PRINT#C%,wv$+ix$
 1520 CLOSE#C%
 1530 ENDPROC
 1540 :
 1550 DEF PROCtxout(L%)
 1560 PRINTTAB(0,L%)
 1570 IF LEN(tx$)<=wd% GOTO 1620
 1580 p%=wd%+1:REPEAT:p%=p%-1:UNTIL MID$
(tx$,p%,1)=" "
 1590 PRINT LEFT$(tx$,p%-1)
 1600 tx$=MID$(tx$,p%+1)
 1610 IF LEN(tx$)>wd% GOTO 1580
 1620 PRINT tx$
 1630 ENDPROC
 1640 :
 1650 DEF PROCof(xf$)
 1660 C%=OPENUP(xf$)
 1670 IF C%>0 THEN ENDPROC
 1680 PRINT"File missing: ";xf$
 1690 END
 1700 :
 1710 DEF FNincs
 1720 L%=ASC(MID$(ix$,r%,1))
 1730 =LEFT$(ix$,r%-1)+CHR$(L%-2*CA%-1)+
MID$(ix$,r%+1)
 1740 DEF FNchr(x%)
 1750 IF x%=26 THEN =""
 1760 ES%=EL% AND x%=13
 1770 EL%=x%=13
 1780 IF x%<32 THEN =" "
 1790 =CHR$(x%)
 1800 :
 1810 DEF PROCspc
 1820 PRINTTAB(0,24)CHR$(ct%);":";ct$;"
.. ";CHR$(stop%);":";stop$;
 1830 REPEAT
 1840 g%=GET AND NOT 32
 1850 UNTIL g%=ct% OR g%=stop%
 1860 EX%=g%=stop%
 1870 ENDPROC
 1880 :
 1890 DEF FNstin
 1900 x$=""
 1910 REPEAT
 1920 x%=BGET#C%
 1930 IF x%<>13 x$=x$+CHR$(x%)
 1940 UNTIL x%=13
 1950 =x$
 1960 :
 1970 DEF PROCdel(x%)
 1980 rd$=LEFT$(rd$,x%-1)+MID$(rd$,x%+1)
 1990 nw%=nw%-1
 2000 ENDPROC
 2010 :
 2020 DEF FNins(x$,L%)
 2030 PRINTTAB(0,L%)x$;":"
 2040 xp%=LEN(x$)+1:x$="":p%=0
 2050 REPEAT
 2060 g%=GET
 2070 IF g%=13 OR g%=127 AND p%=0 GOTO 2
100
 2080 IF g%=127 THEN p%=p%-1:x$=LEFT$(x$
,p%):c$=" " ELSE c$=CHR$(g%):x$=x$+c$:p%
=p%+1
 2090 PRINTTAB(xp%+p%-(g%=127),L%)c$
 2100 UNTIL g%=13
 2110 =x$
 2120 :
 2130 DEF FNrnd
 2140 w=RND(-TIME):L%=ASC("L")
 2150 IF nw%=1 THEN rp%=1 ELSE rp%=RND(n
w%)
 2160 L%=L%+1
 2170 r%=ASC(MID$(rd$,rp%,1))
 2180 IF NR% AND CHR$(L%)<MID$(ix$,r%,1)
 GOTO 2150
 2190 =r%
 2200 :
 2210 DEF FNmv
 2220 w=0:L%=LEN(inp$):dx%=9:t$=ans$
 2230 FOR p%=1 TO L%
 2240 c$=MID$(inp$,p%,1)
 2250 q%=INSTR(t$,c$)
 2260 IF q%>0 t$=LEFT$(t$,q%-1)+"!"+MID$
(t$,q%+1):dx%=FNam(dx%,p%-q%):w=w+.5^(AB
S(dx%))
 2270 NEXT
 2280 =w/FNmx(L%,LEN(ans$))
 2290 :
 2300 DEF FNam(a%,b%)
 2310 IF ABS(a%)<ABS(b%) THEN =a% ELSE =
b%
 2320 :
 2330 DEF FNmx(a%,b%)
 2340 IF a%>b% THEN =a% ELSE =b%
```

Basic Line Editor

Paul Pibworth presents a utility to simplify the editing of Basic programs.

In the course of my employment I have to work with both Acorn and RML machines. There is one feature of the latter that I have envied, and that is the ability to edit a single line in Basic, as though it were on a word processor. On a Beeb, you must either list the line in question and then copy it, making any changes necessary, or, on a Master 128, use Edit to edit the entire program. This weakness is cured by the program presented here. This assembles a machine code line editor which is invoked with a single function key press.

ENTERING THE PROGRAM

The listing given here should be typed in and saved, using a filename other than 'ED'. When the program is run, it assembles the machine code and saves it with 'ED' as the filename. The code is just over 1K long, and is assembled to run at address &7700, which puts it just under the mode 7 screen. If a different mode is used, the address in line 100 should be changed to move the code. For shadow modes, an address of &7B00 is suitable, while for mode 0, &2B00 must be used. You might also wish to lower the value of HIMEM to prevent the code from being overwritten accidentally.

As it stands, the program is designed for use on a Master 128. To use the program on a machine with Basic II, add the following lines to the listing:

```
 120 rom1=&67:rom2=&80
 190 inslin=&BC8D
2525 LDA#5:STA&37:LDA#7:STA&38:JSR&8951
2545 LDY#5
```

For a Compact, the changes needed are:

```
 120 rom1=&23:rom2=&84
 190 inslin=&BA3B
```

USING THE EDITOR

Before use, the editor must be installed. This is done automatically when the code is assembled, and can be done thereafter by typing:

```
  *ED
```

(Users with cassette systems will instead need to *LOAD ED, and then CALL the address at which it was assembled.)

Once installed, pressing function key f0 will pop-up a window at the bottom of the screen, the height depending on the mode in use. You are then prompted to enter the line number of the line to be edited, followed by Return. You can press Escape at this point to abort the operation. The requested line is displayed in the window, with the cursor positioned at the start. The cursor left and right keys can be used to move the cursor along the line, and typing a character will insert it at the current cursor position. Pressing Delete will remove the character to the left of the cursor, closing up the resulting gap. Once editing is complete, pressing Return will enter the amended line into the program, while Escape will abort the editing without changing the program.

HOW IT WORKS

When called, the code sets up some flags, and then sets up a window at the bottom of the screen. A set of tables is used to determine the window dimensions for the particular mode. In mode 7, a left margin of colour control characters is set up so that the window's contents will appear in green.

The number of the line to be edited is entered in decimal. This number is then converted into the format used by Basic to store line numbers. Assuming that it is a valid line number, the line is then read byte by byte, being detokenised as it goes, into a buffer at address &A00. Any references to line numbers within a Basic program (GOTO etc) are also stored in a tokenised form. These need to be converted to decimal, and placed into the buffer, in the relevant format, i.e. GOTO123 and not GOTO00123. When transfer is complete, the cursor is placed in the first position after the line number.

The cursor keys are then programmed to return ASCII codes, and a scanning loop entered. This responds to key presses, and performs the moving of the cursor, and inserting and deleting of characters, as necessary. This continues until either Return or Escape is pressed.

The part of the program that caused the most problems was the re-inserting of the line, and I am indebted to Mr. M. Plumley (author of BASIC ROM User Guide) for his help and advice. The line in the buffer is transferred to page &700, and a procedure at &BAEB takes care of the rest. I also used his book to learn how to deal with tokenised numbers within a line.

Before exiting, the window is cleared, and restored to full screen size, and the cursor is restored to where it was when the editor was invoked.

```
   10 REM Program Basic Line Editor
   20 REM Version B1.00
   30 REM Author  Paul Pibworth
   40 REM With help from Mark Plumley
   50 REM BEEBUG  June 1989
   60 REM Program subject to copyright
   70 :
  100 code=&7700
  110 ptr1=&70:ptr2=&72
  120 rom1=&4C:rom2=&84:REM BASIC 4 ROM
token addresses
  130 store=&74
  140 xpos=&76:ypos=&77:mode=&78
  150 width=&79:len=&7A:count=&7C
  160 dec=&A00:REM hold decimal number
  170 hex=&85:REM hold hex number
  180 mem=&87:rem=&89:quote=&8A:REM Flag
s for REMs and Quotes
  190 inslin=&BAEB:REM Insert line
  200 FORpass=0TO3 STEP 3:P%=code
  210 [OPT pass
  220 LDX #keycomm MOD &100
  230 LDY #keycomm DIV &100:JMP &FFF7
  240 .keycomm EQUS "KEY 0 CALL&"+STR$~s
tart+"|M":EQUB 13
  250 .start:LDA #0:STA rem:STA quote
  260 LDA #&86:JSR &FFF4
  270 STX xpos:STY ypos
  280 LDA #&87:JSR &FFF4:STY mode
  290 LDA #28:JSR &FFEE
  300 LDA #0:JSR &FFEE
  310 LDA bot,Y:JSR &FFEE
  320 LDA right,Y:JSR &FFEE:STA width
  330 LDA top,Y:JSR &FFEE
  340 LDA #12:JSR &FFEE
  350 CPY #7:BEQ tt:INC width
  360 BNE ask
  370 .tt LDX #7
  380 .green JSR &FFE7:LDA #130
  390 JSR &FFEE:DEX:BNE green
  400 LDA #28:JSR &FFEE
  410 LDA #1:JSR &FFEE
  420 LDA bot,Y:JSR &FFEE
  430 LDA right,Y:JSR &FFEE
  440 LDA top,Y:JSR &FFEE
  450 LDA #30:JSR &FFEE
  460 .ask LDX #mess1 MOD256
  470 LDY #mess1 DIV256:JSR display
  480 :
  490 \ take in decimal num
  500 .wipe LDX #blank MOD256
  510 LDY #blank DIV256:JSR display
  520 LDX #0:LDA #32
  530 .zero STA dec,X:INX:CPX #5
  540 BNE zero
  550 .inputloop LDY #0
  560 .keypress JSR &FFE0:CMP #13
  570 BEQ complete:CMP #127:BEQ wipe
  580 CMP #27:BEQ complete:PHA:LDX #1
  590 .shift LDA dec,X:DEX:STA dec,X
  600 INX:INX:CPX #5:BNE shift:DEX
  610 PLA:STA dec,X:JSR &FFE3:INY
  620 CPY #5:BNE keypress
  630 .complete
  640 CMP #27:BNE notesc:JMP esc
  650 .notesc
  660 \ convert dec>hex, store in "hex"
  670 LDA #0:STA hex:STA hex+1
  680 LDY #5      \ counts 5 digits
  690 .con1 DEY:TYA:TAX:LDA dec,X
  700 TAX:CPX #32:BEQ con3
  710 CPX #(ASC"0"):BNE add
  720 .con2 CPY #0:BNE con1
  730 .con3 JMP search
  740 .add LDA hex:CLC:ADC lowbytes,Y
  750 STA hex:LDA hex+1:ADC hibytes,Y
  760 STA hex+1:DEX:CPX #(ASC"0")
  770 BNE add:JMP con2
  780 :
  790 \ look to see if line no. valid
  800 .search
  810 LDA #1     \ get page
  820 STA ptr2:LDA &18:STA ptr2+1
  830 .searchloop LDY #0:LDA (ptr2),Y
  840 CMP #255:BEQ noline:CMP hex+1
  850 BNE linelen1:INY:LDA (ptr2),Y
  860 CMP hex:BNE linelen2:JMP found
  870 .linelen1 INY
  880 .linelen2 INY:LDA (ptr2),Y
  890 CLC:ADC ptr2:STA ptr2:LDA ptr2+1
  900 ADC #0:STA ptr2+1:JMP searchloop
  910 .found LDY #0:CLC:LDA ptr2:ADC #2
  920 STA mem    \ mem loc of linelen
  930 LDA ptr2+1:ADC #0:STA mem+1
  940 JMP read1
  950 :
  960 .noline LDX #mess2 MOD 256
  970 LDY #mess2 DIV 256:JSR display
  980 JMP ask
  990 :
 1000 \ read and detokenise
 1010 \ put into buffer (from&A05)
 1020 \ X=position in &A00
 1030 \ Y=position in orig line
 1040 .read1 LDX #5:LDY #0
 1050 LDA #12:JSR &FFEE
 1060 .read2 INY:TYA:PHA \ keep Y on the
 stack
 1070 LDA (mem),Y:STA &70 \ read and sto
re a byte
 1080 CMP #13:BEQ get4 \ is it end of li
ne
 1090 CMP #34:BNE iftoken \ 34=quote
 1100 LDA #255:EOR &8A:STA &8A \ set/uns
et quote flag
 1110 LDA &70 \ reload the byte
 1120 .iftoken CLC:CMP #&80:BCS detok \
is it a token
 1130 .buffer STA &A00,X:INX \ put into
buffer
 1140 .get3 PLA:TAY:JMP read2
 1150 .get4 PLA:JMP edit
 1160 :
 1170 .detok LDA #0:CMP &89:BEQ tok1 \ c
heck if REM flag set
 1180 LDA &70:JMP buffer
 1190 .tok1 CMP &8A:BEQ tok2 \ ch if quo
te flag set
 1200 LDA &70:JMP buffer
 1210 .tok2 LDA &70:CMP #&8D
 1220 BEQ numtoken
 1230 \ now set up for BASIC 4
 1240 LDA #rom1:STA &71
 1250 LDA #rom2:STA &72
 1260 :
 1270 .tok3 LDA &70:CMP #&F4:BNE tok4
 1280 LDA #1:STA &89   \ set flag on REM
 1290 .tok4
 1300 \ now look for token in ROM
 1310 \ &71 & &72 hold TOKEN TABLE addre
sses, less10
 1320 LDY #10
 1330 .tok5
 1340 LDA (&71),Y:CMP &70:BEQ table
 1350 CLC:LDA &71:ADC #1:STA &71
 1360 LDA &72:ADC #0:STA &72:JMP tok5
 1370 .table
 1380 \ now go back to prev token
 1390 DEY:LDA (&71),Y:CLC:CMP #&70:BCC t
able
 1400 \ now go foreward two, unless it i
s AND
 1410 \ (AND is preceeded by &81)
 1420 CLC:CMP #&81:BEQ table2:INY
 1430 .table2
 1440 \ read each byte, put in buffer if
 <127
 1450 INY:LDA (&71),Y:CLC:CMP #&7F:BCS t
able3
 1460 STA &A00,X:INX:JMP table2
 1470 .table3 JMP get3
 1480 :
 1490 \ Detokenise a tokenised number
 1500 \ Store in location "hex"
 1510 \ Convert from hex to dec
 1520 \ Put into buffer, with no leading
 zeros
 1530 .numtoken
 1540 INY:LDA (mem),Y:ASL A:ASL A:PHA:AN
D #&C0
 1550 INY:EOR (mem),Y:STA hex+1
 1560 INY:PLA:ASL A:ASL A:EOR (mem),Y:ST
A hex
 1570 :
 1580 .hextodec \ convert detok line num
 1590 LDA #48:STA &80:STA &81:STA &82
 1600 STA &83:STA &84:TXA:PHA:LDY #255
 1610 .htd2 INY:CPY #5:BEQ htd6:LDX #48
 1620 .htd3:INX:LDA hex:CMP hibytes,Y
 1630 BEQ htd4:BCS htd5:JMP htd2
 1640 .htd4 LDA hex+1:CMP lowbytes,Y
 1650 BCS htd5:JMP htd2
 1660 .htd5 SEC:LDA hex+1:SBC lowbytes,Y
 1670 STA hex+1:LDA hex:SBC hibytes,Y
 1680 STA hex:STX &80,Y:JMP htd3
 1690 .htd6    \ trans to buffer, drop l
eadinf zeros
 1700 PLA:TAX:LDY #255
 1710 .htd7 INY:CPY #4:BEQ htd8
 1720 LDA &80,Y:CMP #48:BEQ htd7
 1730 .htd8 LDA &80,Y:STA &A00,X:CPY #4
 1740 BEQ htd9:INX:INY:JMP htd8
 1750 .htd9 PLA:TAY:INY:INY:INY:INX
 1760 JMP read2
 1770 :
 1780 .edit STX len
 1790 LDA #13:STA &A00,X:LDA #0
 1800 .edit2 CPX #255:BEQ edit3
 1810 INX:STA &A00,X:JMP edit2
 1820 .edit3:LDA #4:LDX #1:JSR &FFF4 \ *
FX4,1
 1830 JSR printline
 1840 .cursor:LDY #5 \ set cursor
 1850 JSR cursorplace
 1860 .cursor2:JSR &FFE0:CMP #13
 1870 BEQ ret:CMP #27:BEQ esc
 1880 CMP #136:BEQ goleft:CMP #137
 1890 BEQ goright:CMP #0:BMI cursor2
```

```
1900 CMP #127:BEQ delete:CMP #32
1910 BCS insert
1920 .esc LDA #&7E:JSR &FFF4:JMP fx40
1930 :
1940 .goleft CPY #0:BNE gl2:JMP cursor2
1950 .gl2 JSR downcount:JSR cursorplace
1960 JMP cursor2
1970 .goright CPY len:BNE gr2
1980 JMP cursor2
1990 .gr2 JSR upcount:JSR cursorplace
2000 JMP cursor2
2010 :
2020 .delete CPY #0:BNE del1
2030 JMP cursor2
2040 .del1 JSR del2:JSR printline
2050 JMP goleft
2060 :
2070 .insert
2080 LDX len:CPX #255:BNE ins1
2090 LDA #7:JSR &FFEE:JMP cursor2
2100 .ins1 JSR ins2:JSR printline
2110 JMP goright
2120 :
2130 .ret JSR fx40:JMP ret2
2140 :
2150 .upcount \ Incr count
2160 INY:RTS
2170 :
2180 .downcount \ dec count
2190 DEY:RTS
2200 :
2210 .cursorplace STY store:TYA
2220 LDY #0
2230 .curpl2 SEC:SBC width:INY
2240 BCS curpl2:ADC width:DEY:PHA
2250 LDA #31:JSR &FFEE:PLA:JSR &FFEE
2260 TYA:JSR &FFEE:LDY store:RTS
2270 :
2280 .del2 STY store:TYA:TAX:DEX
2290 JMP contract2
2300 .contract LDA &A00,Y:STA &A00,X
2310 INX:INY
2320 .contract2 CPY len:BNE contract
2330 LDA #13:STA &A00,X
2340 LDA #0:STA &A00,Y:DEC len
2350 LDY store:RTS
2360 :
2370 .ins2 STA store+1:STY store
2380 LDY len:INC len:LDX len
2390 .expand LDA &A00,Y:STA &A00,X
2400 DEX:DEY:CPX store:BNE expand
2410 LDA store+1:STA &A00,X:LDY store
2420 RTS
2430 :
2440 .printline STY store
2450 LDA #30:JSR &FFEE:LDX #0
2460 .pr2 LDA &A00,X:JSR &FFEE:INX
2470 CPX len:BNE pr2:LDA #32
2480 JSR &FFEE:LDY store:RTS
2490 :
2500 .ret2 LDX #255:.ret3:INX
2510 LDA &A00,X:STA &700,X:CPX len
2520 BNE ret3
2530 LDA #0:STA &B
2540 LDA #7:STA &C
2550 JMP inslin
2560 :
2570 .fx40 LDA #4:LDX #0:JSR &FFF4
2580 :
2590 .reset LDY mode:CPY #7:BNE nottt
2600 LDA #28:JSR &FFEE
2610 LDA #0:JSR &FFEE
2620 LDA bot,Y:JSR &FFEE
2630 LDA right,Y:JSR &FFEE
2640 LDA top,Y:JSR &FFEE
2650 .nottt LDA #12:JSR &FFEE
2660 LDA #26:JSR &FFEE:LDA #31
2670 JSR &FFEE:LDA xpos:JSR &FFEE
2680 LDA ypos:JMP &FFEE
2690 :
2700 .display STX ptr2:STY ptr2+1
2710 LDY #0:LDA (ptr2),Y
2720 .display2 JSR &FFEE:INY
2730 LDA (ptr2),Y:CMP #255
2740 BNE display2:RTS
2750 :
2760 .lowbytes EQUB 16:EQUB 232
2770 EQUB 100:EQUB 10:EQUB 1
2780 .hibytes EQUB 39:EQUB 3
2790 EQUB 0:EQUB 0:EQUB 0
2800 .mess1
2810 EQUS"Enter line number to edit."
2820 EQUW &D0A:EQUB 255
2830 .mess2
2840 EQUB 7:EQUS"No Such Line !"
2850 EQUW &D0A:EQUB 255
2860 .blank EQUB 13
2870 EQUS STRING$(10," ")
2880 EQUB 13:EQUB 255
2890 .right EQUB 79:EQUB 39:EQUB 19
2900 EQUB 79:EQUB 39:EQUB 19
2910 EQUB 39:EQUB 39
2920 .bot EQUB 31:EQUB 31:EQUB 31
2930 EQUB 24:EQUB 31:EQUB 31
2940 EQUB 24:EQUB 24
2950 .top EQUB 28:EQUB 25:EQUB 19
2960 EQUB 21:EQUB 25:EQUB 19
2970 EQUB 18:EQUB 18
2980 ]
2990 NEXT
3000 A$="SAVE ED "+STR$~code+" "+STR$~P
%
3010 PRINT A$:OSCLI A$
3020 CALL code
```

Genealogy Without Tears

Hector Goodman describes how he uses his computer for family history, without resorting to specialised applications software.

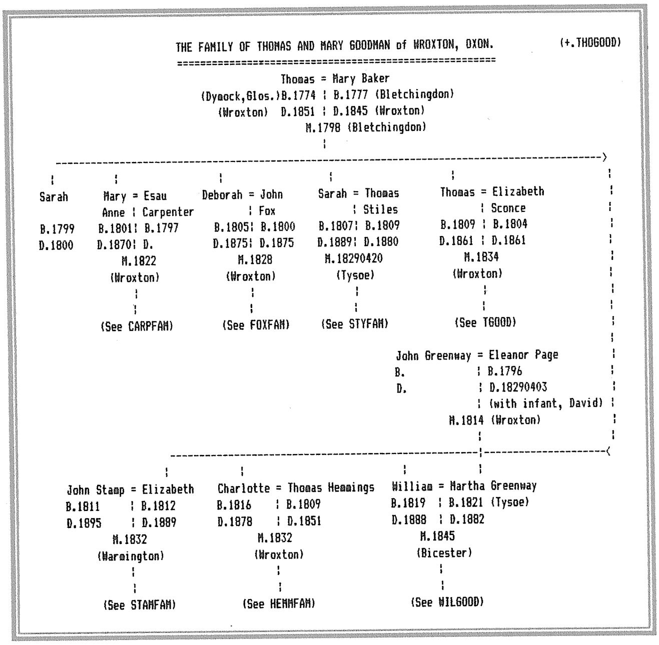
In the last five years, in common with many other people, I have been researching the history of four related families including my own.

As a result of extracting data from scores of parish registers, Census returns, the International Genealogical Index, the Registers of births, marriages and deaths at St. Catherine's House, wills at Somerset House and many other sources of information, it soon became apparent that I urgently needed a good filing system for collating and sorting the mass of data that was growing almost by the day.

I already had the View word processor in my computer, but it wasn't until after I had made a number of unsuccessful attempts at constructing family trees longhand on large sheets of paper, it occurred to me that I could exploit View for this purpose. By breaking the family lineage up into a series of smaller trees, I found I could accommodate each quite satisfactorily using View and printing on A4 paper (see Fig.1). I use the full width of the View screen (132 characters) and adopt the condensed print mode.

As I have had to compile many such trees, I find the ViewSheet AUTOBOOT program (BEEBUG Vol.4 No.8), which I have adapted for View, invaluable as a means of selecting the required file on entry.

I also use View to keep a record of extracts of Census returns and parish registers in chronological order as in Fig.2.



```
                     THE FAMILY OF THOMAS AND MARY GOODMAN of WROXTON, OXON.          (+.THOGOOD)
                     ======================================================
                                     Thomas = Mary Baker
                          (Dymock,Glos.)B.1774 ¦ B.1777 (Bletchingdon)
                             (Wroxton)  D.1851 ¦ D.1845 (Wroxton)
                                              M.1798 (Bletchingdon)
                                                 ¦
 ---------------------------------------------------------------------------------------------->
 ¦          ¦               ¦                  ¦                     ¦                          ¦
Sarah     Mary = Esau     Deborah = John     Sarah = Thomas       Thomas = Elizabeth            ¦
          Anne ¦ Carpenter         ¦ Fox            ¦ Stiles             ¦ Sconce               ¦
B.1799   B.1801¦ B.1797     B.1805¦ B.1800    B.1807¦ B.1809       B.1809 ¦ B.1804              ¦
D.1800   D.1870¦ D.         D.1875¦ D.1875    D.1889¦ D.1880       D.1861 ¦ D.1861              ¦
             M.1822            M.1828          M.18290420             M.1834                    ¦
           (Wroxton)         (Wroxton)          (Tysoe)              (Wroxton)                  ¦
               ¦                 ¦                 ¦                     ¦                      ¦
               ¦                 ¦                 ¦                     ¦                      ¦
          (See CARPFAM)      (See FOXFAM)     (See STYFAM)          (See TGOOD)                 ¦
                                                                                                ¦
                                                         John Greenway = Eleanor Page           ¦
                                                         B.            ¦ B.1796                 ¦
                                                         D.            ¦ D.18290403             ¦
                                                                       ¦ (with infant, David)   ¦
                                                                  M.1814 (Wroxton)              ¦
                                                                       ¦                        ¦
                  ------------------------------------------------------¦-----------------------<
                         ¦           ¦                      ¦          ¦
     John Stamp = Elizabeth    Charlotte = Thomas Hemmings   William = Martha Greenway
     B.1811     ¦ B.1812       B.1816    ¦ B.1809            B.1819 ¦ B.1821 (Tysoe)
     D.1895     ¦ D.1889       D.1878    ¦ D.1851            D.1888 ¦ D.1882
            M.1832                   M.1832                      M.1845
         (Warmington)              (Wroxton)                   (Bicester)
              ¦                        ¦                           ¦
              ¦                        ¦                           ¦
         (See STAMFAM)            (See HEMMFAM)               (See WILGOOD)
```

Figure 1. Part of a family tree using View

View on its own, however, did not solve my problem of how to store the basic data in a convenient and readily accessible form, but quite by chance I came across the Masterfile II program and realized that this could be the answer to my prayer. I don't think I could have made a better choice!

My first task was to compile the record description (Menu program B). After trying various formats, I found the following was best suited to my particular needs:

FIELD	TYPE	LENGTH	TITLE
1	S	13	SURNAME
2	S	13	MAIDEN NAME
3	S	22	FORENAMES
4	S	1	EVENT
2	S	8	DATE
6	S	25	PARENTS/SPOUSE
7	S	20	PARISH/COUNTY
8	S	8	SOURCE (of data)
9	S	120	NOTES

Note that EVENT refers to such as Birth/Bapt., Marriage, Burial/Death. Also, I print my dates in reverse order, e.g. 12th July 1885 becomes 18850712.

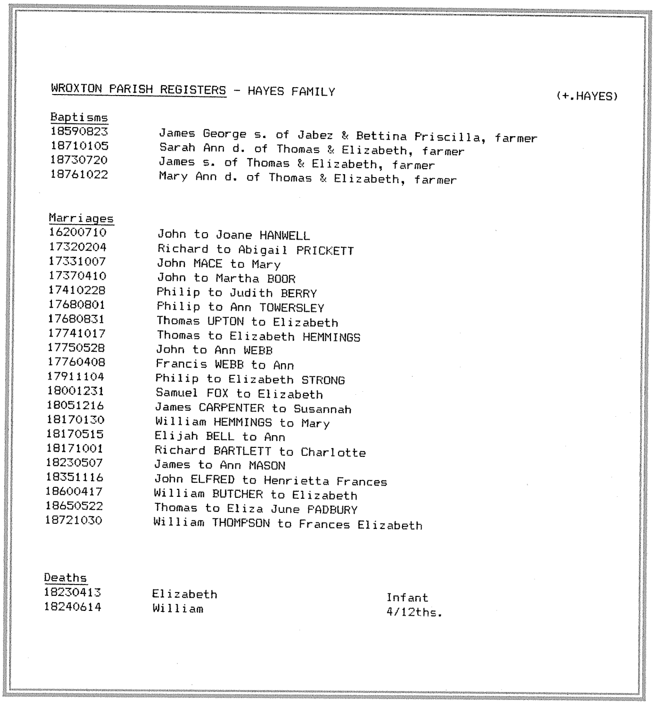
By using a new disc solely for each family file, I find that this description (230 characters in 9 fields) allows for a capacity of 400 records per 100K disc (generally sufficient, I would think, for one family). The format that I have adopted may not suit everyone, but the beauty of the Masterfile II program is that its extraordinary flexibility caters for virtually unlimited alternatives. It also permits one to make any alterations and/or additions to a record as further historical data comes to hand. I ALWAYS make a backup copy of each file - to lose a whole family's data accidentally would be disastrous.

Prior to printing to my particular requirements, I first tag-sort the file in alphabetical order of forenames (field 3), and include the record serial number, as the accompanying specimen indicates. When printing, I use the following configuration (Masterfile menu program D):

1. Printer line length 132
2. Left margin 5 (allows for filing)
3. Control codes 27,15,27,78,6 (condensed print)

The control codes invoke the skip-over function when using continuous stationery, and condensed print.

The above-mentioned procedures enable me to have all my data in a most convenient form for reference, and where necessary, for further printing or sorting purposes (see sample printout in Fig.3).



```
WROXTON PARISH REGISTERS - HAYES FAMILY                                    (+.HAYES)

Baptisms
18590823      James George s. of Jabez & Bettina Priscilla, farmer
18710105      Sarah Ann d. of Thomas & Elizabeth, farmer
18730720      James s. of Thomas & Elizabeth, farmer
18761022      Mary Ann d. of Thomas & Elizabeth, farmer


Marriages
16200710      John to Joane HANWELL
17320204      Richard to Abigail PRICKETT
17331007      John MACE to Mary
17370410      John to Martha BOOR
17410228      Philip to Judith BERRY
17680801      Philip to Ann TOWERSLEY
17680831      Thomas UPTON to Elizabeth
17741017      Thomas to Elizabeth HEMMINGS
17750528      John to Ann WEBB
17760408      Francis WEBB to Ann
17911104      Philip to Elizabeth STRONG
18001231      Samuel FOX to Elizabeth
18051216      James CARPENTER to Susannah
18170130      William HEMMINGS to Mary
18170515      Elijah BELL to Ann
18171001      Richard BARTLETT to Charlotte
18230507      James to Ann MASON
18351116      John ELFRED to Henrietta Frances
18600417      William BUTCHER to Elizabeth
18650522      Thomas to Eliza June PADBURY
18721030      William THOMPSON to Frances Elizabeth


Deaths
18230413      Elizabeth                              Infant
18240614      William                                4/12ths.
```

Figure 2. Parish register details using View

Quite apart from their many other uses, I can thoroughly recommend these programs to any reader who has, or is thinking of taking up genealogy, and doesn't want to go to the expense of specialist software.

```
REC  SURNAME       MAID/NAME     FORENAMES           EV DATE     PARENTS/SPOUSE          PARISH/COUNTY        SOURCE
NOTES
---  -------       ---------     ---------           -- ----     --------------          -------------        ------

  1 GOODMAN                      Percy Oliver        M 1930.... Ada E.Golby             Banbury, Oxon.       CRO.
M.Apl./Jun.1930. 1930-1936-no record of any issue.
  2

  3 GOODMAN                      Ada May             M          Harold William          ? Wroxton, Oxon.     PR.

  4 LAKEY          GOODMAN       Aileen              D 1969.... Fred Lakey

  5 GOODMAN                      Aileen              B 19291208 Chas.James & Florence.  Finsbury Park.

  6 GOODMAN                      Alban               M 18600926 Hannah Hayes            Banbury, Oxon.       PR.

  7 GOODMAN                      Alban               B 18350903 Thomas & Elizabeth.     Wroxton, Oxon.       PR.

  8 GOODMAN        WINNIATT      Ann                 M 17261022 Richard                 Dymock,Glos.         PR.IGI.

  9 STEEVENS       GOODMAN       Anne                M 17451112 (1)John Steevens.       Dymock,Glos.         PR.IGI.
m.(2) John Banks on 17471226.
 10 GOODMAN                      Anne                B 17280423 Richard & Ann.          Dymock,Glos.         PR.IGI.

 11 GOODMAN                      Anne Maria          B 18390213 Thomas & Elizabeth.     Wroxton, Oxon.       PR.
```

Figure 3. Sample printing using Masterfile II

View is an Acorn product, supplied as part of a Master 128 or Compact system, but may be purchased for the model B for £42.06. Masterfile II is a BEEBUG product costing £16.50. Prices quoted are for BEEBUG and RISC User members only, post & packing extra. See the latest Retail Catalogue for full details.

Sheridan Williams, author of Masterfile, also uses both View and Masterfile for genealogical purposes, and echoes Mr Goodman's comments. He has found another use for Masterfile in making sense out of unorganised data. For example, if you obtain the Mormon records for your family name, you will find that they are ordered on Name as shown in figure 4. However, if you now use Masterfile to produce 'tag' sorts on Parents/Child, Year and Place, and print out the new ordered lists, all sorts of connections become obvious.

NAME	PARENTS/CHILD	TYPE	YEAR	PLACE
AGNES	JOHN	M	1622	FAREHAM
ANN	JOHN	C	1704	PORTSEA ST T
ANN	RICHARD/CATHERINE	C	1729	BRAMBRIDGE
ANN	JOHN/ELIZABETH	C	1783	PORTSEA ST M
ANN	JOHN/ANN	C	1703	WINCHELSEA
ANN	HENRY/ELIZABETH	C	1711	PORTSEA ST M
ANN	NICHOLAS	M	1799	PORTSEA ST M
JOHN	ANN	M	1701	WINCHELSEA
JOHN	ELIZABETH	M	1776	PORTSEA ST M

Where Type is: C=Christening, M=Marriage

Figure 4. Sample of Mormon family records

A Fast Backup Utility for DFS

David Spencer reduces the tedium of backing-up discs.

Backing-up a DFS disc can be a considerable chore, especially if only a single drive is available. On systems packed with 'memory hungry' ROMs it can take nearly twenty disc swaps to backup a single side of an 80-track disc. The program given here helps to reduce this by only backing up the parts of the disc which are actually used. Many discs are only about 25% full, and therefore this will reduce to a quarter the number of swaps needed. Additionally, with less data being backed-up the process takes less time. This speed increase benefits single and dual drive users alike.

USING THE PROGRAM

Start by entering the listing below and saving it. For safety, don't save it to the disc you are going to use to try it out with - just in case you have made a mistake entering the program.

When run, the program prompts for the source and destination drives, these being specified as normal drive numbers. If the drive numbers are different, you are asked if you wish to swap discs. This allows operations such as copying the first side of one disc onto the second side of another in a single drive. If the two drive numbers are the same then you will automatically be asked to change discs as needed. The copying then starts, with disc-change prompts being displayed if needed. The number of swaps necessary depends on the amount of data on the disc, and the available memory in the computer.

If the two discs are of differing size (one 40-track, the other 80, or vice versa), then an error is generated before any data is written to the destination disc, and the backup is terminated. The other possible error is 'Disc Error' which can indicate a number of problems, including a corrupted source disc, an unformatted destination disc, or a 40-track disc in an 80-track drive.

The program works perfectly well with a 6502 second processor (or Turbo board). Indeed, the extra memory available will reduce the number of disc swaps even more, particularly if Hi-Basic is installed.

HOW IT WORKS

Very simply, the program works by identifying the areas of the disc which are used, and copying those to exactly the same places on the destination discs. On a DFS disc there are a maximum of 31 files, and each file occupies a contiguous block of disc space. Allowing for the sectors required to store the catalogue, this means that there will be at most 32 disjoint areas of used space on any disc. The program reads in the catalogue of the source disc in line 270, and then uses the procedure *PROCmakemap* to build up a list of used disc space in the form of a start sector and a length (in sectors).

The bulk of the main program works through the map of the disc contents reading in blocks into memory until all the memory is used up, or the entire disc has been read. The process is then reversed and the contents of memory written to the destination disc. This is repeated until all the relevant areas of the source disc have been copied. During the process, the program keeps a list of where all the data in memory belongs on the disc.

Lines 100 to 140 allocate the workspace. The variables *map* and *list* point to the used-space map and the buffer contents list, and their values depend on whether a second processor is in use or not. The variable *buffer* points to the transfer buffer, the length of which is stored in *size*. This is set in line 130 to use all of the free memory minus 300 bytes which are needed by Basic to store the variables and procedure stack. The procedure *PROCrw* uses the OSWORD &7F call to read and write disc sectors directly.

```
   10 REM Program Fast DFS Backup
   20 REM Version B1.00
   30 REM Author  David Spencer
   40 REM BEEBUG  June 1989
   50 REM Program subject to copyright
   60 :
  100 A%=&EA:X%=0:Y%=255:tube=USR&FFF4 A
ND &FF00
  110 IF tube THEN map=&300:list=&380 EL
SE map=&900:list=&980
```

```
  120 block=&70:IF tube=0 THEN MODE 135
ELSE MODE 128
  130 size=(HIMEM-TOP-700) AND &FF00
  140 DIM buffer size, b2 256
  150 PRINTSPC9;"BEEBUG FAST DFS BACKUP"
  160 PRINTSPC9;"----------------------"
  170 PRINT:REPEAT
  180 INPUT "Source drive " source
  190 UNTIL source>=0 AND source<=3
  200 PRINT:REPEAT
  210 INPUT "Destination drive " dest
  220 UNTIL dest>=0 AND dest<=3
  230 IF source=dest THEN swap=TRUE ELSE
 swap=FALSE
  240 IF NOT swap THEN PRINT'"Swap discs
 (Y or N) ";:key=GET AND &DF:PRINTCHR$ke
y:IF key=ASC"Y" THEN swap=TRUE
  250 PRINT:scheck=FALSE
  260 IF swap PROCprompt(FALSE)
  270 PROCrw(FALSE,buffer,source,0,2)
  280 PROCmakemap(buffer,map)
  290 ss=buffer?&107 + 256*(buffer?&106
AND 3)
  300 ptr=map
  310 REPEAT
  320 sleft=size/256:bpoint=buffer:list2
=list
  330 REPEAT
  340 !list2=!ptr AND &FFFF
  350 IF sleft>=(ptr!2 AND &FFFF) THEN l
ist2!2=ptr!2 AND &FFFF:ptr=ptr+4 ELSE li
st2!2=sleft:!ptr=!ptr+sleft:ptr!2=ptr!2-
sleft
  360 sleft=sleft-list2!2
  370 PROCrw(FALSE,bpoint,source,!list2
AND &FFFF,list2!2 AND &FFFF)
  380 bpoint=bpoint+256*list2!2:list2=li
st2+4
  390 UNTIL sleft=0 OR !ptr=-1
  400 !list2=-1
  410 IF swap PROCprompt(TRUE)
  420 IF NOT scheck THEN scheck=TRUE:PRO
Crw(FALSE,b2,dest,1,1):IF 256*(b2?6 AND
3)+b2?7<>ss THEN PRINT"Disc sizes don't
match":END
  430 bpoint=buffer:list2=list
  440 REPEAT
  450 PROCrw(TRUE,bpoint,dest,!list2 AND
 &FFFF,list2!2 AND &FFFF)
  460 bpoint=bpoint+256*(list2!2 AND &FF
FF)
  470 list2=list2+4
  480 UNTIL !list2=-1
  490 IF !ptr<>-1 AND swap THEN PROCprom
pt(FALSE)
  500 UNTIL !ptr=-1
  510 PRINT'"Backup complete"
  520 END
  530 :
 1000 DEF PROCprompt(flag)
 1010 PRINT"Insert ";
 1020 IF flag PRINT"destination"; ELSE P
RINT"source";
 1030 PRINT" disc and press Space"
 1040 *FX15 1
 1050 REPEAT UNTIL GET=32
 1060 ENDPROC
 1070 :
 1080 DEF PROCrw(flag,buff,drive,add,cou
nt)
 1090 ?block=drive:block!1=buff
 1100 block?5=3:block?6=&53+8*(flag=TRUE
)
 1110 block?7=add DIV 10
 1120 block?8=add MOD 10
 1130 REPEAT
 1140 IF count>10-block?8 THEN chunk=10-
block?8 ELSE chunk=count
 1150 block?9=chunk+&20
 1160 ret=0:REPEAT
 1170 A%=&7F:X%=block MOD 256
 1180 Y%=block DIV 256:CALL &FFF1
 1190 IF block?10 THEN ret=ret+1
 1200 UNTIL ret=6 OR block?10=0
 1210 IF block?10 PRINT"Disc Error":END
 1220 block!1=block!1+256*chunk
 1230 block?8=block?8+chunk
 1240 IF block?8>9 THEN block?8=block?8-
10:block?7=block?7+1
 1250 count=count-chunk
 1260 UNTIL count=0
 1270 ENDPROC
 1280 :
 1290 DEF PROCmakemap(cat,store)
 1300 st=0:end=2:off=cat?&105
 1310 REPEAT
 1320 IF FNstart(off)=end AND off<>0 THE
N end=end+FNlen(off) ELSE !store=st:stor
e!2=end-st:st=FNstart(off):end=st+FNlen(
off):store=store+4
 1330 off=off-8
 1340 UNTIL off<0
 1350 !store=-1
 1360 ENDPROC
 1370 :
 1380 DEF FNstart(off)
 1390 =(cat?(&106+off) AND 3)*256 + cat?
(&107+off)
 1400 :
 1410 DEF FNlen(off)
 1420 =(cat?(&106+off) AND &30)*16 + cat
?(&105+off) - (cat?(&104+off)<>0)
```

File Handling for All (Part 12)

Mike Williams and David Spencer conclude their series on file handling with some practical advice about developing and testing your own database programs.

We have come a long way in this series since we started, and you may well have found the last few articles dealing with topics more advanced than you are likely to need on most occasions. However, we hope you still found the ideas to be interesting. One factor which you may have realised is that as our file handling requirements become more sophisticated, so more storage (both in memory and on disc) is taken up with things like pointers, indexes and the like at the expense of the data itself.

While few people will want to write a full-blooded database (buying a commercial software package like BEEBUG's own Masterfile II is often simpler), there are often uses for file-handling programs, and we hope that we have given you enough encouragement and help for you to do this for yourself. At its most basic, any file-handling program needs to be able to accept data from the keyboard and store it in a file, and to read data from a file and display it on the screen (or print it out).

In this concluding article we want to look at some of the particular problems which arise when developing file-handling programs, and see what can be of help both with this and with the testing of such programs. Of course, in many ways file-handling programs are no different to any others. However, such programs often turn out to be quite lengthy, and a well planned and structured program is therefore all the more necessary.

PLANNING AND DEVELOPMENT

One of the things you need to do at the outset is to determine the format of your data files, and once done keep to it. Changing file formats part way through program development is risky and a lot of work, which only serves to underline the importance of good initial planning. Remember our earlier idea of a File Description Record to hold the details of the format of each individual file. This is very useful if you want to write a program which will handle different files with different record sizes and numbers of fields. On the other hand, if your file application is very specific, and is going to stay that way, then the use of a File Description Record may be quite unnecessary.

This question of just how flexible and open to make a program is very difficult to answer in advance. File-handling programs more than most are likely to benefit from plenty of time spent in the early planning stages, so that when you come to start writing the code you are fairly sure just what it is you want your program to do. Having said that, experience shows that it is often wise to err on the side of flexibility.

This can be desirable in other ways. When data is being written to or read from a file some form of buffer is needed to hold the contents of a record, either by using an array or by allocating part of memory and using indirection operators. Although your initial thoughts may be that your program will only ever need one file open at a time, it adds very little to the coding to allow for more than one. Any procedures to read and write records will then need to specify which of several buffers to use, but while only one file is in use only one buffer need be defined.

The question of whether one or more buffers is needed to hold records needs to be resolved at an early stage in the development, together with the definitions of two crucial procedures, those to read and write records. On the whole it seems better to restrict these to precisely that, i.e. to transfer a record from a memory buffer to a file, or from a file to a memory buffer. The display or printing of a record should be a separate procedure, as should be the input of data (from the keyboard into a buffer), or the editing of data (within a buffer).

As we saw earlier in this series, this leads to the need for four fundamental functions or procedures:

Input a record (from keyboard to memory)
Write a record (from memory to file)
Read a record (from file to memory)
Display a record (from memory)

Once these have been established you can then add further procedures to handle such tasks as modifying or deleting records. We developed procedures for all these functions much earlier in this series.

In addition, if you decide to use any form of File Description Record (and sometimes even if you don't), you will probably need a procedure to create a file before you can write anything to it anyway. It often helps as well if a file is created big enough at the outset to hold all the records you are ever like to need. This avoids any later problems should the remaining free space on the disc become fragmented or insufficient.

Like any well written program, those for file handling will benefit from the extensive use of functions and procedures to give the program structure. In many cases your program will need a number of main procedures corresponding probably to your main menu options, and a further set of utility functions or procedures, often shared by more than one main procedure. It may even help to allocate specific line number ranges for particular sections, for example having your main procedures start from line 10000 onwards and you utility functions and procedures starting from 20000 onwards. This will make it easier to locate any function or procedure, and keep the layout of the whole program more logical.

Since the essence of any file-handling program is just that (reading and writing records) it is often worthwhile not to bother too much initially about screen layouts. The important part is the file-handling itself. Once you have this up and running correctly the rest can be sorted out without too much difficulty. Developing sophisticated screen displays at too early a stage might then hinder or obscure the later discovery of any bugs in the file-handling part of your program.

TESTING AND DEBUGGING

So much for advice on the planning and writing of a database program, but a fundamental problem often arises when you want to test whether your code works correctly or not. For example, even the simplest program will need to read in some data and write it to a file, but how can you test this if you have no way of displaying the contents of the file? If you first write the code to read data from a file and display it, you have no data file to try it out on. If you write the whole lot before testing any of it and it doesn't work, you have no idea whether it is that part of the program which writes a record to the file which is failing, or the part which reads the data from the file.

```
000000 00 0B 6E 77 6F 72 42 20 ..nworB
000008 79 6D 6D 69 4A 00 0F 65 ymmiJ..e
000010 6E 61 4C 20 65 72 65 68 naL ereh
000018 77 6F 4E 20 31 32 00 0B woN 12..
000020 33 34 33 34 33 34 2D 33 343434-3
000028 34 32 30 00 0C 72 65 62 420..reb
000030 6D 65 76 6F 4E 20 68 74 mevoN ht
000038 38 00 06 65 69 62 62 65 8..eibbe
000040 44 00 18 6E 6F 64 6E 6F D..nodno
000048 4C 20 2C 74 65 65 72 74 L ,teert
000050 53 20 72 65 74 65 78 45 S retexE
000058 20 33 32 00 0B 30 31 30  32..010
000060 31 20 31 30 31 2D 31 30 1 101-10
000068 00 08 79 6C 75 4A 20 68 ..yluJ h
000070 74 35                   t5
```

Fig.1 Dump of a simple data file

This is a recurrent problem when developing any file-handling program. Ideally what you need to be able to do is to 'see' into the data file itself. Well this is possible, though you may need a little experience to interpret the results. The command:

```
*DUMP <filename>
```

will output a dump of a named file to the screen (and to a printer as well if Ctrl-B is pressed when the command is entered). The dump shows, eight bytes at a time, the values stored in each byte of the file, and where possible the ASCII character corresponding to each byte. Figure 1 shows a dump from the start of a typical file. Each row consists of eight byte values followed by the characters corresponding to the same eight bytes. Each row commences with the address (number) of the first byte in that row.

Dumps like this can be very useful in sorting out problems with file-handling programs, and so we'll deal with the use of this facility in more detail. There are two points to note. The value of each byte is given as a two digit hex value as is the address of each byte, so to use this facility you do really need to familiarise yourself with the hexadecimal (base 16) number system. The second point is that bytes which do not correspond to a printable character (codes 0 to 31 decimal) are represented by a full stop.

HEXADECIMAL NUMBERS

In this there are 16 digits (0 to 9 followed by A, B, C, D, E, and F representing the decimal numbers 10, 11, 12, 13, 14, and 15). In a two digit number as here the left-hand digit is the number of sixteens, and the right-hand digit the units. Thus &20 (2*16+0 = 32) is the code for space, while &41 (4*16+1 = 65) is the code for the letter 'A', and &5A (5*16+10=90) is the code for the letter 'Z', the '&' indicating in each case that it is a hexadecimal value). It is not too difficult to follow with some practice, and there is little alternative when interpreting file dumps.

There are further complications. If you use INPUT# and PRINT# to write to a file and to read from a file, then any data is automatically stored in a predetermined way. An integer is stored as a group of four bytes preceded by &40, a real number is stored as a five-byte floating point number preceded by &FF, while a string of characters, which may be up to 255 in length, is stored as a corresponding sequence of ASCII codes preceded by two bytes, &00 to indicate a string, and a second byte which indicates (in hex of course) the number of characters in the string. And as an added complication, the characters are stored in reverse order!

In the earlier parts of this series concerned with serial file handling (Vol.7 Nos.3 to 5), all data was stored in string format, which does make life a lot easier, with some integers being used just in the File Description Record (number of records, number of fields etc.). Thus the decimal integer '20' would be stored as:

```
  &40 &00 &00 &00 &14
```

(four bytes, most significant first), and the five characters of the string 'Smith' would be stored in reverse order as:

```
  &00 &05 &68 &74 &69 &6D &53
```

where &68 (decimal 104) is the ASCII code for 'h', &74 (decimal 116) is the ASCII code for 't', and so on. Fortunately, even backwards, character strings are not too difficult to identify in a file dump, but integers, and indeed real numbers if you choose to use them, can be more difficult.

Although it is tedious, the only way on many occasions is to print out a dump of a file and carefully check it through byte by byte, marking each integer, real or string as you identify it. You will soon realise too, that when you are first testing out a file-handling program you should keep your test data as simple as possible. For example, suppose you have written the code to create a file with a file description record and you want to test this. Create a file with, say, 10 records each consisting of two fields called Field1 and Field2, and each field eight characters in length. Figure 2 shows the dump of first 80 (hex 50) bytes of the File Description Record.

```
000000 40 00 00 00 01 40 00 00 @....@..
000008 00 0A 40 00 00 00 00 40 ..@....@
000010 00 00 00 14 40 00 00 00 ....@...
000018 02 00 06 31 64 6C 65 69 ...1dlei
000020 46 40 00 00 00 08 40 00 F@....@.
000028 00 00 00 00 06 32 64 6C .....2dl
000030 65 69 46 40 00 00 00 08 eiF@....
000038 40 00 00 00 00 00 00 00 @.......
000040 00 00 00 00 00 00 00 00 ........
000048 00 00 00 00 00 00 00 00 ........
```

Fig.2 Dump of a the first 80 bytes of a simple File Description Record showing two fields.

Starting from the beginning this shows:

&40	integer 01	number of 256 byte blocks
&40	integer 0A	size of file (10 records)
&40	integer 00	current number of records
&40	integer 14	size of record (20 bytes)
&40	integer 02	number of fields
&00	string 1dleiF	name of first field (Field1)
&40	integer 08	size of first field (8 bytes)
&40	integer 00	type of first field (type 0)
&00	string 2dleiF	name of second field (Field2)
&40	integer 08	size of second field (8 bytes)
&40	integer 00	type of second field (type 0)

You should be able to follow this through the file dump, identifying each integer or string in turn. For more information on the suggested format of the File Description Record see part three in this series (Vol.7 No.3).

Once you believe you have the file creation routine working correctly, you can move on to write and test routines to input and save data in the file, and then test this out in the same way. Just add one record (the data doesn't have to be meaningful, but it is best to be systematic), and then check this by displaying or printing a dump of the file up to and including the first record. Only when you feel sure that this is right should you move on to write the routine to read and display a record. You don't need a file dump to test this, as you have already tested that the file contents are correct. This time the proof of the routines will be obvious in the screen display.

Once you have these basic routines working (to create a file, to write keyboard input to a file, and to display records read from a file), you can in the main use these to test everything else you add to the program. However, if you decide to add pointers to records, as we described in parts 7 and 8 (Vol.7 Nos.7 & 8), you will still need to use a file dump to check that the pointers are correct, if the program fails to work correctly, as this is the only way that the pointers will ever be visible. There are also likely to be other occasions when you will need to revert to a file dump to check what is happening.

Also make sure that your testing is thorough. Does you program work with an empty file, as well as one with records? Does record deletion work correctly when the record in question is the first, or the last in the file? All these special cases do need to be checked.

When you are developing and testing file-handling programs, problems often do arise. One of the commonest is to see the error message "Type mismatch". What has happened is that the file contents have become changed (corrupted) so that what Basic detects in the file does not correspond with what it is being asked to read (and the error does always relate to a read operation). For example, the program tries to read a string of characters, but the current position in the data file does not contain the &00' initial string marker.

Re-creating the data file from scratch quickly becomes quite time consuming as development and testing proceed. For example, if you want to test that a program will correctly delete a record that is neither the first nor the last, you need to create a data file with at least three records. Having to completely re-create the file every time you re-test the delete routine takes time.

One solution, of course, is to keep a copy of your test file, and always take a copy from that for testing. There is another alternative, and that is to modify individual bytes in the data file 'by hand' as it were, to correct the error and render the file re-usable. You need to open the file, move the file pointer using PTR# to the byte to be changed (determined from the file dump), and then use BPUT# to write the correct byte to the file. This can also be useful to modify a corrupted 'live' data file.

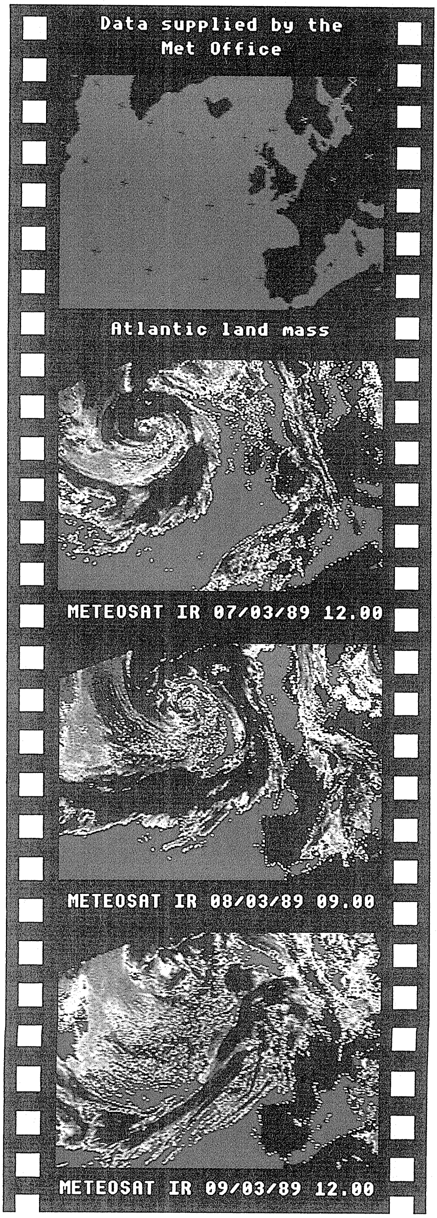
To make this technique more approachable we have parcelled it up as a short utility which is listed here. In use, it asks for the name of the file to be edited (which is checked), and then enters a loop which requests the address (number) of the byte to be changed, and the new value (both in hexadecimal) to be written in that position. The program reads and displays the current value of the byte specified (also in hex) and asks for confirmation before writing the new value in the same position. Escape can be used to terminate the loop whereupon the file is closed. Such a program can sometimes be a life-saver.

That concludes our series on file handling. We very much hope that you have found it to be both interesting and useful.

```
   10 REM Program FileByter
   20 REM Version B1.0
   30 REM Author  Mike Williams
   40 REM BEEBUG  June 1989
   50 REM Program subject to copyright
   60 :
  100 MODE3:ON ERROR GOTO 290
  110 REPEAT
  120 INPUT'"File name: " file$
  130 F%=OPENUP(file$)
  140 IF F%=0 PRINT"No such file"
  150 UNTIL F%>0
  160 REPEAT
  170 INPUT''"Enter byte number and new
value: " B$,V2$
  180 IF ASC(B$)<>38 B$="&"+B$
  190 IF ASC(V2$)<>38 V2$="&"+V2$
  200 B%=EVAL(B$):V2%=EVAL(V2$)
  210 PTR#F%=B%:V1%=BGET#F%
  220 PRINT"Existing value of byte ";B$;
" is &";STR$~(V1%)
  230 PRINT"Confirm change to ";V2$;" (Y
/N)";
  240 REPEAT:G$=CHR$(GET AND &DF):UNTIL
INSTR("YN",G$)
  250 IF G$="Y" THEN PTR#F%=B%:BPUT#F%,V
2%
  260 UNTIL FALSE
  270 END
  280 :
  290 MODE3:CLOSE#0
  300 REPORT:PRINT" at line ";ERL
  310 END
```

Weather Pictures on Your Micro

John Woodthorpe describes how the latest advances in technology can bring weather satellite pictures right into your own home, and you don't need a satellite dish to do it.



Data supplied by the Met Office
Atlantic land mass
METEOSAT IR 07/03/89 12.00
METEOSAT IR 08/03/89 09.00
METEOSAT IR 09/03/89 12.00

If you've ever been impressed by the satellite pictures on the TV weather forecasts, or tempted by the adverts for receiving dishes, hardware and software to allow you to decode the information for yourself, then BBC Telesoftware may have something to interest you. For the last few months, they have been broadcasting data and software to allow anyone with a BBC (or IBM PC compatible) computer and teletext decoder to receive and display images from Meteosat, one of the most widely used weather satellites.

If your only contact with broadcast teletext is the display on your TV, you may be surprised to learn that the page numbering system is in hex, not decimal. This allows for the existence of a greater number of pages than at first apparent, but necessitates an alphanumeric keyboard (try keying page 7C0 on your TV remote control). Until recently these pages have rarely been used, but now they are becoming packed with goodies - and all included in the price of your TV licence.

The home of Ceefax Telesoftware is BBC2, and all the software etc. mentioned here is available on that channel. The software on offer normally changes weekly, but page 7FF houses an archived suite of the weather downloading and display programs. This is accessed via a special program Utils which acts as a menu, allowing you to select the current version of a particular broadcast program, or learn of updates. Utils itself is always available for downloading from the normal Telesoftware service.

The following programs are provided:

SatLoad - allows the satellite image catalogue to be displayed and selections made for downloading.

SatShow - allows the satellite data header to be displayed.

Display - converts the downloaded data file into a mode 2 screen and saves it to disc under a filename which is a modified version of the Julian date (allowing easy sorting into date and time order).

ArcDisp - does the same on an Archimedes, but giving a much better quality image.

SatSeq - reads the saved filenames from disc and allows them to be displayed on screen in sequence.

In addition, text files give the operating manual for the software, including modifications needed for PC compatibles, and the protocol adopted in transmitting the satellite data in case you want to write your own software. The satellite image catalogue is updated at 24 hour intervals (between midnight and 10 a.m.) with the data gathered about twelve hours earlier.

The data files, currently of the North Atlantic and Western Europe as transmitted from Meteosat, are intercepted and edited by the Telesoftware people as the files pass from the Meteorological Office to the BBC weather department. The editing corrects the perspective to a more normal map projection, and simplifies the data to reduce it from 100K or more to a 20K mode 2 screen.

The data file broadcast on Ceefax needs to be downloaded, taking about ten minutes, and then converted to a screen image before it can be viewed. The conversion can also take a little while (eight minutes or so on my Master 128), but the use of a second processor reduces the time taken considerably (to about three minutes). Unfortunately, the downloading program will not work with a second processor, and so a certain amount of enabling and disabling is needed if speed is important.

The screen images can be superimposed on a plot of the land area and viewed in positive (thick cloud is white), or negative (thick cloud is dark) by modifying the display program. A monochrome monitor really is better to view the image, with the cloud thickness showing up as different grey levels. This is emphasised by some of the other images transmitted - digitised pictures of the Telesoftware team themselves - very strange in colour.

The resulting screen images can be *LOADed back in mode 2, but the sequence program SatSeq can also be used to give an interesting 'slide show' of the cloud movements over several weeks. A single ADFS disc can hold a month's worth of images. With my single drive, I keep the downloading and display software, along with a seven days of unconverted data files, on the same disc. This still leaves space for two weeks of screen images, enough to give a decent view of the cloud movements. Typical weather satellite displays are shown in the accompanying illustrations.

It would perhaps give a better indication if data files were updated more frequently, say every twelve hours or so, but the service as it stands is a very interesting development of the broadcast Teletext idea. As well as transmitting pictures of themselves, the Telesoftware team have also made available a picture of Phobos, one of the moons of Mars, taken by the Soviet Phobos 2 satellite, and there are indications that data from different weather satellites may be used to give different views of Britain, continental Europe etc., and that weather radar images may also be transmitted.

The software will work on all versions of the BBC computer (with some modifications for the model B or DFS), with any storage system supporting random access files (i.e. not cassette), and the documentation, whilst detailed, is very readable. All told, this service is very well worth taking advantage of if you have a Teletext adaptor, and if computer-driven Teletext has never appealed to you before, why not take a fresh look at it? The information on offer is frequently more up-to-date than that on Prestel, and if you have a TV licence, you're paying for it already.

Thanks are due to BBC Telesoftware for permission to publish the satellite images.

Technical Note

In order to receive broadcast teletext, you will need a teletext adaptor for your micro. One that is particularly recommended is the GIS Advanced Teletext Receiver which was reviewed in BEEBUG Vol.7 No.2. This normally costs £149 inc. VAT from GIS Ltd, Croxton Park, Croxton, Cambridge PE19 4SY, tel. (0480) 87464. It is also available to BEEBUG members at 5% discount (£141.55) plus £3 p&p direct from BEEBUG Retail.

Please note that this adaptor will only work with the model B and Master 128 because of the connections required, and may not be used with a Compact. Once installed, a lead from your TV aerial is plugged into the adaptor which can then be tuned to pick up the teletext broadcasts on all four channels.

An Extended Disassembler

Delve into machine code programs with this utility from Kevin Harding.

A disassembler is a utility which takes a machine code program from memory and converts it back into the 6502 instruction mnemonics, thereby making it much easier to understand than the actual machine code. Essentially, the disassembler is performing the opposite function to Basic's assembler. There are a number of uses for a disassembler, and these include delving into the system ROMs of the computer, and examining other machine code programs for which you don't have the assembler listing. This latter function can be very useful for beginners to machine code programming who want to look at other people's work. The disassembler given here is in the form of a sideways ROM image, and will work on any BBC computer with sideways RAM fitted.

ENTERING THE PROGRAM

Because of the length of the disassembler (over 1400 lines of Basic), the program is presented here as a hex dump. However, don't let this put you off typing in the program, because the short loader that is used to enter the hexdump is designed to make the entry error proof. Alternatively, this month's magazine disc contains the full source code (and this can also be supplied as a printed listing - see later).

Start by entering the loader program from listing 1 and saving this to disc. Next, delete (or rename) any file called 'DISASS'. This is because the loader allows you to enter the dump in sections, and detects where to start by looking at the file 'DISASS'. Having done this, run the loader and start entering line by line the data from listing two. When entering each line, you should omit the address and the colon - these are displayed on the screen for you. You can also omit the spaces, as these are only to aid readability, although no harm will result if they are left in. Additionally, leave out the blank lines. As each line is entered, the loader checks to make sure that the line is the correct length, and that the built-in checksum matches. If either fails, you will be asked to re-enter the line. This ensures that it is almost impossible to make a mistake during entry.

If you get bored while entering the dump, simply exit by pressing Escape. When you re-run the loader it will pick up at the point where you left off. Do though, make sure that you don't delete the 'DISASS' file once you have started entering the dump. When entry is complete, the loader will exit, and the file 'DISASS' will contain the complete ROM image for the disassembler. This should then be loaded into sideways RAM. On a Master or Compact, this can be done using:

```
    *SRLOAD DISASS 8000 WQ
```

On a model B you should use your normal command for loading ROM images. In either case, press Ctrl-Break to initialise the ROM.

USING THE DISASSEMBLER

The disassembler is controlled by just one star command. This takes the form:

```
    *DIS <start> <end> [C] [R] [P]
```

where start is the address of the first instruction to be disassembled, and end is the address of the last instruction. Both of these numbers should be given in hex without any '&'. The optional qualifying letters C and R control the exact range of opcodes handled by the disassembler. If neither letter is present then only opcodes available on the original 6502 (as fitted to the model B) will be disassembled. Adding the letter 'C' will also disassemble the extra opcodes offered by the 65C02 fitted to the Master 128 and Compact. If an 'R' is added then all the opcodes offered by the Rockwell version of the 65C02 are disassembled. This processor was fitted to many 6502 second processors and Master Turbo boards, although not to all of them. It should be noted that adding an 'R' includes the instructions covered by adding 'C', and therefore only one of the two is needed. Adding a 'P' will cause the disassembled output to be printed as well as displayed on the screen.

If any of the range of addresses selected falls within the sideways ROM area (&8000 to &BFFF), then the program will prompt for a ROM number to be entered. You can either specify a number in the range 0 to 15, or press Return in which case the ROM socket containing the disassembler will be disassembled.

The disassembler's display is always in mode 7, and unless the 'P' option is selected will be in page mode. The display consists of five fields. At the left is the address of the instruction in white. This is followed by the hex codes for the bytes making up the instructions (in green), and their ASCII equivalents in cyan. Next comes the disassembled instruction itself. This is shown in yellow for 6502 instructions, blue for 65C02 instructions, or red for Rockwell 65C02 ones. Remember that these latter two are only displayed if the 'C' or 'R' options are specified. Otherwise, they are treated as undefined instructions and displayed as '???'. Finally, if the instruction refers to an operating system routine or a vector address, then the name of that routine/vector is displayed in magenta after the instruction. When the output is printed, 65C02 and Rockwell instructions are identified by '*C*' and '*R*' respectively before the instruction mnemonic.

THE ROCKWELL 6502 INSTRUCTION SET

BEEBUG's Exploring Assembler series which ran from Vol.6 No.2 to Vol.7 No.2 covered the instruction set of the standard 6502, and that of the 65C02, in great detail. However, the extra instructions offered by the Rockwell chip were not dealt with.

There are four extra instructions available on the Rockwell chip. The first two take the form:

```
BBR bit,zpage,address    and
BBS bit,zpage,address
```

These test the specified bit of the given zero page memory location and if it is set for BBS, or reset for BBR, then branch to the given address. For example:

```
BBS 4,&70,&980
```

These instructions take three bytes in memory. The first is the opcode, and has the value:

```
&0F + bit*16     for BBR, and
&8F + bit*16     for BBS.
```

The second byte is the zero page address, and the third byte is the address offset relative to the start of the next instruction.

The other two instructions are used to set and reset individual bits in a zero page location. These take the form of:

```
RMB bit,zpage    and
SMB bit,zpage
```

For example:

```
SMB 4,&80
```

will set bit four of location &80. Each instruction takes up two bytes. The first is the opcode, the value being:

```
&07 + bit*16     for RMB, and
&87 + bit*16     for SMB
```

The second byte is the zero page address.

It should be noted that the Basic assembler does not recognise these opcodes, and they can only be assembled by hand using EQUB.

```
   10 REM Hex Dump Loader
   20 REM By David Spencer
   30 :
   40 PROCassemble
   50 ON ERROR GOTO 340
   60 *KEY10 CLOSE#X%|M*KEY 10|M
   70 READ name$,st%,end%
   80 DIM B%(9)
   90 X%=OPENUP name$:IF X%=0 X%=OPENOUT
 name$
  100 PTR#X%=EXT#X%
  110 S%=(EXT#X%+st%) AND &FFF8
  120 REPEAT:REPEAT:REPEAT
  130 PRINT;~S%;":";
  140 INPUT "" H$
  150 L$="":FOR F%=1 TO LEN H$
  160 IF INSTR("0123456789ABCDEF",MID$(H
$,F%,1)) THEN L$=L$+MID$(H$,F%,1)
  170 NEXT
  180 IF LEN L$<>20 VDU7:PRINT"Wrong len
gth - Repeat Line"
  190 UNTIL LEN L$=20
  200 !&70=0
  210 PROCcrc(S% MOD 256):PROCcrc(S% DIV
 256)
  220 FOR E%=0 TO 9
  230 B%=EVAL("&"+MID$(L$,E%*2+1,2))
  240 PROCcrc(B%):B%(E%)=B%
  250 NEXT
  260 IF (!&70 AND &FFFF) VDU 7:PRINT"Ch
ecksum error - Repeat line"
  270 UNTIL (!&70 AND &FFFF)=0
  280 FOR F%=0 TO 7:BPUT #X%,B%(F%):NEXT
  290 S%=S%+8
  300 UNTIL S%=end%
  310 CLOSE #X%:*KEY 10
  320 END
  330 :
  340 IF ERR=17 THEN CLOSE #X%:PRINT:OSC
LI "KEY 10":END
  350 REPORT:PRINT" at line ";ERL
  360 END
  370 :
  380 DEF PROCcrc(byte)
  390 ?&72=byte:CALL &900
  400 ENDPROC
  410 :
  420 DEF PROCassemble
  430 FOR F%=0 TO 2 STEP 2
  440 P%=&900
  450 [OPT F%
  460 LDA &71:EOR &72:STA &71:LDX #8
  470 .loop LDA &71:ROL A:BCC noc
  480 LDA &71:EOR #8:STA &71
  490 LDA &70:EOR #&10:STA &70
  500 .noc ROL &70:ROL &71:DEX:BNE loop
  510 RTS
  520 ]NEXT
  530 ENDPROC
  540 :
  550 DATA "DISASS",&8000,&8CD8
```

```
8000:0000 004C 2680 821A C0FD
8008:0144 4953 4153 5345 2B29
8010:4D42 4C45 5200 312E 0E08
8018:3031 0028 4329 2042 4FB1
8020:4545 4255 4700 C904 9856
8028:D003 4CC2 80C9 09F0 0482
8030:0160 4898 488A 4888 2634
8038:C8B1 F2C9 20F0 F9C9 B3E4
8040:0DD0 1F20 A480 A000 0B32
8048:B959 80F0 0620 E3FF B5D4
8050:C8D0 F520 E7FF 4C9E E7A0
8058:8020 2044 4953 434F 56A5
8060:4D00 A200 B1F2 20F5 8D72
8068:89C9 0DF0 18C9 20F0 F951
8070:14DD 5B80 D004 C8E8 FF70
8078:D0EA C92E D004 E002 82CD

8080:B008 4C9E 80BD 5B80 D9FC
8088:D0F8 20A4 80A0 00B9 5B21
8090:7781 F007 20E3 FFC8 C860
8098:4C8F 8020 E7FF 68AA 133F
80A0:68A8 6860 20E7 FFA0 4837
80A8:08C8 B900 80F0 0620 D042
80B0:E3FF 4CA9 80A9 2020 D47A
80B8:E3FF CC07 80D0 EA4C BDF1
80C0:E7FF 4898 488A 4898 E929
80C8:1865 F285 A8A5 F369 13BB
80D0:0085 A9A2 00A0 FFC8 4E5E
80D8:B1A8 C92A F0F9 BD02 4F38
80E0:81F0 23B1 A820 F589 FF3A
80E8:DD02 81D0 05E8 C84C 5FBA
80F0:DE80 B1A8 C92E D004 99F8
80F8:E001 B00A 68AA 68A8 AB3B

8100:6860 4449 5300 B1A8 5675
8108:C90D F066 C920 F004 763D
8110:C92E D0E8 6868 68A2 AA44
8118:FE8E 8403 A200 C8B1 D3EB
8120:A8C9 20F0 F9B1 A8C9 DB6F
8128:0DF0 10C9 20F0 17C9 D9F6
8130:2CF0 139D 8B03 E8C8 34F1
8138:4C25 81AD 8403 C9FF 254C
8140:D030 E001 902C CA98 A69C
8148:4820 E088 68A8 B058 4C25
8150:AD85 03EE 8403 F00A 7698
8158:85AA AD86 0385 AB4C 38AC
8160:1C81 A5AB CD86 0390 919D
8168:4CA5 AACD 8503 9045 434D
8170:F043 2000 8A00 DC53 79B4
8178:796E 7461 783A 0D2A 6FDA

8180:4449 5320 3C53 7461 D5A7
8188:7274 2041 6464 723E 63FC
8190:203C 456E 6420 4164 ACE2
8198:6472 3E20 2843 2920 C1D8
81A0:2852 2920 2850 2900 430D
81A8:2000 8A00 A142 6164 119F
81B0:2048 6578 0038 AD85 EDAE
81B8:03E5 AA8D 8003 AD86 84C0
81C0:03E5 AB8D 8103 90AA B439
81C8:EE80 03D0 0DEE 8103 AE5B
81D0:D008 A9FF 8D80 038D CA96
81D8:8103 A5AB 20B4 89B0 09AA
81E0:1BAD 8603 20B4 89B0 9B15
81E8:13A5 ABC9 8090 034C 38C4
81F0:C782 AD86 03C9 C0B0 BA7C
81F8:034C C782 8C94 03A0 A4F5

8200:00B9 0D82 F03D 20E3 5B6F
8208:FFC8 4C01 8245 6E74 59F0
8210:6572 2052 6F6D 204E 5394
8218:756D 6265 720D 2830 BC29
8220:2D31 3529 206F 7220 8A30
8228:3C52 4554 5552 4E3E 6227
8230:2066 6F72 2061 6374 8578
8238:6976 6520 524F 4D20 8D60
8240:3A20 00A9 0FA2 0020 13F1
8248:F4FF A200 20E0 FF90 DB44
8250:11A9 7E20 F4FF 2000 5A4F
8258:8A00 1145 7363 6170 89EA
8260:6500 C90D F01D C97F 9DA7
8268:D00B CA30 DDA9 7F20 BF12
8270:E3FF 4C4C 829D 8B03 BDE0
8278:20E3 FFE8 E003 B0EA A55B

8280:4C4C 82E0 00D0 05A9 2FCF
8288:FF4C BE82 CABD 8B03 419E
8290:C930 9024 C93A B020 356C
8298:38E9 308D 9603 E000 ACD7
82A0:F01C CABD 8B03 C930 BBEF
82A8:F014 C931 D00A 18AD 84A5
82B0:9603 690A C910 9006 82CF
82B8:20E7 FF20 FF81 8D96 9587
82C0:03AC 9403 20E7 FFA9 A3EE
82C8:1620 E3FF A907 20E3 3E74
82D0:FFA9 0E20 E3FF A900 FE45
82D8:8D99 038D 8403 8D85 02B7
82E0:0388 C8B1 A8C9 0DF0 5F5A
82E8:3829 DFC9 43D0 04A9 0321
82F0:8030 06C9 52D0 0BA9 F5B7
82F8:C00D 8403 8D84 034C 0A1C

8300:E282 C950 D0DC 8D85 518C
8308:03A9 0F20 E3FF 8C94 7E21
8310:03A2 0020 F4FF A902 7E36
8318:20E3 FFAC 9403 4CE2 C221
8320:82A0 008C 8F03 2085 2265
8328:898D 8603 D930 8AF0 3645
8330:0BC8 C8C8 C8C0 88D0 C741
8338:F34C 5D83 A2FD C8B9 829E
8340:308A 9D8A 02E8 D0F6 6418
8348:C023 900C F005 A90B 303D
8350:4CCC 85A9 034C CC85 C35E
8358:A90A 4CCC 8529 F7C9 70E6
8360:24D0 19A9 428D 8703 7467
8368:A949 8D88 03A9 548D 3A7B
8370:8903 AD86 034A 4A29 67BF
8378:074C CC85 AD86 0329 41E5

8380:DFC9 4CD0 3AA9 4A8D 5740
8388:8703 A94D 8D88 03A9 409F
8390:508D 8903 AD86 03C9 5DEF
8398:4CD0 05A9 034C CC85 AE0B
83A0:A203 8E8B 0320 EC87 AD04
83A8:A920 20E3 FFA9 2820 97C7
83B0:E3FF 209B 88A9 2920 DAF3
83B8:E3FF A201 4C64 87A0 1DF6
83C0:00AD 8603 29E7 D9B8 F43A
83C8:8AD0 428C 9003 8D91 4A38
83D0:03AD 8603 C014 9003 C4A2
83D8:2918 4A4A 4A29 078D 20B4
83E0:8E03 AA98 4A4A A8B9 BB34
83E8:088B DD1C 8BD0 09AC 8459
83F0:9003 AD91 034C 0D84 C604
83F8:AC90 03A2 FDC8 B9B8 E308

8400:8A9D 8A02 E8D0 F6AD 9614
8408:8E03 4CCC 85C8 C8C8 22F6
8410:C8C0 18D0 B1A0 00AD AACF
8418:8603 29E3 D9D0 8AD0 F1B2
8420:4E8C 9003 8D91 03AD 31D6
8428:8603 4A4A 2907 8D8E 122C
8430:03AA 984A 4AA8 B90E B383
8438:8BDD 1C8B D009 AC90 76B7
8440:03AD 9103 4C6F 84AC E62F
8448:9003 A2FD C8B9 D08A 0413
8450:9D8A 02E8 D0F6 AD8E 5BC3
8458:03C0 20B0 0FC9 00D0 8D39
8460:05A9 084C CC85 C902 F984
8468:D002 A900 4CCC 85C8 13DC
8470:C8C8 C8C0 38D0 A52C 47FE
8478:8403 3002 102D A200 53DA
```

```
8480:A000 B94B 8CF0 24CD BCC2
8488:8603 F008 C8C8 C8C8 C5F3
8490:C84C 8284 C8B9 4B8C EA7E
8498:9D87 03C8 E8E0 03D0 FD1B
84A0:F4B9 4B8C E88E 8F03 7FC2
84A8:4CCC 852C 8403 7003 152B
84B0:4C59 85AD 8603 290F 3E91
84B8:C907 F057 C90F F003 C7BF
84C0:4C59 85A9 428D 8703 CEE5
84C8:8D88 03A9 538D 8903 26FE
84D0:AD86 03C9 80B0 05A9 5661
84D8:528D 8903 A201 8E8F 7451
84E0:03E8 E88E 8B03 20EC 0474
84E8:87A9 2020 E3FF AD86 65EA
84F0:0329 704A 4A4A 4A18 2A98
84F8:6930 20E3 FFA9 2C20 300E

8500:E3FF 20B6 88A9 2C20 1FEF
8508:E3FF A002 2085 89AA 68E7
8510:4C18 87A9 538D 8703 A4A0
8518:A94D 8D88 03A9 428D D761
8520:8903 AD86 03C9 80B0 52BE
8528:05A9 528D 8703 A201 5B43
8530:8E8F 03E8 8E8B 0320 155C
8538:EC87 A920 20E3 FFAD F474
8540:8603 2970 4A4A 4A4A A812
8548:1869 3020 E3FF A92C FEF2
8550:20E3 FF20 B688 4C6C 68C8
8558:85A9 3F8D 8703 8D88 7E61
8560:038D 8903 A201 8E8B 50A5
8568:0320 EC87 20E7 FFAD DF29
8570:8103 D010 AD8B 03CD 9028
8578:8003 9008 A903 20E3 3FC4

8580:FFA9 0060 A5AA 6D8B 3EE1
8588:0385 AA90 02E6 AB38 B4AC
8590:AD80 03ED 8B03 8D80 7105
8598:03B0 03CE 8103 2C99 59A1
85A0:0310 05A0 0620 C089 3486
85A8:A981 A200 A000 20F4 BA17
85B0:FF90 16C0 1BD0 12A9 0C3F
85B8:7E20 F4FF A977 20F4 AB27
85C0:FFA9 0320 E3FF A900 953B
85C8:604C 2183 C900 D013 DD28
85D0:A202 8E8B 0320 EC87 906C
85D8:A923 20E3 FF20 B688 6081
85E0:4C6C 85C9 01D0 13A2 BD06
85E8:028E 8B03 20EC 87A9 8643
85F0:2020 E3FF 20B6 884C 994A
85F8:6C85 C902 D015 A201 2E2D
```

```
8600:8E8B 0320 EC87 A920 56BE
8608:20E3 FFA9 4120 E3FF 0720
8610:4C6C 85C9 03D0 15A2 EC39
8618:038E 8B03 20EC 87A9 D761
8620:2020 E3FF 209B 88A2 6EC1
8628:034C 6487 C904 D020 6065
8630:A202 8E8B 0320 EC87 120B
8638:A920 20E3 FFA9 2820 90A9
8640:E3FF 20B6 88A9 2920 BF44
8648:E3FF 20D5 884C 6C85 5EB6
8650:C905 D025 A202 A202 953E
8658:8E8B 0320 EC87 A920 FA79
8660:20EE FF20 B688 AD89 8C15
8668:03C9 58F0 0620 CA88 7B0A
8670:4C6C 8520 D588 4C6C C00E
8678:85C9 06D0 16A2 038E E8C2

8680:8B03 20EC 87A9 2020 FA11
8688:E3FF 209B 8820 D588 8D6A
8690:4C64 87C9 07D0 25A2 DF84
8698:038E 8B03 20EC 87A9 3355
86A0:2020 E3FF 209B 88AD 7B1A
86A8:8903 C958 F008 20CA A783
86B0:88A2 014C 6487 20D5 8516
86B8:884C 6487 C908 D020 4E88
86C0:A202 8E8B 0320 EC87 29BE
86C8:A920 20E3 FFA9 2820 AB1C
86D0:E3FF 20B6 8820 CA88 1B2E
86D8:A929 20E3 FF4C 6C85 7BC0
86E0:C909 D01D A202 8E8B C313
86E8:0320 EC87 A920 20E3 3762
86F0:FFA9 2820 E3FF 20B6 8553
86F8:88A9 2920 E3FF 4C6C D9DC

8700:85C9 0AD0 54A2 028E 2D0D
8708:8B03 20EC 87A9 2020 BF4D
8710:E3FF A001 2085 89AA 9E8F
8718:18A5 AA6D 8B03 8D8C B739
8720:03A5 AB69 008D 8D03 F44F
8728:8A18 6D8C 038D 8C03 8195
8730:088A 2980 F009 28B0 939D
8738:0CCE 8D03 4C45 8728 38C4
8740:9003 EE8D 03A9 2620 5C76
8748:E3FF AD8D 0320 7588 7D82
8750:AD8C 0320 7588 4C6C 8E9E
8758:85A2 018E 8B03 20EC BAE6
8760:874C 6C85 CAF0 08A9 305B
8768:2020 E3FF 4C64 87A9 C565
8770:8520 3489 A200 A001 A391
8778:2085 89DD 248B D019 1894
```

```
8780:C820 8589 DD25 8BD0 7C71
8788:10A0 FAE8 E8BD 248B 2104
8790:20E3 FFC8 D0F6 4C6C BCE2
8798:858A 1869 08AA E0A0 25D8
87A0:D0D4 A002 2085 89C9 CE47
87A8:02F0 034C 6C85 8820 1DEF
87B0:8589 C936 B0F5 4A8D 1E54
87B8:9103 0A0A 186D 9103 CED2
87C0:A8A2 FBB9 C48B C920 4BE4
87C8:F007 20E3 FFC8 E8D0 F0F0
87D0:F2A9 2D20 E3FF A001 8EBC
87D8:2085 8929 01D0 08A9 6D9C
87E0:4C20 E3FF 4C6C 85A9 38D6
87E8:484C E187 8C90 03A5 5D4F
87F0:AB20 7588 A5AA 2075 29C6
87F8:88A9 8220 3489 8E92 FD14

8800:03A0 0020 8589 2075 7FE1
8808:88A9 2020 E3FF C8CA DB8C
8810:D0F1 C003 F00D 20E3 E592
8818:FF20 E3FF 20E3 FFC8 38CA
8820:4C12 88A9 8620 3489 F152
8828:AE92 03A0 0020 8589 871C
8830:2063 88C8 CAD0 F6C0 1D61
8838:04F0 09A9 2020 E3FF 710B
8840:C84C 3788 A983 AC8F FA14
8848:03F0 06A9 8018 6D8F 6CE3
8850:0320 3489 BD87 0320 726F
8858:E3FF E8E0 03D0 F5AC FCDB
8860:9003 60C9 2090 08C9 8F8A
8868:7FB0 0420 E3FF 60A9 FD02
8870:2E20 E3FF 608E 9303 D966
8878:AA4A 4A4A 4A20 8A88 E546

8880:8A29 0F20 8A88 AE93 3CCA
8888:0360 C90A 9007 1869 C77B
8890:3720 E3FF 6069 3020 6B75
8898:E3FF 60A9 2620 E3FF A199
88A0:8C90 03A0 0220 8589 5026
88A8:2075 8888 2085 8920 1C88
88B0:7588 AC90 0360 A926 30B0
88B8:20E3 FF8C 9003 A001 4174
88C0:2085 8920 7588 AC90 8A42
88C8:0360 A92C 20E3 FFA9 F201
88D0:5820 E3FF 60A9 2C20 2DFC
88D8:E3FF A959 20E3 FF60 7671
88E0:A900 8D85 038D 8603 1EE0
88E8:8AA8 A200 B98B 0320 9B2E
88F0:1989 B024 8D8A 0388 6E98
88F8:1005 A900 4C08 89B9 1597
```

```
8900:8B03 2019 89B0 1188 E79E
8908:0A0A 0A0A 0D8A 039D 57E1
8910:8503 E8C0 FFD0 D518 665B
8918:60C9 3090 15C9 47B0 56D0
8920:11C9 41B0 09C9 3AB0 64C7
8928:0938 E930 1860 E937 BBA4
8930:1860 3860 8D97 038C 67D3
8938:9403 2C99 0310 05A0 AFD9
8940:0620 C089 A903 20E3 5E2F
8948:FFAD 9703 20E3 FFAD BFD5
8950:8503 D004 AC94 0360 E3A5
8958:A902 20E3 FFA9 0120 F181
8960:E3FF A920 20E3 FFAD 23AD
8968:9703 C983 D005 A000 C7C0
8970:4CC0 89C9 84D0 05A0 0CD9
8978:024C C089 C981 D0D4 0E7E

8980:A004 4CC0 8998 1865 C530
8988:AA85 F6A5 AB69 0085 330A
8990:F78C 9403 C980 9016 4989
8998:C9C0 B012 AC96 0330 D462
89A0:0D8E 9503 20B9 FFAC 2C8C
89A8:9403 AE95 0360 AC94 0209
89B0:03B1 AA60 C980 9006 CBF3
89B8:C9C0 B002 3860 1860 31D5
89C0:C006 D004 A900 F002 4604
89C8:A9FF 8D99 03B9 188A 8EA5
89D0:85A8 C8B9 188A 85A9 0B2A
89D8:A000 B1A8 8D98 03F0 AA5B
89E0:10C8 A901 20E3 FFB1 73F0
89E8:A820 E3FF CC98 0390 8A50
89F0:F0AC 9403 60C9 6190 E0F3
89F8:06C9 7BB0 0229 DF60 E502

8A00:6885 A868 85A9 A900 031B
8A08:8D00 01A0 01C8 B1A8 3E7C
8A10:99FF 00D0 F84C 0001 AD70
8A18:208A 258A 2A8A 2F8A 3A3B
8A20:0420 2020 2004 2A43 4F04
8A28:2A20 042A 522A 2000 3087
8A30:9042 4343 B042 4353 6BD4
8A38:F042 4551 D042 4E45 3C66
8A40:1042 504C 3042 4D49 B505
8A48:5042 5643 7042 5653 E3E8
8A50:204A 5352 0042 524B EE16
8A58:1843 4C43 D843 4C44 6D78
8A60:5843 4C49 B843 4C56 5177
8A68:CA44 4558 8844 4559 490C
8A70:E849 4E58 C849 4E59 BA80
8A78:EA4E 4F50 4850 4841 18F1
```

```
8A80:0850 4850 6850 4C41 A0A8
8A88:2850 4C50 4052 5449 4015
8A90:6052 5453 3853 4543 8EBB
8A98:F853 4544 7853 4549 5102
8AA0:AA54 4158 A854 4159 91E2
8AA8:BA54 5358 9A54 5853 AEF9
8AB0:8A54 5841 9854 5941 B556
8AB8:C644 4543 E649 4E43 64FE
8AC0:8653 5458 8453 5459 F181
8AC8:E043 5058 C043 5059 ACA9
8AD0:6141 4443 2141 4E44 239B
8AD8:C143 4D50 4145 4F52 E964
8AE0:A14C 4441 014F 5241 112C
8AE8:E153 4243 8153 5441 98DB
8AF0:0241 534C A24C 4458 D0A4
8AF8:A04C 4459 424C 5352 3326

8B00:2252 4F4C 6252 4F52 13B2
8B08:AAAA 2A2A 0B0B FFFF DE67
8B10:FFFF FFFF FFFB AEAB 7241
8B18:ABAE AEAE 0102 0408 134A
8B20:1020 4080 B9FF 4F53 2327
8B28:5244 524D BCFF 7664 A43C
8B30:7563 686F BFFF 4F53 36FF
8B38:4556 454E C2FF 4753 31B6
8B40:494E 4954 C5FF 4753 0927
8B48:5245 4144 C8FF 4E56 39FB
8B50:5244 4348 CBFF 4E56 C4F0
8B58:5752 4348 CEFF 4F53 8F8A
8B60:4649 4E44 D1FF 4F53 6448
8B68:4742 5042 D4FF 4F53 C918
8B70:4250 5554 D7FF 4F53 19E1
8B78:4247 4554 DAFF 4F53 5E97

8B80:4152 4753 DDFF 4F53 1479
8B88:4649 4C45 E0FF 4F53 FA6C
8B90:5244 4348 E3FF 4F53 83C7
8B98:4153 4349 E7FF 4F53 72D2
8BA0:4E45 574C EEFF 4F53 169B
8BA8:5752 4348 F1FF 4F53 4C38
8BB0:574F 5244 F4FF 4F53 27DE
8BB8:4259 5445 F7FF 4F53 32AD
8BC0:434C 4920 5553 4552 F02D
8BC8:5642 524B 5620 4952 8E08
8BD0:5131 5649 5251 3256 38E2
8BD8:434C 4956 2042 5954 5248
8BE0:4556 574F 5244 5657 FA87
8BE8:5243 4856 5244 4348 6845
8BF0:5646 494C 4556 4152 7596
8BF8:4753 5642 4745 5456 5010
```

```
8C00:4250 5554 5647 4250 CD26
8C08:4256 4649 4E44 5646 AD78
8C10:5343 5620 4556 4E54 678F
8C18:5655 5054 5620 4E45 0286
8C20:5456 2056 4455 5620 C083
8C28:4B45 5956 2049 4E53 ED36
8C30:5620 5245 4D56 2043 8EBE
8C38:4E50 5620 494E 4431 E6D8
8C40:5649 4E44 3256 494E 845B
8C48:4433 5680 4252 410A B909
8C50:6453 545A 0174 5354 4AA6
8C58:5A05 9C53 545A 039E AA9C
8C60:5354 5A07 0454 5342 8322
8C68:010C 5453 4203 1454 9CF1
8C70:5242 011C 5452 4203 3296
8C78:1A49 4E43 023A 4445 E6EC

8C80:4302 3442 4954 053C DBD9
8C88:4249 5407 8942 4954 CACD
8C90:007C 4A4D 5008 5A50 1711
8C98:4859 0B7A 504C 590B 992B
8CA0:DA50 4858 0BFA 504C 57A3
8CA8:580B 124F 5241 0932 F853
8CB0:414E 4409 5245 4F52 D0EC
8CB8:0972 4144 4309 9253 7CDD
8CC0:5441 09B2 4C44 4109 5040
8CC8:D243 4D50 09F2 5342 16A4
8CD0:4309 0000 0000 0000 1D73
```

If you want the original source code for disassembler then the best solution is to buy this month's magazine disc, which also contains a version designed to run from main memory. Alternatively, you can obtain a photocopy of the listing by sending £2 to the address given below:

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

Please mark your envelope 'Disassembler Listing', and do not include any other orders, queries, contributions etc. in the same envelope.

1st course

Investigating Teletext Mode (3)

Mike Williams forays deeper into the potential of teletext mode displays.

I propose this month to investigate a little further some of the ideas which I described last time in building up techniques for large digit displays. I shall also take a more detailed look at the control and use of colour in mode 7.

The basic principle which we used for large digit displays was to think of each digit as being placed in a window. A digit can then be displayed by just pouring all the necessary characters into the window. We don't have to worry about things like Carriage Return and Linefeed because the boundaries of the window do it all for us. All the characters for any one digit were read from a set of DATA statements each time the digit was to be re-displayed.

Let's look at that particular point first. Re-reading the characters every time does seem a bit time consuming, but that is easily overcome by defining an array - let's call it digit$() - and then assigning to each element a string containing all the characters for a digit. We will do this so that the characters which make up a zero are assigned as a string to digit$(0), those for the digit '1' to digit$(1) and so on.

Thus the start of any program (which is using large digits) will need to call a procedure which is something like this:

```
1000DEF PROCset_up_digits
1010 LOCAL C$,C%,C1%,D%
1020 DIM digit$(11)
1030 FOR D%=0 TO 11
1040 C$=""
1040 FOR C%=1 TO 35
1050 READ C1%:C$=C$+CHR$(C1%)
1060 NEXT C%
1070 digit$(D%)=C$
1080 NEXT D%
1090 ENDPROC
9000 DATA 32,32,255,32,32
9010 DATA 32,255,32,255,32
9020 DATA 255,32,32,32,255
9030 DATA 255,32,32,32,255
9040 DATA 255,32,32,32,255
9050 DATA 32,255,32,255,32
9060 DATA 32,32,255,32,32
```

I have only shown the data for the digit zero. You would also need to include the DATA statements defining the other digits, and any other symbols like decimal point and minus sign which you might wish to represent (see last month's article for more details).

We can now write a new procedure to display a given digit at a position (x,y) on the screen. The coding will be:

```
1200 DEF PROCdigit(x,y,digit$)
1210 LOCAL n
1220 n=INSTR("0123456789.- ",digit$)
1230 IF n=0 ENDPROC
1240 VDU28,x,y+7,x+4,y
1250 IF n=13 CLS:ENDPROC
1260 PRINT digit$(n-1);
1270 VDU26:ENDPROC
```

This follows very much the similarly named procedure given last month, but the display of the digit involves just the single instruction in line 1260 (note the use of n-1 to compensate for the fact that the earlier INSTR instruction returns a '1' for zero, a '2' for one, and so on). If you were certain that the routine would be strictly confined to digits it could be even simpler still:

```
1200 DEF PROCdigit(x,y,digit)
1210 VDU28,x,y+7,x+4,y
1220 PRINT digit$(digit);
1230 VDU26:ENDPROC
```

The digit is now specified as a number, and not as a character. The procedure simply defines the window and places the appropriate character string in it.

The one chore which remains with the above method is the need to include 35 different numbers in DATA statements for each digit or other character which is to be defined. However, even this can be reduced without great difficulty. Each character or digit in our examples is being defined as 7 rows of 5 characters each. But since only two characters are permissible (codes 32 and 255), an entire row could be represented by a single binary number.

Thus in defining the figure zero (given before) the first row could be represented as 00100, the second as 01010, and so on. Thus the data for that digit becomes:

```
   DATA 00100,01010,10001,10001
   DATA 10001,01010,00100
```

That's just five numbers per digit instead of the original 35. Of course, we have to re-write our original procedure to decode these binary values. In fact, the simplest way to treat them is probably as strings, and then to extract each binary 'character' in turn and convert it either into the character for space (when 0) or the character 255 (when 1) as follows:

```
   1000DEF PROCset_up_digits
   1010 LOCAL C$,C%,C1%,C1$,C2$,D%
   1020 DIM digit$(12)
   1030 FOR D%=0 TO 11
   1040 C$=""
   1040 FOR C%=1 TO 7
   1050 READ C1$
   1060 FOR C1%=1 TO 5
   1070 C2$=MID$(C1$,C1%,1)
   1080 IF C2$="0" THEN C$=C$+CHR$32 ELSE
C$=C$+CHR$255
   1090 NEXT C1%
   1110 NEXT C%
   1120 digit$(D%)=C$
   1130 NEXT D%
   1140 ENDPROC
   9000 DATA 00100,01010,10001,10001
   9010 DATA 10001,01010,00100
```

The procedure now contains an extra inner loop from line 1060 to line 1090. This scans the five binary digits of each number, and builds up a string of characters with codes 32 (space) or 255. Undoubtedly the coding is more complex, but we now need to type in just 7 binary numbers per digit, nor 35 as before. And this method works just as well with any other character we choose to define on a seven by five grid.

Well that's enough I am sure on techniques for defining and using large digits, although we haven't exhausted all the possibilities by any means. What I want to deal with now is the use of colour, and I shall be using our large digits display as an example later.

CONTROLLING THE USE OF COLOUR

In the teletext mode (mode 7), colours are specified by special control characters. Because they are characters, they occupy a character position on the screen display, but because their function is purely that of control, they are invisible, appearing just as spaces. Nevertheless, it is very important to take into account the positions of these control characters when planning mode 7 screen displays, and you will sometimes find them a constraint on what you able to do.

There are three separate colours which you can control, the text colour, the graphics colour and the background colour. If you look in the back of the User Guide you will find tables showing you all the mode 7 characters (together with their corresponding codes). Now you may have been puzzled as to why there are two separate tables for mode 7 and yet just one for the other modes. The reason is that the colour control codes not only determine the colour of what follows, but whether the characters are to be interpreted as text characters (as given in the first table) or as graphics characters (as given in the second).

Thus ASCII code 97 appears as a letter 'a' if it follows a text control code, but as a graphics character if it follows a graphics control code. This is why teletext displays (like Prestel for example) sometimes show lines of lower case characters where the graphics control code has been corrupted. All very puzzling if you don't realise what is going on.

In teletext mode, codes 129 to 135 specify coloured text, and codes 145 to 151 specify coloured graphics. If you want to specify a coloured background, then two codes are required, one to specify the colour (and this comes first) and one (CHR$157) to specify that this applies to the background. It appears to make no difference whether text or graphics colours are used for this.

Thus a green background would result from either:

```
        CHR$130;CHR$157
or:     CHR$146;CHR$157
```

As a further experiment, just try typing in the following two lines in immediate mode and note the difference:

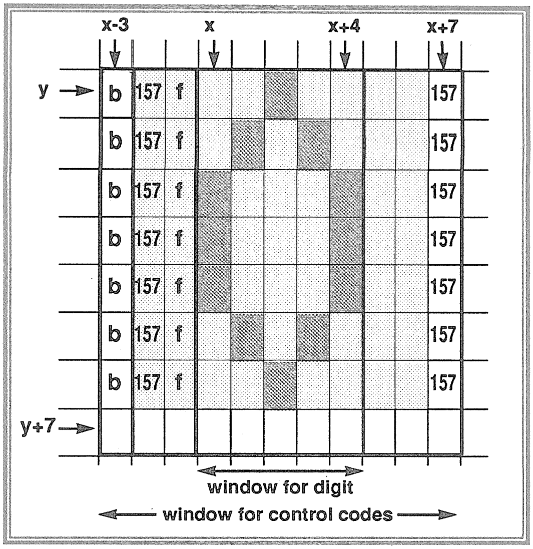


Figure 1. Layout for large coloured digit

```
PRINT CHR$130;CHR$157;CHR$132;"Hello"
PRINT CHR$130;CHR$157;CHR$148;"Hello"
```

Both specify a green background, but the first follows with blue text (CHR$132) in contrast to the second with blue graphics (CHR$148). In the first instance the word 'Hello' appears as text; in the second example the lower case characters are now treated as graphics characters.

Very often, teletext control codes can be positioned down the left-hand side of the screen, to control the use of colour on each line. In these circumstances, the codes can be positioned once and for all at the start of the program or routine.

If you want to change the colour of something already on the screen, then all you need to do is to change the colour control codes for new ones. There is no need to re-display the text or graphics itself.

Let's now consider further how colour might be applied to large digit displays. If all digits are in the same colour then there is no problem; just put the appropriate colour graphics code down the left-hand side of the screen as in last month's example. If you want a constant coloured background then this can be achieved in the same way. But suppose we would like to have choice of colour, and background, for each large digit individually. How might that be achieved?

One solution is to expand the width of the digit window. If we want to colour the digit itself, then we need to widen the window by one character, but if we want to specify foreground and background then we need to widen the window by three characters. If any one digit is always to appear in the same colour, then the definitions of the large digits would need to include colour control codes in the appropriate positions.

But if we want to select foreground and background colours when a digit is to be displayed (rather than when it is defined), the control codes will need to be placed in the window separately to the digit itself. Let's take the simplest form of PROCdigit given before, and modify this for colour. One way might be:

```
1200DEF PROCdigit(x,y,digit,f,b)
1210VDU28,x-3,y+7,x+4,y
1220FOR I%=1 TO 7
1230PRINT CHR$(b);CHR$(157);CHR$(f)
1240VDU28,x,y+7,x+4,y
1220PRINT digit$(digit);
1230 VDU26:ENDPROC
```

The window is first defined (extended three characters to the left) and the colour codes for foreground and background placed in position. Then the window is re-defined to the size for the digit alone and the corresponding characters put in place. This does have one unfortunate consequence - if you try it out you will find that all the screen to the right of the large digit is also in the background colour. We need a means to switch it off, and CHR$156 will do just that. So modify line 1210 to:

```
1210 VDU28,x-3,y+7,x+7,y
```

and change line 1230 to:

```
1230 PRINT CHR$(b);CHR$(157);CHR$(f);
  SPC7;CHR$(156)
```

The layout which we are using for each digit is shown in figure 1, using '0' as an example. That's it for now. Like I said last time, attention to detail is essential in working things out correctly, but with the help of a diagram it's not too bad.

The magazine disc/tape contains a demonstration of large coloured digits.

512 Forum

by Robin Burton

I must begin this month's Forum with an apology to any members who have written to me between late March and early May and found that a reply was rather slow in arriving. During this period the mainframe system I had been working on for the past three years had just gone live, with the result that everything else was put on 'hold', as my wife would confirm most forcefully if asked.

By the time you read this normal service will have been resumed, and hopefully in catching up no-one will be missed out.

BBCBASIC(86) REVISITED

A few months ago I reviewed BBCBASIC(86), the MS-DOS package published by M-TEC which provides a high degree of source program and operational compatibility with the BBC micro's Basic II and Basic IV.

Shortly after the review was published I received a 'phone call from a quite irate Dutch member about the review. Let me say at once that Beebug do not normally give out my phone number, but this member was obviously genuine, and since he had troubled to call from the Netherlands, in this case an exception was made.

I have since been expecting a letter from him, the content of which I had agreed to air in the Forum, but since I have not heard anything more I'll deal with the matter before any more time passes. If I omit any points he made perhaps I'll be excused on the grounds that I'm recalling a three month old telephone conversation.

He has himself been using BBCBASIC on the 512 for two or three years, and felt that the conclusions I reached didn't do the package justice, specifically in the area of execution speed. In short his own experiences were that the package did offer a gain in speed of two or three times the native BBC mode for most of the applications for which he used it. Although he agreed that my tests were (probably) accurate, and that screen output was slow, he felt the tests were un-representative and hence misleading.

I can only say that I had hoped that it was clear from the review that the speed comparison table was not supposed to represent a typical, across the board benchmark. The text made it quite clear that the FOR-NEXT loops and screen output had been specially isolated for the tests, and that for these particular operations the results were clearly disappointing. I hope that it was also clear that the overall results you would achieve in real programs would depend on what the program spends most of its time doing.

I gather that this member's main use of the package involves large amounts of calculation, and it's well known that Richard Russel prides himself, quite rightly, on his implementation of mathematical and trigonometric functions. However, I don't believe that the bulk of the average user's programs consist mainly of mathematical functions, while I do believe that screen output is a pretty common need.

It's always a risk in any review that some points may be overlooked and others overstressed, probably both by the reviewer and the reader (although with unlimited space and time perhaps both could be avoided). While I am completely happy to publicise the views of any member (even when they don't agree with me), I also stand by the contents of the review as published and believe that, in context, it was fair, balanced and accurate.

All the results were genuine, were carefully checked and can be reproduced. I should also say that no-one else has taken me to task about this subject, though presumably amongst the readership there are others who use the package.

I can only round off this item by re-stating the main conclusions again. In some areas the MS–DOS version of BBCBASIC(86) is much slower than I had expected, and in the area of screen output to an unacceptable degree. However, on the converse, its source and syntax compatibility coupled with its ease of operation are beyond reproach. As ever, only you can decide which of the strengths or weaknesses of a package are of relevance to you, and which should influence your decision.

512 BBCBASIC

The next item fits in quite neatly with the above mention of BBCBASIC(86) and last month's item on public domain (also sometimes called 'freeware', which I forgot to mention) and shareware software.

Richard Russel, as mentioned before, is the author of BBCBASIC(86), and he also wrote the version of Basic supplied with Acorn's Z80 second processor. What is not so well known is that he also wrote a version of BBC Basic exclusively for the 512, but unlike the previous two packages this one is not commercially marketed.

Having spoken to Richard, I now know a little more about the history of this software. It was originally offered to Acorn with the idea that it would be supplied with the 512, but for whatever reason Acorn declined to take up their offer. I can only assume that their financial position and general lack of interest in supporting the 512 extended to ready-made products like this too. I wonder how many more 512s there might be now if Acorn had included BBC Basic.

The fact that it's not commercially marketed is a two edged sword. On the one hand it means that you probably haven't heard about it because it isn't advertised (though I understand that it has been publicised on some bulletin boards), while on the other it means that if you can get hold of a copy it costs very little.

As I said, it was written for the 512, not for PCs running MS-DOS, so if you need a language for the 512 at home and a PC at work this won't fit the bill. It should also be borne in mind that if your use is in any way commercial, you should always buy your software and pay any multi-user licences as appropriate.

If, however, your need is for a language for the 512 for your own use then this one should suit you very well. The major difference between this and the MS-DOS version is that, as it was written for the 512/BBC micro combination and DOS Plus, with no compromises forced by alien PC hardware or MS-DOS, it provides native BBC screen modes from DOS Plus, including a proper mode 7 display. A further beneficial implication of this is that the output to screen is very much faster than in the MS–DOS version.

The software is not supported, and there are a couple of limitations specific to this version, though these are explained and quite easy to avoid. There is no printed documentation supplied with it either, though not surprisingly it's at least as easy to use as the commercial MS–DOS package, so that shouldn't deter you at all.

The final point about this software is rather good news for Forum readers. If you would like to acquire a copy all you need to do is to write to me at BEEBUG and mark the envelope '512 BBC BASIC'. You must enclose a stamped self-addressed envelope with a (512) formatted 5.25" disc. In return you will receive a free copy of the 512 version of BBC Basic, and it will only have cost you the price of the postage.

PROBLEM SOLVER

I've had several queries about *Problem Solver*, a product which claims to improve the PC compatibility of the 512 at a cost of £25.00. The reason that I have previously failed to mention it is that I have not been able to test it myself. The following is not a review, but rather is just the story so far.

I have two copies of *Problem Solver* both of which are version 1.0, but my tests have as yet been completely inconclusive. First impressions are that documentation is absolutely minimal, and very general in its claims for improved compatibility, since there is no list of software for which problems should be cured.

As you may recall, I use the 512 with a model B, but in spite of trying all three versions of *Problem Solver* (from both copies) with each of the two main DOS Plus versions (1.2 and 2.1), all I have achieved so far is to hang the machine, with seemingly no way out except re-booting.

I have also tried both copies in two Masters, one with an internally fitted 512, one external. I can then get it to load, and it does not hang the machine or interfere with normal DOS commands, but it doesn't do much else either, except that it prevents some software running that did work without it. By the way, in case you think this is just me, Andy Smith at Acorn reached precisely the same conclusions.

For the moment I'm taking the view that both copies in my possession are faulty. I certainly know of one member who had a very similar experience with his first copy, though a replacement was subsequently supplied which did work.

I have written to Shibumi Soft and for the moment I'll go no further, except to suggest that if any of you have this software and would like to supply me with your findings and comments I'll be pleased to include them in a later issue, when hopefully we can publish a proper review.

If you are able to contribute information about *Problem Solver*, I'd be interested to hear about specific packages for which it does (or doesn't) cure compatibility problems, but please include details of your hardware and software (i.e. Model B, Master, internal/external tube, version of DOS used, applications etc.).

Whether I am, or am not, finally able to successfully test this product myself, I would hope that we can at least draw up a table of conclusions showing which hardware/software configurations it helps (or hinders). After all, while £25.00 isn't a fortune, if it's unsuccessful it is a fairly expensive experiment.

COMMUNICATIONS

A few letters have again mentioned the RS232 problem when using DOS Plus, so I assume that some readers would like to use the 512 to access PC bulletin boards, of which there are quite a number. I conclude that perhaps some of you missed my earlier comments about comms software compatibility.

In simple terms there is no problem with accessing the serial port from DOS, and this is easy to prove by connecting two 512s (or your 512 and a PC) 'back to back' by means of a serial cable. This I have done several times with an Amstrad portable belonging to a friend, and we have successfully sent files in both directions.

Of course, if you can copy files to a common disc format it's probably a better way to transport data. It's faster and of course it's more convenient, as you don't need both machines in the same place at the same time. However, many machines (like the Amstrad portable) rule this out because they use 3.5" discs, and I have only 5.25" drives, so this isn't possible.

Accessing bulletin boards is of course a completely different matter, and you must use suitable software which must be capable both of driving the modem and 'talking' to the board when you connect. The problem is that most comms software expects to be able to 'illegally' access the serial port on a PC, and hence does not work in the 512. This is because, in many PCs, the legal calls simply operate too slowly for any but the slowest data transfer rates.

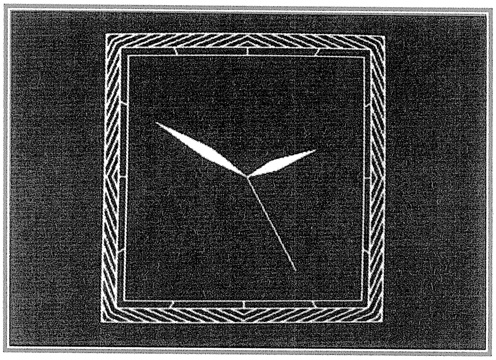
I now have an evaluation copy of *COMM+*, mentioned in a previous Forum, and will be including a report within the next couple of issues. This package is the only comms software I know which uses legal calls and therefore does work in the 512. Due to lack of time I haven't yet investigated it in any detail, but my impressions are that it is a very complete and sophisticated package.

For anyone who can't wait for the review, contact: Margolis & Co., 105 Foundling Court, Brunswick Centre, Marchmont St., London WC1N 1AN. Mention the fact that you have a 512, as there's a special offer for 512 users. You get BOTH the PC-DOS and the DOS Plus versions for £75.00 plus £2.00 P&P. This is a reduction of almost 25% on the normal retail price.

A Clock Face for the Master

Bernard Hill adds hands and face to your Master for an attractive show of timekeeping.

If you're like me, between sessions of programming I tend to walk away from my micro leaving it switched on. The 'Acorn MOS' prompt is rather uninteresting, so why not leave this little program running instead?


The Master clock face

It is quite simply an old-fashioned analogue clock, and while it is intended for the Master, it works well with Basic I or II on any BBC micro with a real-time clock fitted (provided it responds to OSWORD 14). If the program is being used solely on a Master then the function FNtime defined at line 1560 is unnecessary, and the call to the function at line 280 can be replaced with TIME$. With Basic I or II, however, there is a noticeable loss of speed in the graphics drawing caused by the trig. functions. The program can be terminated with Escape.

To use it, just type the program in and run it, after saving a copy for future use. However, a certain amount of customising is also possible:

1. The colours of the hands can be changed in line 120. To use cyan hands, replace with VDU19,2,6;0; (6 being cyan).

2. The colour of the border can be changed similarly: to produce a blue border replace line 130 with VDU19,1,4;0; (since 4 is blue).

3. The sizes of the three squares making up the face are listed in line 200. Simply change to any values up to 512, with r1<r2<r3.

4. The density of the "wood-grain" border is changeable with the variable rd in line 200.

5. The shape of the hands is modifiable. In lines 160 and 180, the variables represent:

widm	width of waist of minute hand
wm	position of waist of minute hand
lenm	length of minute hand

widh, lenh and **wh** are similar variables for the hour hand.

```
   10 REM Program Clock
   20 REM Version B1.0
   30 REM Author  Bernard Hill
   40 REM BEEBUG June 1989
   50 REM Program subject to copyright
   60 :
  100 MODE1
  110 ON ERROR GOTO 340
  120 VDU19,2,3;0;:REM hands colour
  130 VDU19,1,1;0;:REM background colour
  140 VDU23;11,0;0;0;0;
  150 REM hour hand wid,len,midpt
  160 widh=10:lenh=250:wh=100
  170 REM min hand wid,len,mid
  180 widm=7:lenm=350:wm=150
  190 REM border sqr radii & patt step
  200 r3=475:r1=400:r2=425:rd=20
  210 VDU29,640;512;
  220 GCOL0,3:PROCface:GCOL0,2
  230 m=0:h=0:s=0:PROCgettime
  240 PROChands(TRUE,h,m)
  250 lastm=m:lasth=h:PROCsec(s)
  260 REPEAT:REPEAT:PROCgettime
  270 UNTIL s<>lasts
  280 PRINTTAB(8,31);FNtime;
  290 PROCsec(lasts)
  300 IF s=0 THEN PROChands(FALSE,lasth,
lastm):PROChands(TRUE,h,m)
  310 PROCsec(s):lastm=m:lasth=h
  320 UNTIL FALSE
  330 :
  340 MODE7
  350 IF ERR<>17 THEN REPORT:PRINT" at l
ine ";ERL
  360 END
  370 :
 1000 DEF PROCface:PROCgsq(r3)
 1010 GCOL0,129:CLG:GCOL0,128
 1020 PROCsq(r2):PROCsq(r3)
 1030 FOR r=r2 TO SQR2*r3 STEP rd
 1040 PROCocto(r):NEXT
 1050 PROCgsq(r2):CLG
 1060 FOR i=0 TO 55 STEP 5:MOVE 0,0
 1070 DRAW 725*SINRAD(i*6),725*COSRAD(i*
6)
 1080 NEXT
 1090 PROCgsq(r1):CLG:PROCgsq(r2)
 1100 PROCsq(r1):PROCsq(r2):ENDPROC
 1110 :
 1120 DEF PROCgsq(x)
 1130 VDU24,-x;-x;x;x;
 1140 ENDPROC
 1150 :
 1160 DEF PROCsq(r)
 1170 MOVE -r,-r:DRAW -r,r:DRAW r,r
 1180 DRAW r,-r:DRAW -r,-r:ENDPROC
 1190 :
 1200 DEF PROCocto(r)
 1210 FOR i=0 TO 2*PI STEP PI/4
 1220 s=SINi:c=COSi
 1230 IF i=0 THEN MOVE r*s,r*c ELSE DRAW
 r*s,r*c
 1240 NEXT:ENDPROC
 1250 :
 1260 DEF PROCarm(a,r1,r2)
 1270 s=SINRAD(a*6):c=COSRAD(a*6)
 1280 MOVE r1*s,r1*c:DRAW r2*s,r2*c
 1290 ENDPROC
 1300 :
 1310 DEF PROCgettime
 1320 A%=14:X%=&70:Y%=0:?&70=1:CALL -15
 1330 lasts=s:s=FNbcd(?&76)
 1340 m=FNbcd(?&75)
 1350 h=FNbcd(?&74) MOD 12:ENDPROC
 1360 :
 1370 DEF FNbcd(a)=EVAL(STR$~a)
 1380 :
 1390 DEF PROCsec(s)
 1400 GCOL4,1:PROCarm(s,0,350)
 1410 GCOL0,1:ENDPROC
 1420 :
 1430 DEF PROChands(on,h,m)
 1440 GCOL0,2:h=(h+m/60)*5
 1450 a1=RAD(h*6+widh):a2=RAD(h*6-widh)
 1460 IF on THEN p=85 ELSE p=87
 1470 MOVE 0,0:MOVE wh*SINa1,wh*COSa1
 1480 PLOT p,wh*SINa2,wh*COSa2
 1490 PLOT p,lenh*SINRAD(6*h),lenh*COSRA
D(6*h)
 1500 a1=RAD(m*6+widm):a2=RAD(m*6-widm)
 1510 MOVE 0,0:MOVE wm*SINa1,wm*COSa1
 1520 PLOT p,wm*SINa2,wm*COSa2
 1530 PLOT p,lenm*SINRAD(6*m),lenm*COSRA
D(6*m)
 1540 ENDPROC
 1550 :
 1560 DEF FNtime
 1570 X%=0:Y%=1:A%=14:?&100=0:CALL -15
 1580 =$&100
```

The New BBC Microcomputer (continued from page 8)

fully Archimedes and RISC OS compatible with no dilution or watering down of the Archimedes' essential features. Clearly it is aimed more at the home user (than the business market for which dual disc drives if not a hard disc are a standard and essential requisite). It is also likely that Acorn will target the machine strongly into the educational market, where it may well prove successful as a lower cost entry system for primary schools, and as a cheaper network station for Econet systems.

If you have still not made the plunge and gone for an Archimedes, then the A3000 will surely give food for thought, and has much to offer. It doesn't have all the expansion capability of the latest Archimedes 400/1 series, but it offers better performance and more memory than the original A310 was capable of.

And what about the price of £649 (plus VAT)? That is always a difficult question to answer, and in fact there is no absolute answer. If the A3000 captures the public imagination and becomes THE machine to buy and be seen buying, Acorn would probably feel that they had priced the machine too low. If it is seen as a relative failure, then critics will no doubt argue that it is because Acorn set the price too high. On first impressions Acorn has got the price about right. The Archimedes has already established itself as one of the most desirable micros around, and packaging that up as Acorn has done at a lower price has to be a good move.

As to when you will be able to buy one of the new machines, Acorn says that it will be available in quantity at the time of the BBC Acorn User Show in July, which is where the machine will have its first public showing. All in all, exciting times lie ahead for Acorn.

NOTE: BEEBUG is already accepting orders for the A3000, and BEEBUG members needn't wait till July to see a machine in action. We have one on demonstration in our St Albans showroom now. Pay us a visit.

Spin a Disc (3)

David Spencer gets to grips with his general purpose disc access ROM.

At the end of last month's article we defined the format of the parameter block to be used by our general disc access routine, and said that it would be implemented as an OSWORD call contained within a ROM image. This month, we will present the actual code, giving detailed explanations wherever possible.

Before getting down to the actual code, we need to make a couple of changes to the definition of the parameter block from last month. Referring to last months article, the parameter block was defined to be nine bytes long. We shall now add another two bytes. The first (byte 9) is the speed at which to step between tracks. The second addition, at byte 10, is used to specify the track that the drive is currently on. If a value of 255 is given then this means that the current track is unknown. We will come back to both these parameters next month.

RUNNING THE PROGRAM

The listing should be typed in and saved. When the program is run, it will assemble the ROM image and save it to disc with the filename 'DiscROM'. This should then be loaded into sideways RAM and initialised with Ctrl-Break.

LOOKING AT THE PROGRAM

The workings of the program are fairly easy to understand if they are taken step by step. The first part of the code is the ROM header. This picks up the particular OSWORD call and copies the parameter block into memory starting at location &B0. The routine *discop* is then called.

The first function of *discop* is to call the routine *setnmi*. This deals with the setting up of a routine to handle the interrupts generated by the disc controller. This is a two stage process. Firstly, a service call is issued to all the other ROMs (lines 1210-1220) to give the current user of non-maskable interrupts (MNIs) a chance to stop using them! Secondly, the NMI handling routine must be poked into memory starting at location &D00. This code is copied directly from the ROM image, starting at the label *nmicode*. This is in fact the routine for reading data - for writing, the routine is modified after it has been copied. Finally, the address of the data in memory is poked into the NMI routine.

The next stage involves the routine *set1770*. This converts the drive and side numbers, and the density from the format used in the parameter block into a value which is put in the control latch (see last month). A check is then made to see if the 1770 thinks it is executing a command. This is necessary due to a hardware bug in the chip itself. If the status register indicates that a command is in progress, then the 1770 is sent a command to force it to stop (lines 740-750).

Once the 1770 has been set up, the drive head is moved to the correct track using the routine *seek*. This takes the current track number and the desired track number, and uses the 1770 Seek-Track command to move the head. If the current track number is not known (a value of 255 is specified), then the drive head is restored to track 0 first. *Seek* makes use of a sub-routine called *command*. This waits for the 1770 to finish its current command, and then sends the command specified by the A register to the 1770. When the command has finished, the

NMI routine will set a flag in location &A2, and command will not exit until this is set. Alternatively, if Escape is pressed then an immediate exit is made.

The routine *transfer* performs the actual data transfer. This is relatively straightforward, and does little more than translating the operation code given in the OSWORD call to a 1770 command and executing it. The data is transferred one sector at a time, and the command is therefore issued as many times as is necessary to transfer the specified number of sectors. The subroutine comm issues the read and write commands to the 1770, and detects any errors generated. If an error does occur, the operation is repeated up to six times before the routine gives up. If the error is a 'sector not found error' then the drive head is repositioned between each retry in case the error was caused by a problem with moving the head.

Finally, before the routine exits, *nmioff* tidies up the NMI routine. It does this by releasing the use of NMIs to whichever ROM was using them before our routine took over.

TESTING THE ROUTINE

The use of the OSWORD call will be covered in full next month. For the time being though, it can be tested using the following method.

Put a DFS format disc in drive 0 and set up the following parameter block:

```
?&900=0
!&901=&2000
?&905=8
?&906=0
?&907=0
?&908=1
?&909=0
?&90A=255
```

Then call the routine using:

```
A%=115:X%=0:Y%=9:CALL &FFF1
```

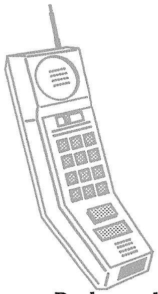
The disc drive should spring into action and the prompt return after a second or two. If you then look at the 256 bytes of memory starting at address &2000 (using a memory editor) you should be able to see the names of all the files on the disc. This is because the command has read in the first sector of the disc catalogue.

Next month we will show how to use the routine fully, and use it to implement a high speed ADFS formatter.

```
   10 REM Program Disc Access ROM
   20 REM Version B1.0
   30 REM Author  David Spencer
   40 REM BEEBUG  June 1989
   50 REM Program subject to copyright
   60 :
   70 A%=0:X%=1:os=(USR &FFF4 AND &FF00)
/256
   80 IF os=1 THEN PROCmodelb
   90 IF os=2 THEN PROCbplus
  100 IF os=3 THEN PROCmaster
  110 status=wd1770:track=wd1770+1
  120 sector=wd1770+2:data=wd1770+3
  130 DIM code 1000
  140 FOR pass=0 TO 3 STEP 3
  150 P%=&8000:O%=code
  160 [OPT pass+4
  170 EQUB 0:EQUB 0:EQUB 0
  180 JMP service
  190 EQUB &82
  200 EQUB copy AND &FF:EQUB 1
  210 EQUS "BEEBUG DISC ACCESS ROM"
  220 .copy EQUB 0
  230 EQUS "(C) BEEBUG 1989":EQUB 0
  240 :
  250 .service
  260 CMP #8:BEQ osword:RTS
  270 .osword LDA &EF:CMP #115
  280 BEQ osword2:LDA #8:RTS
  290 .osword2 TYA:PHA:PHP
  300 CLI:JSR discop:PLP
  310 PLA:TAY:LDX &F4:LDA #0:RTS
  320 :
  330 .discop LDA &F0:STA &BE
  340 LDA &F1:STA &BF:LDY #10
  350 .discop2 LDA (&F0),Y:STA &B0,Y
  360 DEY:BPL discop2
  370 LDA &B0:JSR setnmi
  380 TSX:STX &A4
  390 LDA &B5:JSR set1770
  400 JSR seek
  410 LDA &B0:JSR transfer
  420 LDA &BA:LDY #10:STA (&BE),Y
  430 JMP nmioff
  440 :
  450 .seek
  460 LDA &BA:CMP #&FF:BNE seek2
  470 LDA &B9:AND #3
  480 JSR command:LDA #0:STA &BA
  490 .seek2 LDA &BA:STA track
  500 LDA &B6:STA data
  510 LDA &B7:STA sector
  520 LDA &B9:AND #3:ORA #16
  530 JSR command:LDA track
  540 STA &BA:RTS
```

```
 550 :
 560 .transfer
 570 LDA #&80:LDX &B0:CPX #0:BEQ read
 580 LDA #&A0:CPX #1:BEQ write:LDA #&F0
 590 .write LDX track:CPX #60:BCS read
 600 ORA #2
 610 .read CPX #3:BCS done
 620 PHA:LDA &B7:STA sector:PLA
 630 JSR comm:DEC &B8
 640 BEQ done:LDA &B0:CMP #2:BEQ done
 650 INC &B7:LDA &B5:LDX #10
 660 CMP #8:BCS ss:LDX #16
 670 .ss CPX &B7:BNE transfer
 680 LDA #0:STA &B7
 690 INC &B6:JSR seek:JMP transfer
 700 .done RTS
 710 :
 720 .set1770
 730 LDX &B5:LDA ctab,X:STA control
 740 LDA status:AND #1:BEQ set17702
 750 LDA #&D0:STA status
 760 .set17702 RTS
 770 :
 780 .command
 790 PHA:LDA #0
 800 STA &A1:STA &A2
 810 .command2 LDA status:AND #1
 820 BNE command2:PLA:STA status
 830 .command3 BIT &FF:BMI error3
 840 LDA &A2:BEQ command3
 850 RTS
 860 :
 870 .comm
 880 PHA:LDA #6:STA &A3
 890 .comm2 PLA:PHA:JSR command
 900 LDA &A1:BNE error:PLA:RTS
 910 .error LDA &A0:AND #&40:BNE error3
 920 LDA &A0:AND #&10
 930 BEQ error2:LDA #&FF:STA &BA
 940 JSR seek
 950 .error2 DEC &A3:BNE comm2
 960 .error3 LDX &A4:TXS
 970 LDA #&FF:LDY #0:STA (&BE),Y
 980 LDA &A0:INY:STA (&BE),Y
 990 JMP nmioff
1000 :
1010 .nmicode
1020 PHA:LDA status:AND #&1F:CMP #3
1030 BNE notdata
1040 LDA data:.nbase STA &FFFF
1050 INC nbase-nmicode+&D01
1060 BNE nohigh
1070 INC nbase-nmicode+&D02
1080 .nohigh PLA:RTI
1090 .notdata AND #&58:BEQ noerr
1100 STA &A0:LDA #1:STA &A1
1110 .noerr LDA #1:STA &A2:PLA:RTI
1120 :
1130 .nmicode2
1140 LDA &FFFF:STA data
1150 INC nbase-nmicode+&CFE
1160 BNE nohigh2
1170 INC nbase-nmicode+&CFF
1180 .nohigh2
1190 :
1200 .setnmi
1210 PHA:LDA #143:LDX #12
1220 JSR &FFF4:STY &A7
1230 LDX #nmicode2-nmicode-1
1240 .setnmi2 LDA nmicode,X
1250 STA &D00,X:DEX:BPL setnmi2
1260 PLA:BEQ setnmi4
1270 LDX #nohigh2-nmicode2-1
1280 .setnmi3 LDA nmicode2,X
1290 STA &D0A,X:DEX:BPL setnmi3
1300 LDA #&5F:STA &D05
1310 LDA &B1:STA &D0B
1320 LDA &B2:STA &D0C:RTS
1330 .setnmi4 LDA &B1:STA &D0E
1340 LDA &B2:STA &D0F:RTS
1350 :
1360 .nmioff
1370 LDA #&40:STA &D00:LDA #143
1380 LDX #11:LDY &A7:JMP &FFF4
1390 :
1400 .ctab EQUB line1:EQUB line1+1
1410 EQUB line1:EQUB line1+1
1420 EQUB line2:EQUB line2+1
1430 EQUB line2:EQUB line2+1
1440 EQUB line3:EQUB line3+1
1450 EQUB line3:EQUB line3+1
1460 EQUB line4:EQUB line4+1
1470 EQUB line4:EQUB line4+1
1480 ]NEXT
1490 OSCLI ("SAVE DiscROM "+STR$~code+"
"+STR$~O%)
1500 END
1510 :
1520 DEF PROCmodelb
1530 control=&FE80
1540 wd1770=&FE84
1550 line1=1:line2=5:line3=9:line4=&B
1560 ENDPROC
1570 :
1580 DEF PROCbplus
1590 control=&FE80
1600 wd1770=&FE84
1610 line1=&31:line2=&35:line3=&39:line
4=&3B
1620 ENDPROC
1630 :
1640 DEF PROCmaster
1650 control=&FE24
1660 wd1770=&FE28
1670 line1=5:line2=&15:line3=&25:line4=
&35
1680 ENDPROC
```

The Comms Spot



by Peter Rochford

This month it is back to basics in the Comms Spot and time for some explanation on how your Beeb sends and receives data via its serial port.

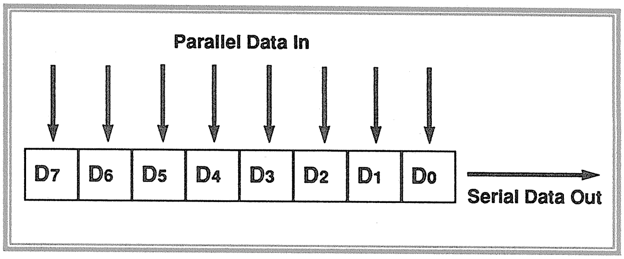


Figure 1. Parallel to serial data conversion

As you should all probably know by now, the Beeb is an eight bit computer, and eight bits make one byte. The computer handles all its operations internally, one byte at a time, using an eight bit data bus and sending all eight bits along eight separate data lines simultaneously.

When communicating with an external device, such as a printer, it is usually via the computer's parallel port. This enables the computer to communicate with the printer in much the same way as it processes data internally, i.e. eight bits at a time.

However, when you need to communicate with an external piece of hardware over a long cable, this form of communication becomes impossible. In the case of a printer for example, the extra length of the cable causes timing problems. This means simply that the eight bits of the byte do not arrive together at the receiving end, making a nonsense of what it is being sent. Hence the reason that the cables on your disc drives are so short and that the cable to your printer is supplied only in a metre or two metre length.

To get round this problem we use another form of data transmission called serial transmission. Here the eight bits of the byte are sent sequentially instead of together. It is therefore slower than parallel data transmission.

This job is handled by the serial port of your computer, where each byte is chopped up into its component eight bits to be transmitted one after the other. This is illustrated in Fig.1.

This is an over simplification of course, because if this was the case, the receiving end would not know when each byte started and ended, and could get out of step with the sender. To overcome this, the computer introduces a start bit to tell the receiving computer that it is about to send another eight bits of data. After these have been transmitted, it then sends a stop bit to indicate that all bits of that byte have been sent.

Checking the integrity of the received data is done by using what is known as a parity bit. This can only be done when the first seven bits of the byte are used, such as when sending ASCII characters. The unused eighth (top) bit, then becomes a parity bit. There are two types of parity, odd and even.

What happens is that the sending computer counts all bits set to one in the byte being sent. If odd parity is used, the computer will set the parity bit to make the total number of bits set to one odd. If even parity is used the parity bit will be set to make the number of ones even. If the number of ones in the received byte does not match the parity bit, then the computer signals an error.

This does not guarantee total integrity of the received data, as it is possible for the data to be corrupted in such a way as to still make the parity bit appear correct for that byte.

When transmitting full eight bit data no parity is used. With this type of data a form of error checking such as Xmodem in the comms software is used. Xmodem sends its data in 128 byte blocks. A checksum is calculated for the total value of the bytes in the block and sent along with the block. At the receiving end the software compares the data in the received block with the checksum. If it is OK then the next block is requested or else the last block is requested again. Xmodem can still be used of course with 7 bit data as well as 8 bit and virtually guarantees that your file of data will be transferred correctly.

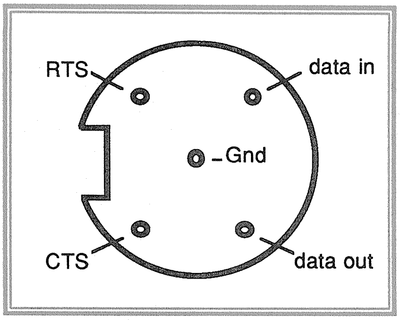


Figure 2. The Beeb RS423 port view into socket

The serial port in your Beeb is called an RS423 interface. This is a subset of the original 'standard' RS232 that has been in existence for many years as an interface for the sending and receiving of serial data.

The full RS232 consists of some twenty lines, but many of these are only for very specialised use, and in practice most computers only use up to about 10 of them. To give a list of the lines or a diagram of the full RS232 here, would be of little interest. Fig. 2 shows the RS423 in your Beeb which uses only five of the lines common to the RS232. In fact, for many applications you could get away with just using three of them.

The TXD pin is for transmitting data from your computer, whilst RXD is for the received data. The centre pin is is the 0V line and used for signal ground. These three connections on their own are enough for simple data reception and transmission.

The other two lines are used for what is termed 'handshaking'. Even computers have manners! The computer needs some way of knowing when it is supposed to be sending or receiving data. RTS is the Request To Send line and is used by the computer to tell the receiving device connected to it that it should stop sending data because the receiver buffer is getting full up. The CTS (Clear To Send) line performs the opposite function. If the remote device's receiver buffer is nearly full, then it signals via the CTS line that it does not want to receive any more data for the time being. Note that this use of RTS and CTS is subtly different to the use defined in the RS232 standard, and this can sometimes cause problems when connecting serial devices to the Beeb.

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

BOUNCER BALL

(Vol. 8 No.1 Page 55)

There is a slight problem with this program which results in no bonus being given when the letters B-O-N-U-S are lit. This can be cured by replacing SC% by s% in lines 1270 and 1700.

JULIA SETS

(Vol. 8 No. 1 Page 6)

The address of 'acomp' in Basic I was given incorrectly as &AD99, when in fact it should be &ADA0.

DECIMAL ARITHMETIC

(Vol. 7 No. 9 Page 30)

Listing 1, line 1160 should be changed to:

```
.move_left TXA:CPX#0:BEQ MSD_filled
```

DECIMAL ARITHMETIC

(Vol. 7 No. 10 Page 38)

In line 260 of listing 2,

```
MID$(arg$,1)
```

should be changed to:

```
MID$(arg$,2)
```

Share Investor (Part 2)

Graham Marsh concludes the description of his investment analysis program, and provides some technical detail for those who may wish to modify the program for their own requirements.

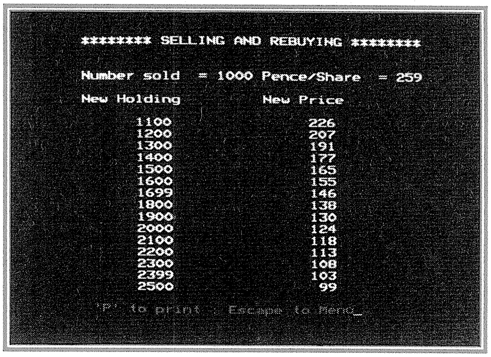
This month we will add the two remaining menu options to the program listed in the last issue. The safest way is to enter the additional lines of code and first save to disc (or tape) as a normal program. You should also save a spooled copy under a different name (see your User Guide for information on spooled files). Now load the program from last month, and use *EXEC to append the additional code from the spooled file to the original program. Again for safety, save the combined program under a new name.

It is, of course essential that the program from part one was entered with the line numbers exactly as listed. You can now run the complete program as described last month. Note that line 1510 in last month's listing (a REM statement) is quite redundant following reorganisation of the program.

The new functions are for selling and rebuying, and for new issues, and these are described next.

OPTION 4 - SELL AND REBUY

If a normally successful company is expected to go through a bad patch, or it is anticipated that its shares will be depressed by other events, it could be worth selling to limit the cash loss, and then to rebuy at a later date. This could be in a matter of weeks or even days, when recovery can be expected because of the basically sound nature of the company involved. This would obviously increase the share holding in the company for the same cash outlay, and bring better returns when share dividends are distributed. The same would be true if rights issues were made since the allocation is based upon the total share holding. In addition there would be a more significant capital appreciation which would result from a greater holding.



```
******** SELLING AND REBUYING ********

Number sold  = 1000 Pence/Share  = 259

New Holding             New Price

    1100                   226
    1200                   207
    1300                   191
    1400                   177
    1500                   165
    1600                   155
    1699                   146
    1800                   138
    1900                   130
    2000                   124
    2100                   118
    2200                   113
    2300                   108
    2399                   103
    2500                    99

 'P' to print : Escape to Menu_
```

Figure 1. Sell and rebuy

The program shows the numbers of shares which could be purchased at a range of prices (downwards from the current level) by selling your current holding, and takes all relevant broker fees and stamp duty into account.

OPTION 5 - NEW ISSUES

These transactions are different from normal share purchases in that neither broker's fee nor transfer stamp charges are involved in the purchase, and the number of shares bought is simply the number obtained by dividing the cash available by price of each share. Continue gives the usual monthly analysis.

That completes the description of the analysis program and its features. The program information which follows is intended to help those who may wish to modify the program, or otherwise tailor it to their own needs.

NOTE: In printing last month's listing, the '£' signs used in the program were accidentally reproduced as a back-facing single quote. The lines in question which should be altered are 1020, 1580, 2150, 2390, 2400 *and* 2410. *Any '£' signs are listed correctly in the code given here.*

PROCEDURES AND FUNCTIONS

Most of the procedures and functions should be clear enough to follow but some comment might be appropriate.

The function **FNeven** returns the break even value for a specified number of months. If the function is called with a parameter of zero (months) a short term value only is returned which only takes account of the costs of buying and selling, but break even values for one month or more include the savings account interest factor.

FNcost represents the value of the shares purchased including the costs of buying, thus making it clear how important this is, and that an initial rise in share value is needed just to recover this.

FNcommission calculates the broker's commission on any transaction using either the minimum fee or the percentage, whichever is the greater.

FNmonths simply calculates the number of months the shares have been held.

FNcash gives the cash increase over the original investment.

FNnumber calculates the number of shares you can afford for a given investment.

FNnewprice is used in Option 4 for calculating the revised (reduced) price in order to achieve a given increase in holding.

FNsaleval is the cash resulting from a sale taking dealing fees into account.

There are two 'continue' procedures, one for continuing to the monthly analysis and another to proceed to the menu after that. These are called **PROCcont1** and **PROCcont2**.

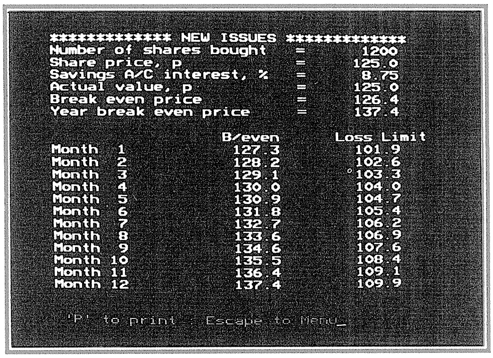
PROCerror is used as a two-fold facility to report errors, and also to return to the menu at any time after Escape.

The other functions and procedures used in the program should be clear from their names, and relate in most cases to the five menu options.

VARIABLES

The program uses a number of global variables as follows:

Ca	Cash
Co	Commission
Sp1	Initial share price
Sp2	Final share price
Cost	Cost of purchasing shares
Nr1	Number of shares bought
Nr2	Number of shares sold
Yr1	Year bought
Yr2	Year sold
M1	Month bought
M2	Month sold
CaInc	Cash increase



```
************** NEW ISSUES *************
Number of shares bought    =       1200
Share price, p             =      125.0
Savings A/C interest, %    =       8.75
Actual value, p            =      125.0
Break even price           =      126.4
Year break even price      =      137.4

                      B/even      Loss Limit
Month  1               127.3        101.9
Month  2               128.2        102.6
Month  3               129.1        103.3
Month  4               130.0        104.0
Month  5               130.9        104.7
Month  6               131.8        105.4
Month  7               132.7        106.2
Month  8               133.6        106.9
Month  9               134.6        107.6
Month 10               135.5        108.4
Month 11               136.4        109.1
Month 12               137.4        109.9

  'P' to print : Escape to Menu_
```

Figure 2. New issue analysis

```
 1360 DEF PROCsellnbuy
 1370 FL=4:PROCtitle1("SELLING AND REBUY
ING")
 1380 REPEAT:Nr2=FNinput(0,CHR$134+"Numb
er sold"+CHR$135,0,15,2):UNTIL Nr2>0
 1390 REPEAT:Sp2=FNinput(0,CHR$134+"Penc
e/Share"+CHR$135,20,35,2):UNTIL Sp2>0
 1400 PROCreport4:PROCcont1
 1410 ENDPROC
 1420 :
 1430 DEF PROCnew_issues
 1440 FL=5:PROCtitle1("NEW ISSUES"):PRIN
T:
 1450 REPEAT:Nr1=FNinput(0,"Number of sh
ares",0,28,3):UNTIL Nr1>0
 1460 REPEAT:Sp1=FNinput(0,"Share price,
 pence",0,28,6):UNTILSp1>0
 1470 Cost=Sp1*Nr1/100
 1480 CLS:PROCreport5:PROCcont1
 1490 ENDPROC
 1500 :
 2000 IF N%=4 THEN PROCsellnbuy
 2010 IF N%=5 THEN PROCnew_issues
 2480 DEF PROCreport4
 2490 LOCAL H
 2500 PRINTTAB(0,4);CHR$134;"New Holding
";TAB(20);CHR$134;"New Price":PRINT
 2510 FOR H=10 TO 150 STEP10
 2520 IF INT(FNnewprice)<2THEN PRINT"Not
 possible for share to reach a low":PRIN
T:PRINT"enough price to increase holding
 further":PROCcont1:ENDPROC
 2530 PRINT INT(Nr2*(1+H/100));
 2540 PRINTTAB(18)INT(FNnewprice)
 2550 NEXT:@%=10
 2560 ENDPROC
 2570 :
 2580 DEF PROCreport5
 2590 PROCreport("Number of shares bough
t",28,Nr1,0)
 2600 PROCreport("Share price, p",28,Sp1
,1)
 2610 PROCreport("Savings A/C interest,
%",28,D%/100,2)
 2620 PROCreport("Actual value, p",28,Sp
1,1)
 2630 PROCreport("Break even price",28,F
Neven(0),1)
 2640 PROCreport("Year break even price"
,28,FNeven(12),1)
 2650 PRINT:PRINTTAB(18)"B/even";:PRINTT
AB(30)"Loss Limit"
 2660 FOR Month=1 TO 12
 2670 @%=&09:PRINT"Month ";SPC(-(Month<1
0))Month;
 2680 @%=&20109:PRINTTAB(15)FNeven(Month
);TAB(32)E%/100*FNeven(Month)
 2690 NEXT
 2700 @%=10:ENDPROC
 2710 :
```

EVEN THE GREATEST PERFORMANCES CAN BE IMPROVED

NEW 32K ROM **WORDWISE PLUS II** For model B, B+ and Master

All the features of Wordwise Plus, and more...

Produced with the co-operation of Computer Concepts, Wordwise Plus II (WW+2) is a powerful yet easy to use word-processing system. Simply remove your existing word-processor chip (if any), and plug in the WW+2 ROM.

Many operations are menu driven, so that prompts are given when needed. Safety-nets make it almost impossible to lose text by accident.

WW+2 has the features of many more expensive programs. Automatic headings and footings, page/line numbering, word counts etc. The usual move, copy and delete operations are easily achieved with the function keys.

Up to eleven separate documents may be held in memory at once, and text readily transferred between them.

WW+2 is 100% compatible with old Wordwise/Plus files, ADFS and Spell Master.

We have received many letters of praise from WW+2 users, and are grateful for permission to quote the following;

"..every bit as good as I was led to believe. So good, indeed, that I would recommend all Wordwise Plus users to upgrade to WW+2." D Sims, Cambridge.

"..excellent additions to an already outstanding piece of software."

"The WW+2 ROM alone is superb, and the extra utilities make it very good value."

See also page 139 BBC Acorn User April 1989.

Upgrades from Wordwise/Plus are available.

Just some of the new WW+2 extensions...

Many Interword-type features are present, such as menu driven loading and saving of text. Over 12 extra CTRL keys allow you to delete a line, mark a word or paragraph, find any string etc.

Improved search and replace options now work from edit mode, and on segments too. Lots of new "short cuts" are present. A single keypress (instead of five) selects the text or any segment.

Long documents may be previewed in 80 columns, even on an ordinary model B.

Programs supplied

Two discs of example programs are supplied with WW+2, and include a very fast sorting routine, printer buffer, mail-merge program, label printer, multiple column printout, Basic program editor, and many other utilities. Even floating point maths is now possible.

Send for our WW+2 information sheet.

Thousands of users still prefer the Wordwise system. You can order WW+2 by credit card on (0752) 847286. Or send c.w.o. to the address below. Official orders welcome.

28 day no quibble money-back guarantee.

Wordwise Plus II Prices (inc. VAT)

Upgrade from Wordwise Plus	£37.95
Upgrade from Wordwise	£47.15
Wordwise Plus II complete	£56.35
Spell Master	£51.00
(£50 when purchased with WW+2)	
16K sideways RAM module	£16.00

Please allow £1 postage

Order with Access on (0752) 847286

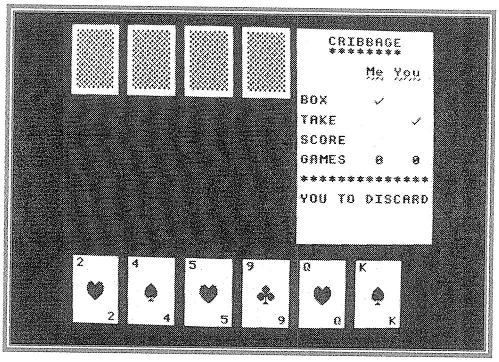
IFEL
36 Upland Drive
Derriford
Plymouth PL6 6BD
(0752) 847286

♣♦♠♥ The Game of Cribbage ♥♠♦♣

In this authentic computer implementation, Graham Crossley explains the rules and play for the traditional game of Cribbage.

THE OBJECT AND RULES

Cribbage is a card game for two players, where the object is to score points for certain combinations of cards. The first player to reach 121 points is the winner of the game. In this computer implementation, the Beeb plays one of the hands, and will prove a tough opponent to beat.



```
CRIBBAGE
********
           Me  You
BOX        ✓
TAKE            ✓
SCORE
GAMES      0    0
**************
YOU TO DISCARD
```

♣ *Selecting the 'box'* ♣

Cribbage may seem a complex game to master, but a little practice with this program will soon get you going. The rules played by this version are as follows:

Six cards are dealt to each player after which both must discard 2 in order to make up a third hand called the "box". The deck is then cut and the top card turned face up and placed to the left between the players. The hands are now played, with opponents laying cards alternately while trying to score points. When both players have used all their cards, the value of their hands is also added to their scores. Players take it in turns to have the value of the box hand added to their score.

SCORING POINTS

Two points may be scored on the cut. If a Jack is cut then the player who owns the box is awarded 2 points. When hands (including the box) are valued they score points as follows:

1. Firstly the card which was cut becomes a fifth card and is included in the hand's value. The player who does not own the box has his hand counted and added to his total first. This is known as the 'take'.

2. Each different combination of cards which add up to 15 scores 2 points. The Ace counts as 1 and J, Q, K as 10 each.

3. Two cards the same (i.e. a pair) scores 2, while three or four the same score 6 and 12 points respectively.

4. Runs of three or more cards score the number of points that there are cards in the run, i.e. 3 for a run of three.

5. If all four cards in a players hand are of the same suit 4 points are scored, and if the cut card is also of the same suit then a fifth point is added. When the box is being valued then only the 5 points may be scored.

6. If the hand holds the Jack of the same suit as the cut card then 1 point is scored.

7. The same card(s) may be used in several combinations, e.g. 6678K would score 10 - 2 for the 15, 2 for the pair and 6 in two runs.

PLAYING THE HANDS

During play the running total of cards played so far is maintained until either the total reaches 31 or neither player can lay a card without the total exceeding 31. The player who lays the last card scores 1 point, but 2 points if the total was 31. The cards played are then erased, the running total zeroed and the process repeated until all cards have been played.

Points may also be scored as follows:

1. If a player makes the total 15 then 2 points are scored.

2. Two (i.e. a pair), three or four cards the same played in succession score 2, 6 or 12 points respectively.

3. An unbroken run of three or more cards played earns a player the number of points as there are cards in the run. The difference here is that runs may be in the reverse order.

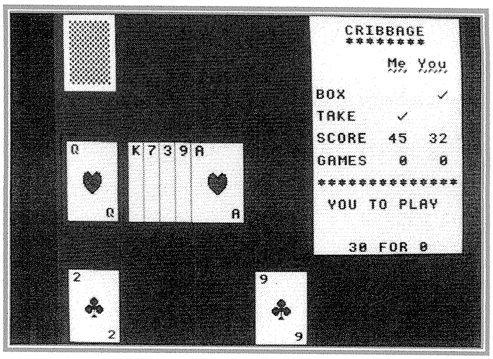
4. Combinations of the above are possible ie, if a 6 and 5 have been played making a running total of 11, playing a 4 will score 5; 3 for the run and 2 for the 15. Similarly, if the last card played was a 2 and the total is at 29 then playing another 2 will score 4; 2 for the pair and 2 for the 31. Three 5's played in succession scores 8; 6 for three cards the same and 2 for the 15.

ABOUT THE PROGRAM

Due to its size, the program has been split into two with *Crib1* chaining *Crib2*. It is important that line numbers are not changed when you type the programs in. The programs have also been written in a relatively 'dense' fashion to save space.

During play a text window shows which player has the 'box', which has the 'take', the players' scores and the number of games won. Messages are displayed below the line of asterisks with the running total, and any points scored shown on the lower line. Cards to be discarded or played are selected by typing the number shown below the appropriate card(s) followed by Return. A choice may be cancelled with the Delete key, while hitting the zero key during play indicates that you cannot play. However, if you can play then you must do so.

If you wish to learn the game of Cribbage then this program is ideal. If you have played before it will give you a good run for your money.


```
 CRIBBAGE
 ********
        Me  You
BOX          ✓
TAKE     ✓
SCORE   45   32
GAMES    0    0
**************
 YOU TO PLAY

   30 FOR 0
```

♣ *A game in progress* ♣

♣ ♦ ♠ ♥ ♣ ♦ ♠ ♥ ♣ ♦
♥ **Have fun.** ♠
♣ ♦ ♠ ♥ ♣ ♦ ♠ ♥ ♣ ♦

```
   10 REM Program Cribbage Loader
   20 REM Version B1.0
   30 REM Author  Graham Crossley
   40 REM BEEBUG  June 1989
   50 REM Program subject to copyright
   60 :
  100 *KEY0 *TAPE|MFORN%=0TOTOP-PAGE STE
P4:N%!&E00=N%!PAGE:NEXT|MPAGE=&E00|MOLD|
MRUN|F|M
  110 *KEY1 DEL.10,140|MRUN|F|M
  120 RESTORE130:FORN%=128TO146:READA%,B
%,C%,D%,E%,F%,G%,H%:VDU23,N%,A%,B%,C%,D%
,E%,F%,G%,H%:NEXT
  130 DATA 204,204,51,51,204,204,51,51,3
,7,15,15,15,7,59,127,128,192,224,224,224
,192,184,252,255,255,253,125,57,1,3,7,25
4,254,126,124,56,0,128,192,24,124,126,25
5,255,255,255,255,48,124,252,254,254,254
,254,254,255,127,127,63,31,15,7,3
  140 DATA 254,252,252,248,240,224,192,1
28,1,3,7,15,31,31,63,63,0,128,192,224,24
0,240,248,248,63,63,63,31,13,1,3,7,248,2
48,248,240,96,0,128,192,1,3,7,15,31,63,1
27,255,128,192,224,240,248,252,254,255,2
55,127,63,31,15,7,3,1
  150 DATA 255,254,252,248,240,224,192,1
28,206,219,219,219,219,219,206,0,1,3,6,1
40,216,112,32,0
  160 MODE7:CHAIN"CRIB2"
```

```
   10 REM Program Cribbage
   20 REM Version B1.0
   30 REM Author  Graham Crossley
   40 REM BEEBUG  June 1989
   50 REM Program subject to copyright
   60 :
  100 IFPAGE>&E00 VDU21:*FX138,0,128
  110 MODE1:VDU19,2,2;0;:GCOL0,130:GCOL0
,0:CLG:GCOL0,128:MOVE16,656:DRAW208,656:
DRAW208,395:DRAW16,395:DRAW16,656:VDU24,
800;311;1264;1007;:CLG:GCOL0,131:VDU24,7
92;319;1256;1015;:CLG:COLOUR0:COLOUR131:
VDU28,25,21,38,1,4,23,1,0;0;0;0;
  120 PRINTTAB(3,0)"CRIBBAGE"'TAB(3)"***
*****"''TAB(7)"Me You"'TAB(7)"~~ ~~~"''"
BOX"''"TAKE"''"SCORE"''"GAMES   0    0"''
STRING$(14,"*")
  130 VDU21:*FX138,0,129
  140 END
  150 ONERRORVDU26,12:REPORT:PRINT" at l
ine ";ERL:END
  160 PROCi:REPEAT:PROCg:PROCw:UNTILFNnm
  170 END
 1000 DEFPROCg
 1010 PROCx:M%=RND(-TIME):G%=0:sc%?0=0:s
c%?1=0:REPEAT:PROCsh:PROCd:PROCst:PROCdi
:PROCsb:PROCkt:PROCpl:PROCch:UNTILG%
 1020 ENDPROC
 1030 DEFPROCi
 1040 DIMh% 4,s% 4,hi% 15,k% 1,sc% 1,R%
7,d% 51,p% 51,g%(1),w$(1):box%=RND(2)-1:
w$(0)="  I  ":w$(1)=" YOU":n$="   1     2
     3     4     5     6":FORN%=0TO51:d%
?N%=N%:NEXT
 1050 ENDPROC
 1060 DEFPROCst
 1070 box%=ABS(box%-1):COLOUR0:COLOUR131
:VDU28,33,9,38,7,4,12:PRINTTAB(4*box%,0)
CHR$146;:go%=ABS(box%-1):PRINTTAB(4*go%,
2)CHR$146;
 1080 ENDPROC
 1090 DEFPROCsh
 1100 PROCl(2,3,0,39,0,8):PROCl(2,3,0,79
1,0,8):PROCl(2,0,0,415,0,8)
 1110 PROCm("  SHUFFLING"):FORL%=0TORND(
9):FORN%=0TO51:M%=RND(51-N%)+N%-1:T%=d%?
N%:d%?N%=d%?M%:d%?M%=T%:NEXT:NEXT:FORN%=
0TO51:p%?N%=d%?N%:NEXT:PROCso(0,4):PROCs
o(6,10):PROCm("")
 1120 ENDPROC
 1130 DEFPROCso(n%,l%)
 1140 REPEAT:F%=TRUE:FORN%=n%TOl%:IFp%?N
%MOD13>p%?(N%+1)MOD13 T%=p%?N%:p%?N%=p%?
(N%+1):p%?(N%+1)=T%:F%=FALSE
 1150 NEXT:UNTILF%
 1160 ENDPROC
 1170 DEFPROCd
 1180 PROCl(0,5,0,39,6,8):PROCn(0,130,0,
n$):PROCl(1,3,0,791,0,8)
 1190 ENDPROC
 1200 DEFPROCdi
 1210 PROCm("SELECTING BOX"):H%=0:J%=0:F
ORN%=0TO14:RESTORE(3000+N%*10):READA%,B%
,C%,D%,E%,F%:h%?0=p%?A%MOD13+1:h%?1=p%?B
%MOD13+1:h%?2=p%?C%MOD13+1:h%?3=p%?D%MOD
13+1
 1220 V%=FNs4:IFV%>H% H%=V%:J%=1:hi%?1=N
% ELSEIFV%=H% J%=J%+1:hi%?J%=N%
 1230 NEXT:IFJ%=1N%=1ELSEN%=RND(J%)
 1240 RESTORE(3000+(hi%?N%*10)):READA%,B
%,C%,D%,k%?0,k%?1:PROCro(12,4)
 1250 ENDPROC
 1260 DEFFNs4
 1270 S%=0:F%=FALSE:PROCss(p%?A%DIV13,p%
?B%DIV13,p%?C%DIV13,p%?D%DIV13,4,TRUE):P
ROCfs(1):PROCth:PROCps(6):PROCr4(1):PROC
r3(4):PROCft(3,4,6,FALSE)
 1280 =S%
 1290 DEFFNs5(n%,f%):F%=FALSE
 1300 S%=0:PROChs(n%):PROCj:PROCss(s%?0,
s%?1,s%?2,s%?3,s%?4,f%):PROCso(n%,n%+3):
PROChs(n%):F%=FALSE:PROCfs(5):PROCpp:PRO
Cps(10):PROCr5:PROCr4(5):PROCr3(10):PROC
ft(4,10,10,TRUE)
 1310 =S%
 1320 DEFPROChs(n%)
 1330 FORN%=n%TOn%+4:h%?(N%-n%)=p%?N%MOD
13+1:s%?(N%-n%)=p%?N%DIV13
 1340 NEXT:ENDPROC
 1350 DEFPROCj
 1360 FORN%=0TO3:IFh%?N%=11ANDs%?N%=s%?4
 S%=1
 1370 NEXT:ENDPROC
 1380 DEFPROCfs(n%)
 1390 RESTORE3150:FORL%=1TOn%:READW%,X%,
Y%,Z%:IFh%?W%=h%?X%ANDh%?X%=h%?Y%ANDh%?Y
%=h%?Z% S%=S%+12:F%=TRUE
 1400 NEXT:ENDPROC
 1410 DEFPROCth
 1420 IFF% ENDPROC ELSE RESTORE3160:FORL
%=1TO4:READX%,Y%,Z%:IFh%?X%=h%?Y%ANDh%?Y
%=h%?Z% S%=S%+6:F%=TRUE
 1430 NEXT:ENDPROC
 1440 DEFPROCps(n%)
 1450 IFF% F%=FALSE:ENDPROC ELSERESTORE3
170:FORL%=1TOn%:READX%,Y%:IFh%?X%=h%?Y%
S%=S%+2:F%=TRUE
 1460 NEXT:F%=FALSE:ENDPROC
 1470 DEFPROCss(a%,b%,c%,d%,e%,f%)
 1480 IFF% ENDPROC ELSET%=a%=b%ANDb%=c%A
NDc%=d%:IFf%ELSEIFT%ANDd%=e% S%=S%+5:END
PROC ELSEENDPROC
 1490 IFT% S%=S%+4:IFd%=e% S%=S%+1
 1500 ENDPROC
```

```
 1510 DEFPROCr5
 1520 F%=FALSE:IFh%?4-h%?3=1ANDh%?3-h%?2
=1ANDh%?2-h%?1=1ANDh%?1-h%?0=1 S%=S%+5:F
%=TRUE
 1530 ENDPROC
 1540 DEFPROCr4(n%)
 1550 IFF% ENDPROC ELSERESTORE3150:FORL%
=1TOn%:READW%,X%,Y%,Z%:IFh%?Z%-h%?Y%=1AN
Dh%?Y%-h%?X%=1ANDh%?X%-h%?W%=1 S%=S%+4:F
%=TRUE
 1560 NEXT:ENDPROC
 1570 DEFPROCr3(n%)
 1580 IFF% ENDPROC ELSERESTORE3160:FORL%
=1TOn%:READX%,Y%,Z%:IFh%?Z%-h%?Y%=1ANDh%
?Y%-h%?X%=1 S%=S%+3
 1590 NEXT:ENDPROC
 1600 DEFPROCpp
 1610 IFF% ENDPROC ELSERESTORE3180:FORL%
=1TO10:READV%,W%,X%,Y%,Z%:IFh%?V%=h%?W%A
NDh%?W%=h%?X% S%=S%+6:F%=TRUE:IFh%?Y%=h%
?Z% S%=S%+2
 1620 NEXT:ENDPROC
 1630 DEFPROCft(a%,b%,c%,d%)
 1640 PROCvs(a%):PROCa(a%):IFd% PROCg4
 1650 PROCg3(b%):PROCg2(c%)
 1660 ENDPROC
 1670 DEFPROCvs(n%)
 1680 FORL%=0TOn%:IFh%?L%>10 h%?L%=10
 1690 NEXT:ENDPROC
 1700 DEFPROCa(n%)
 1710 T%=0:FORL%=0TOn%:T%=T%+h%?L%:NEXT:
IFT%=15 S%=S%+2
 1720 ENDPROC
 1730 DEFPROCg4
 1740 RESTORE3150:FORL%=1TO5:READW%,X%,Y
%,Z%:IFh%?W%+h%?X%+h%?Y%+h%?Z%=15 S%=S%+
2
 1750 NEXT:ENDPROC
 1760 DEFPROCg3(n%)
 1770 RESTORE3160:FORL%=1TOn%:READX%,Y%,
Z%:IFh%?X%+h%?Y%+h%?Z%=15 S%=S%+2
 1780 NEXT:ENDPROC
 1790 DEFPROCg2(n%)
 1800 RESTORE3170:FORL%=1TOn%:READX%,Y%:
IFh%?X%+h%?Y%=15 S%=S%+2
 1810 NEXT:ENDPROC
 1820 DEFPROCl(f%,n%,x%,y%,c%,i%)
 1830 FORN%=x%TOx%+n%:xx%=32+N%*192+384-
48*i%:VDU24,xx%;y%;xx%+159;y%+224;5:IFf%
=0PROCfu(xx%,y%,p%?(c%+N%))ELSEIFf%=1PRO
Cfd(xx%,y%)ELSEIFf%=2GCOL0,130:CLG
 1840 NEXT:ENDPROC
 1850 DEFPROCfd(x%,y%)
 1860 GCOL0,1:GCOL0,131:CLG:MOVEx%+16,y%
+208:FORL%=1TO6:VDU128,128,128,128,10,8,
8,8,8:NEXT:ENDPROC
 1870 DEFPROCfu(x%,y%,cd%)
 1880 LOCALt%,c%,s%:c%=cd%MOD13+1:s%=cd%
DIV13:GCOL0,s%MOD2:GCOL0,131:CLG:s%=129+
s%*4:t%=-((c%=1)*65+(c%=10)*145+(c%=11)*
74+(c%=12)*81+(c%=13)*75+(c%>1ANDc%<10)*
(c%+48))
 1890 MOVEx%+12,y%+216:VDUt%:MOVEx%+52,y
%+140:VDUs%,s%+1,10,8,8,s%+2,s%+3:MOVEx%
+120,y%+36:VDUt%:GCOL0,2:MOVEx%,y%:DRAWx
%,y%+224
 1900 ENDPROC
 1910 DEFPROCsb
 1920 PROCm("YOU TO DISCARD"):N%=0:VDU28
,0,31,39,4:COLOUR0:REPEAT:*FX21
 1930 K%=GET-49:IFK%>-1ANDK%<6ANDN%<2 PR
OCb
 1940 IFK%=78ANDN%>0 PROCe
 1950 UNTILK%=-36ANDN%=2:k%?0=k%?0+6:k%?
1=k%?1+6:PROCm(""):PROCro(14,10):PROCl(2
,1,4,39,0,8):PROCl(0,3,0,39,6,8):PROCn(0
,130,0,MID$(n$,1,27)+STRING$(7," "))
 1960 ENDPROC
 1970 DEFPROCkt
 1980 PROCl(0,0,0,415,16,8):IFp%?16MOD13
+1=11 @%=2:S%=2:go%=box%:COLOUR131:COLOU
R0:VDU28,27,21,38,21,4,12,7,23,1,0;0;0;0
;:PRINT"JACK CUT=2";:K%=INKEY300:PRINT:P
ROCsc:go%=ABS(box%-1)
 1990 ENDPROC
 2000 DEFPROCm(w$)
 2010 COLOUR131:COLOUR1:VDU28,25,19,38,1
7,4,23,1,0;0;0;0;:CLS:PRINTw$;
 2020 ENDPROC
 2030 DEFPROCb
 2040 IFN%=1ANDk%?0=K% ENDPROC ELSEk%?N%
=K%:N%=N%+1:PROCn(2,128,K%*6+3,STR$(K%+1
))
 2050 ENDPROC
 2060 DEFPROCe
 2070 N%=N%-1:PROCn(0,130,k%?N%*6+3,STR$
(k%?N%+1))
 2080 ENDPROC
 2090 DEFPROCn(f%,b%,x%,n$)
 2100 VDU28,0,31,39,31,4,23,1,0;0;0;0;:C
OLOURf%:COLOURb%:PRINTTAB(x%,0)n$;
 2110 ENDPROC
 2120 DEFPROCro(d%,e%)
 2130 FORN%=0TO1:p%?(d%+N%)=p%?k%?N%:NEX
T:IFk%?0>k%?1 T%=k%?0:k%?0=k%?1:k%?1=T%
 2140 FORN%=0TO1:FORL%=k%?N%TOe%:p%?L%=p
%?(L%+1):NEXT:k%?1=k%?1-1:NEXT
 2150 ENDPROC
 2160 DEFPROCpl
 2170 IFG%ENDPROC ELSEP%=-1:diff%=1:same
%=1:T%=0:M%=3:Y%=4:I%=FALSE:U%=FALSE:FOR
N%=0TO9:p%?(N%+20)=p%?N%:NEXT:p%?24=p%?1
6:p%?30=p%?16:REPEAT:IFgo%PROCy ELSEPROC
o
```

```
 2180 UNTILG%OR(Y%=0ANDM%=-1)
 2190 ENDPROC
 2200 DEFPROCy
 2210 IFU%go%=0:ENDPROC ELSEC%=FNc
 2220 IFC%<>5ELSEU%=TRUE:IFI%S%=1:PROCsc
:PROCx:go%=0:ENDPROC ELSEgo%=0:ENDPROC
 2230 PROCu:p%?C%=&FF:Y%=Y%-1:IFM%=-1I%=
TRUE:IFY%=0ANDO% S%=1:PROCsc:PROCx
 2240 IFNOTG%go%=0
 2250 ENDPROC
 2260 DEFPROCo
 2270 IFI%go%=1:ENDPROC
 2280 IFM%>-1ELSEIFU%S%=1:PROCsc:PROCx:g
o%=1:ENDPROC ELSEgo%=1:ENDPROC
 2290 IFT%=0C%=M%:PROCq:ENDPROC
 2300 I%=TRUE:FORN%=0TOM%:IFT%+FNv<32I%=
FALSE
 2310 NEXT:IFNOTI%ELSEIFU%S%=1:PROCsc:PR
OCx:go%=1:ENDPROC ELSEgo%=1:ENDPROC
 2320 IFM%=0C%=0:PROCq:ENDPROC
 2330 FORN%=0TOM%:IFFNs NEXT:PROCq:ENDPR
OC ELSE NEXT
 2340 FORN%=0TOM%:IFFNr NEXT:PROCq:ENDPR
OC ELSE NEXT
 2350 FORN%=0TOM%:IFFNt NEXT:PROCq:ENDPR
OC ELSE NEXT
 2360 FORN%=0TOM%:IFFNf NEXT:PROCq:ENDPR
OC ELSE NEXT
 2370 FORN%=0TOM%:IFFNp NEXT:PROCq:ENDPR
OC ELSE NEXT
 2380 N%=M%:REPEAT:IFT%+FNv<32 C%=N%:N%=
0
 2390 N%=N%-1:UNTILN%=-1:PROCq:ENDPROC
 2400 DEFFNs
 2410 IFp%?N%MOD13+1=R%?P%AND(same%=3ORs
ame%=2)ANDT%+FNv<32 C%=N%:N%=M%:=TRUE EL
SE =FALSE
 2420 DEFFNr
 2430 IFdiff%>1ANDABS((p%?N%MOD13+1)-R%?
P%)=1ANDT%+FNv<32 C%=N%:N%=M%:=TRUE ELSE
 =FALSE
 2440 DEFFNt
 2450 IFT%+FNv=31 C%=N%:N%=M%:=TRUE ELSE
 =FALSE
 2460 DEFFNf
 2470 IFT%+FNv=15 C%=N%:N%=M%:=TRUE ELSE
 =FALSE
 2480 DEFFNp
 2490 IFp%?N%MOD13+1=R%?P%ANDT%+FNv<32 C
%=N%:N%=M%:=TRUE ELSE =FALSE
 2500 DEFFNv
 2510 IFp%?N%=&FF =&FF
 2520 L%=p%?N%MOD13+1:IFL%>10 =10 ELSE =
L%
 2530 DEFFNc
 2540 IFY%=0=5ELSEPROCm(" YOU TO PLAY")
 2550 REPEAT:REPEAT:UNTILFNh:IFK%>-1PROC
n(2,128,K%*6+3,STR$(K%+1)):REPEAT:N%=GET
:UNTILN%=127ORN%=13 ELSEIFK%=-1N%=13
 2560 IFN%=127PROCn(0,130,K%*6+3,STR$(K%
+1))
 2570 UNTILN%=13:IFK%>-1PROCn(2,130,K%*6
+3," "):PROCl(2,0,K%,39,0,8)
 2580 PROCm(""):=K%+6
 2590 DEFFNh
 2600 K%=GET-49:IFK%=-1 =FNtr ELSEIFK%<-
1ORK%>3 =FALSE ELSEN%=K%+6:IFp%?N%=&FFOR
T%+FNv>31 =FALSE ELSE =TRUE
 2610 DEFFNtr
 2620 FORN%=6TO9:IFT%+FNv<32 N%=9:NEXT:=
FALSE
 2630 NEXT:=TRUE
 2640 DEFPROCq
 2650 PROCl(2,0,M%,791,0,8):PROCu:PROCz:
M%=M%-1:IFY%=0 U%=TRUE:IFM%=-1ANDO% S%=1
:PROCsc:PROCx
 2660 IFNOTG% go%=1
 2670 ENDPROC
 2680 DEFPROCu
 2690 P%=P%+1:R%?P%=p%?C%MOD13+1:PROCl(0
,0,1,415,C%-1,8-P%):S%=0:N%=C%:T%=T%+FNv
:IFT%=15S%=2
 2700 IFP% S%=S%+FNal+FNrn
 2710 IFT%=31 S%=S%+2
 2720 PROCrt:K%=INKEY300:PROCsc:IFT%=31
O%=FALSE:PROCx
 2730 ENDPROC
 2740 DEFPROCx
 2750 COLOUR130:VDU28,7,19,22,12,4:CLS:V
DU28,28,21,36,21:COLOUR131:CLS:I%=FALSE:
U%=FALSE:T%=0:P%=-1:same%=1:diff%=1
 2760 ENDPROC
 2770 DEFPROCz
 2780 IFC%=M% ENDPROC ELSE FORN%=C%TOM%-
1:p%?N%=p%?(N%+1):NEXT
 2790 ENDPROC
 2800 DEFFNal
 2810 N%=(R%?P%=R%?(P%-1)):IFsame%=4ORNO
TN% same%=1:=0ELSEIFsame%=3ANDN% same%=4
:=12ELSEIFsame%=2ANDN% same%=3:=6ELSEsam
e%=2:=2
 2820 DEFFNrn
 2830 N%=(ABS(R%?P%-R%?(P%-1))=1):IFNOTN
%diff%=1:=0ELSEIFdiff%=1diff%=2:=0ELSEIF
R%?P%<>R%?(P%-2)diff%=diff%+1:=diff%ELSE
diff%=2:=0
 2840 DEFPROCsc
 2850 @%=3:O%=TRUE:IFS%=0ENDPROC ELSECOL
OUR0:COLOUR131:VDU28,31,11,38,11,4:sc%?g
o%=sc%?go%+S%:PRINTTAB(go%*4,0)sc%?go%;:
IFsc%?go%>=121 G%=TRUE
 2860 ENDPROC
 2870 DEFPROCrt
 2880 @%=2:COLOUR131:COLOUR0:VDU28,28,21
```

```
,37,21,4,12,7,23,1,0;0;0;0;:PRINTT%;" FO
R ";S%;
 2890 ENDPROC
 2900 DEFPROCk:COLOUR0:COLOUR131:VDU28,2
5,19,38,19,4:PRINT"PRESS ANY KEY";:REPEA
TUNTILGET:CLS:ENDPROC
 2910 DEFPROCch
 2920 IFG%ENDPROC ELSEPROCl(0,3,0,791,20
,8):PROCl(0,3,0,39,26,8):go%=box%:FORJ%=
1TO2:go%=ABS(go%-1):S%=FNs5(20+go%*6,TRU
E):PROCm(w$(go%)+" SCORE "+STR$(S%)):PRO
Ck:PROCsc:IFG% J%=2:NEXT:ENDPROC ELSE NE
XT
 2930 PROCl(2,3,0,791,0,8):PROCl(2,3,0,3
9,0,8):PROCl(0,3,0,39*box%+791*ABS(box%-
1),12,8):S%=FNs5(12,FALSE):PROCm(" BOX S
CORE="+STR$(S%)):PROCk:go%=box%:PROCsc
 2940 ENDPROC
 2950 DEFPROCw
 2960 @%=3:PROCm("  "+w$(go%)+" WIN"+STR
ING$(19," ")+"ANOTHER Y/N?"):g%(go%)=g%(
go%)+1:COLOUR0:COLOUR131:VDU28,31,13,38,
13:PRINTTAB(go%*4,0)g%(go%);
 2970 ENDPROC
 2980 DEFFNnm
 2990 REPEAT:K%=GETAND&5F:UNTILK%=78ORK%
=89:IFK%=89 VDU28,31,11,38,11:PRINT"
   ":=FALSE ELSEPROCm("OK,BYE FOR NOW")
:=TRUE
 3000 DATA0,1,2,3,4,5
 3010 DATA0,1,2,4,3,5
 3020 DATA0,1,2,5,3,4
 3030 DATA0,1,3,4,2,5
 3040 DATA0,1,3,5,2,4
 3050 DATA0,1,4,5,2,3
 3060 DATA0,2,3,4,1,5
 3070 DATA0,2,3,5,1,4
 3080 DATA0,2,4,5,1,3
 3090 DATA0,3,4,5,1,2
 3100 DATA1,2,3,4,0,5
 3110 DATA1,2,3,5,0,4
 3120 DATA1,2,4,5,0,3
 3130 DATA1,3,4,5,0,2
 3140 DATA2,3,4,5,0,1
 3150 DATA0,1,2,3,0,1,2,4,0,1,3,4,0,2,3,
4,1,2,3,4
 3160 DATA0,1,2,0,1,3,0,2,3,1,2,3,0,1,4,
0,2,4,0,3,4,1,2,4,1,3,4,2,3,4
 3170 DATA0,1,0,2,0,3,1,2,1,3,2,3,0,4,1,
4,2,4,3,4
 3180 DATA0,1,2,3,4,0,1,3,2,4,0,2,3,1,4,
1,2,3,0,4,0,1,4,2,3,0,2,4,1,3,0,3,4,1,2,
1,2,4,0,3,1,3,4,0,2,2,3,4,0,1
```

ADVERTISING IN BEEBUG

For advertising details, please contact

Sarah Shrive

on (0727) 40303 or FAX: (0727) 60263

or write to:

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

TECHNICAL SUPPORT

A free technical help service is available exclusively for members between Monday and Friday from 9.00am to 1.00pm and 2.00pm to 4.30pm. As this is a free service please be ready to quote your membership number.

Unfortunately, no technical support is available on Saturdays and we regret that we are not able to offer advice on programming problems.

For telephone enquiries specifically on programs published in the magazine please ask for the 'magazine department'.

Enquiries can also be made in writing to our normal address, and an SAE for the reply is appreciated.

RISC USER

The Archimedes Support Group

Our Risc User magazine is now in its second volume and is enjoying the largest circulation of any magazine devoted to the Archimedes. The list of members seeking support from the Risc User group is growing steadily and as well as private individuals includes schools, colleges, universities and industry and government establishments.

Existing Beebug members, interested in the new range of Acorn micros, may either transfer their membership to the new magazine or extend their subscription to include both magazines. A joint subscription will enable you to keep completely up-to-date with all innovations and the latest information from Acorn and other suppliers on the complete range of BBC micros. RISC User has a massive amount to offer, particularly at this time while documentation on the Archimedes and RISC OS is still limited.

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

RISC OS
A series on the new Archimedes operating system, including writing multi-tasking programs.

ACORN LAUNCH THE A 3000
The new BBC microcomputer.

PARTIAL RENUMBER UTILITY
A very useful machine code utility for renumbering parts of a Basic program.

MOUSE POINTER DEFINER
An easy to use utility for designing any style of mouse-controlled pointer.

A MULTI-TASKING NOTE-PAD
A comprehensive RISC OS multi-tasking application which offers a multi-page, multi-file notepad with editing and printing facilities.

THE R140 UNIX WORKSTATION
A look at Acorn's entry into the business market.

ARC PROCEDURE LIBRARY

RISC USER TOOLBOX
A comprehensive toolbox module for the Archimedes.

MOVIE MAKER
An exciting full-feature application for creating a wide variety of animated sequences and cartoons.

MOUSE MENU
A general purpose WIMP based menu, which is fully RISC OS compatible.

PIXEL EDITOR
A full screen pixel editor with 3 zoom levels, cut, copy and paste facilities.

ARCHIMEDES VISUALS
Some stunning visual effects.

MAKING THE MOST OF LOGISTIX
How to use this spreadsheet.

IMAGE RESCALER
A fast utility which can rescale screen images to any proportion.

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

As a member of BEEBUG you may extend your subscription to include RISC User for only £8.50 (overseas see below).

Name:...
Memb No:...
Address: ....
....
....
....

SUBSCRIPTION DETAILS

Destination	Additional Cost
UK,BFPO &Ch Is	£ 8.50
Rest of Europe and Eire	£13.00
Middle East	£15.00
Americas and Africa	£17.00
Elsewhere	£19.00

*I wish to receive both BEEBUG and **RISC User**.* *I enclose a cheque for £ .... or alternatively*
I authorise you to debit myACCESS/Visa/Connect account: ______/ ______/ ______/ ______/
Signed: .... *Expiry Date:* _____/ _____/

***Send to:** RISC User, Dolphin Place, Holywell Hill, St Albans, Herts AL1 1EX, or telephone (0727) 40303*

UK Bulletin Boards

THE MIDLANDS

AIX 386 BBS 24hrs;	Worcester (0905) 754127 & 52538 V21,V23,V22,V22bis Subscription
Academics 24hrs	Birmingham 021-705 9677 V21,V23
Central BBS 24hrs	Birmingham 021-711 1451 V21,V23,V22,V22bis Courier HST 9600 baud; PEP; MNP error correction; UK Echomall coordinator
Chrono's Lair 24hrs	Birmingham 021-744 5581 V21,V23,V22,V22bis
Corrupt Computing 24hrs	Coventry (0203) 76831 V21,V23
Digital Matrix 24hrs	Birmingham 021-705 5187 V21,V23,V22,V22bis
Exchange Centre 24hrs	(0787) 50511 V21,V23,V22,V22bis
Intel Ace 24hrs	Oundle (0832) 73003 V23
The Junction 24hrs 08:00-23:59 00:00-07:59	Crewe (0270) 580099 V21,V23,V22,V22bis V21,V22,V22bis, Courier HST 9600 baud
Murdoch's Hangout 24hrs	Birmingham 021-711 2620 V22,V22bis
MWCFE *24hrs*	Leamington Spa (0926) 21844 *V21,V23,V22,V22bis*
Project Stourbridge 24hrs	(0384) 401770 V21,V23,V22,V22bis
Sherwood Forest BBS 24hrs	Nottingham (0802) 397 113 V21,V23,V22,V22bis
Trinity 4 24hrs	Leamington Spa (0926) 28294 V21,V23,V22,V22bis
TUG II 24hrs	Droitwich (0905) 775191 V21,V22,V22bis Courier HST 9600 baud; MNP error correction; Tandy

THE NORTH EAST

Consett Wildoati 24hrs	Consett (0207) 503870 V21,V22,V22bis; Courier HST 9600 baud; MNP error correction
Forum 80 MF: 7pm-11pm WE: 1pm-11pm	Hull (0482) 859 169 V21,V23
Hamnet MF: 6pm-8am WE: 24hrs	Hull (0482) 485 150 V21; Radio Hams
Kirklee ITeC 24hrs	Batley, Yorks (0924) 442598 V23, Information Technol-ogy Centre
Leoonfield RCPM 24hrs	(0964) 550745 V21,V22
LEMS Wildcati 24hrs	Leeds (0532) 600749 V21,V23
Log On Tyne Fido 24hrs	Tyneside 091-477 3339 V21,V23,V22,V22bis
Merlin 24hrs	Bradford (0274) 573481 V21,V23,V22,V22bis
N.Yorks Opus 24hrs	Knaresborough (0423) 888085 V21,V23,V22,V22bis
Poacher CBCS 24hrs	Grantham (0476) 62450 V21,V23,V22,V22bis; MNP error correction
Stockton QBBS 24hrs	Stockton (0642) 588989 V21,V22

POSTBAG

TIME SERIES FOR THE TUBE

Published programs often do not work on a co-processor, not for any fundamental reason, but because the author may simply not be aware of what is required. If like me you need the memory capacity and speed of the co-processor to handle large volumes of data, this can be a major annoyance. A case in point is the otherwise very useful time series program published in BEEBUG Vol.6 No.4. This can be made to work on the co-processor by changing just three lines, to eliminate illegal and unnecessary zero-page references, and to save the screen correctly. They are as follows:

```
  110 oscli=&FFF7:DIM command 40,title 80
,xlabel 70,ylabel 22:p%=42:S%=0
  120 DIM data(99),cusum(99):flag=0:filen
ame$="NO DATA":param=FALSE:lones=1
 2320 *SAVE SCREEN FFFF30000+5000
```

Paul Holmes

We do not now test programs separately for tube or co-processor compatibility. If a program runs on a model B or Master, it is clearly only necessary to switch the co-processor off (if fitted) to have a working program. In many cases the co-processor adds little to the program, as most of the programs which we publish will have been designed to run on a standard model B or Master anyway. Indeed, in some cases (particularly games) the extra speed can render a program unusable.

The example Mr.Holmes quotes is unusual in that a co-processor would allow the program to handle more data. However, in view of the now diminished interest in co-processors, we are unlikely to make any further changes to our current policy in this respect.

DESIGNED FOR THE MASTER

BEEBUG's Design program predates the Master and does not appear to work properly on that machine. I eventually found the cure myself in the form of the *Convert* program supplied on the Master Welcome disc, and now Design seems to operate in all respects as intended - the problems with user-defined characters and shading in bar-chart mode have disappeared.

I have also found that *Convert* fixes other programs originally written for the model B where character definitions are loaded directly into memory instead of being created 'legally' with VDU23 statements.

I suppose that the reason why I overlooked the *Convert* routine is that my Welcome disc seldom moves from the back of the box, but other readers may find this information helpful.

B.Hickman

There is more on the Welcome disc than most users realise (see this month's hints), and many of the files are quite useful. We hope to give more information about other programs in future issues of BEEBUG.

SORTING IN A FOREIGN LANGUAGE

I have never found anything on alphabetical sorting in foreign languages in British or American computer magazines or books. These are the ones we usually buy and read here in Iceland (and elsewhere).

You did start on a matter close to this in the First Course article in Vol.7 No.5, showing how to sort any character string into a pre-defined order using masks. How about foreign names and other words using the upper half of the ASCII code?

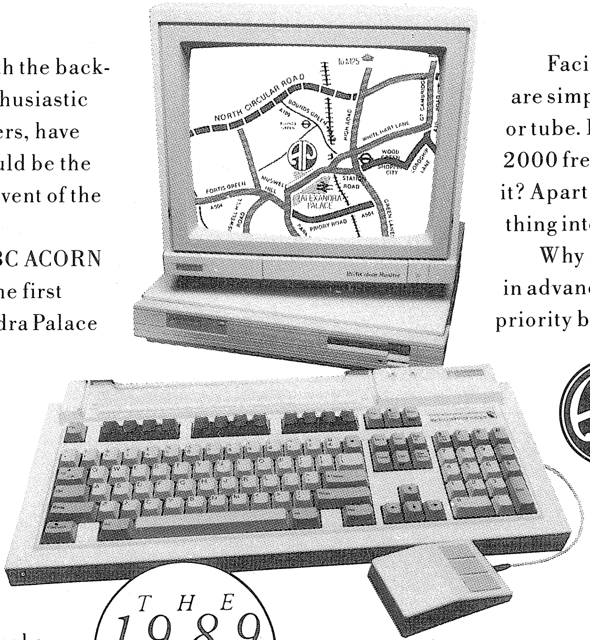
I need a fast sort routine to arrange strings in alphabetical order according to any alphabet or language in use. Can you help?

Johannes Thordarson

Any sort routine will do the job. The only modification is to replace the > or < operator normally used in the comparison part of the routine with an equivalent FNgreaterthan or FNlessthan function. This compares two strings character by character using a mask to determine the correct order. A possible definition of such a function might be:

```
 1000 DEF FNgreaterthan(a$,b$,mask$)
 1010 LOCAL a1$,b1$,f$,f1,f2,i:i=0
 1020 IF LEN(b$)<LEN(a$) f$=a$:a$=b$:b$=
f$:f1=TRUE ELSE f1=FALSE
 1030 REPEAT:i=i+1
 1040 a1$=MID$(a$,i,1):b1$=MID$(b$,i,1)
 1050 UNTIL a1$<>b1$ OR i=LEN(a$)
 1060 f2=(INSTR(mask$,a1$)>INSTR(mask$,b
1$))
 1070 IF f1 THEN =NOT f2 ELSE =f2
```

BBC Acorn User Show



BBC ACORN USER, with the backing of the BBC and the enthusiastic support of Acorn Computers, have together planned what could be the most exciting computing event of the year.

The new, all-action BBC ACORN USER SHOW is back, at the first home of BBC TV – Alexandra Palace featuring:-

- new product launches
- new software
- massive computing exhibition
- informative seminars
- technical clinics
- workshops
- demonstrations

All this and more will make the new BBC ACORN USER SHOW a real must for everyone interested in computers and their applications.

There are daily competitions, free draws with fabulous prizes to be won and celebrity guest appearances.

THE 1989 BBC ACORN USER show

Facilities at the tastefully restored Alexandra Palace are simply superb and it's so easy to get to by road, rail or tube. If you're driving you'll be glad to know there are 2000 free car parking spaces. So why not make a day of it? Apart from the beautiful grounds you'll find something interesting and exciting around every corner.

Why not save time and money by booking your ticket in advance? Cut the coupon now and ensure your priority booking.

ALEXANDRA PALACE · LONDON
Friday July 21st 3pm-9pm
Saturday July 22nd 10am-6pm
Sunday July 23rd 10am-6pm

BOOK NOW AND SAVE

Advance Tickets: Adults £2.50 Under 16s £1.50

At the door: Adults £3.50 Under 16s £2.50

Please send me ______ Adult Tickets at £2.50 each

Please send me ______ Under 16s Tickets at £1.50 each.

I enclose cheque/PO for £______ payable to SAFESELL Ltd., Market House, Cross Road, Tadworth, Surrey KT20 5SR.

DON'T SEND CASH.

Name ______

Address ______

______

______

______

RUI

HINTS and tips

This month's hints and tips were edited by Mike Williams. We pay £5 for each hint published and £15 for the star hint.

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

FONTS ON THE MASTER

J.R.Barker

The Master Welcome disc provides three files called 7by8, thin and italic which provide alternative screen character fonts. Each file contains the machine code to load it, thus:

```
    *DIR LIBRARY
    *thin
```

will load that font. The thin font is particularly useful in mode 2. The same fonts, with characters 160-176 replaced by icons are used in the art program TimPaint on the Welcome disc.

The standard font, and any of the three extra ones, can be modified using the font designer program Chardes, also on the Welcome disc. Fonts produced by Chardes can be saved and reloaded just like the others (see also BEEBUG Vol.5 No.6 pages 46 & 47).

LOADING DATA FROM A LIBRARY DIRECTORY

William Hine

The *<filename> facility on BBC micros is very powerful and useful when combined with the *LIB command on disc filing systems. It gives easy and automatic access to your own library of machine code programs no matter in which directory you are currently working. This is particularly helpful if you use the ADFS.

The library directory can be on any disc, not necessarily the one you are currently working with if you have a dual-drive system. The library is set with:

```
    *LIB <library>
```

where <library> is the name of your library directory. If you attempt to load and run a program (e.g. *Utils) which cannot be found in the current directory, the software will automatically search the library directory for it.

The same technique can also be adapted to the loading of data such as function key definitions, and user-defined character sets. To do this you need to find an 'execution address' which will persuade the filing system to return directly to Basic once it has loaded your data, instead of attempting to run it. To find a suitable address, run this short program and note the result for future reference:

```
    10 I=&BFFF
    20 REPEAT:I=I+1:UNTIL ?I=&60
    30 PRINT ~I
```

Now select the library directory and use this form of the *SAVE command, but replacing 'EEEE' by the four digit result from the program:

```
    *SAVE keydefs B00+100 EEEE
    *SAVE charset C00+100 EEEE
```

Once the library has been set, you need only enter *keydefs or *charset to reload the relevant files.

This will not work on a Master 128 which uses private RAM for these functions.

USING SPELLMASTER TO HANDLE SIDEWAYS RAM

Ian Waugh

If you have SpellMaster installed in your Master, you may find it more convenient to use the *DLOAD and *DSAVE commands to load and save banks of sideways RAM, rather than the commands built into the Master. For example, to set up a RAM buffer for printing, use:

```
    *DLOAD Buffer 5
```

TAPE-TO-DISC ON A MASTER

John Woodthorpe

The tape-to-disc transfer program supplied as part of the BEEBUG Discmaster collection was written before the Master appeared and does not work on that machine. However, the changes are quite small. In line 540, change STR$~!&2F8 to STR$~!&2F7 and change line 760 to:

```
    760 *ADFS
```

if you are transferring tape to ADFS.

FINDING THE SCREEN MODE

Peter Cumberland

If a program needs to know the screen mode currently in use, this is stored at &355. In Basic, it is possible to write a short function to return this:

```
    DEF FNmode=?&355
```

but this takes no account of whether shadow memory is in use (on a Master for example), and thus returns a value in the range 0 to 7.

Personal Ads

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

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

Master 512 £400, ADT £20, ADI £15, ARA2 £8, Commstar2 £8. Tel. (06285) 27046.

Master 128, Micro Vitec Med Res. monitor, twin D/S D/D Cumana drives with PSU, Epson LX80 printer, manuals vol 1 and 2 £650 o.n.o. Tel. (09904) 3025.

Magic Modem and BEEBUG Command ROM £55, AMX Design ROM and manual (boxed and unused) £60, 40T D/S disc drive £60. Tel. (048 93) 2812 after 6pm.

WANTED: Teletext adaptor and ATS ROM with PSU. Tel. (048 93) 2812 after 6pm.

BBC B OS 1.2, Cumana Dual D/S 40/80T D/D with PSU, Wordwise ROM, disc and tape software, computer magazines, manuals leads etc £350. Tel. (0905) 52855.

Archimedes 305 with Archimedes monochrome monitor and graphic wrtier (Clares) £600. Tel. Southampton 773258 office hours or Southampton 671840 after 8pm.

BBC Master, dual D/D, Interword, AMX Stop Press and MkIII mouse, Eprom cartridge, Eprom programmer, Joystick, Mini Office, 7 books, 30 blank discs. (new £1100) sell for £595. Printer £100. Tel. Frodsham 32993 after 8pm.

14" Medium res. metal case Microvitec colour monitor £135. Also can anyone for a fee, Arc quantise a photo? Tel. (0236) 20199 day, (0286) 833541 eves.

BBC B issue 4, with DFS + software £250 o.n.o. Also lots of ROMs and other peripherals and software. Send s.a.e to 7 Wieland Road, Northwood, Middx, HA6 3RD.

ATPL ROM board £15 o.n.o. Watford shadow RAM board £10, both for BBC B. Tel. (0689) 70263 eves.

BBC B with solidisk DFDC (8271 & 1770), ATPL board inc 16k swRAM, 32k swRAM, PMS 6502 second processor, Music 500, Acorn teletext adaptor, Viglen PC case, twin D/S 80T drives, Epson FX80, Microvitec 1451 med. res. monitor. Offers? will split (0384) 396739 eves and w/e or (0384) 291901 mon-fri 9.00-5.00. 823073.

BBC B issue 4 with DFS, Toolkit ROM, Exmon II, Wordwise + and Solidisk complete with relevant manuals £175 o.n.o. Hitachi AMS 3" D/D + 6 discs containing various software £25 o.n.o. Brother EP44 portable typewriter/printer w.p. functions uses thermal paper or ribbon with single sheet/paper roll. Mains transformer/battery powered £125 o.n.o. computing books for sale, "Structured BASIC" £3.75 "The BBC Micro- an Expert Guide" £3.50 "Assembly Language Programming for the BBC Micro £4.00 Acorn Lisp cassette and booklet £5.00.

Model B issue 7 with Solidisk DD DFS, 200k DSDD, Solidisk 256k ROM/RAM board, Wordwise +, Toolkit +, Replay, Proword. All in Viglen PC console. Also Philips green monitor and some software, cost over £900 accept £400 o.n.o. Tel. Sheffield 553418.

ESM Screenprint ROM £20, Vines Replay £15, Watford Mk2 Lightpen + penpal disc £12, Supersoft 'Word Perfect' £5, Acornsoft View Printer Driver £4, Micro Aid Supercalc cassette (transfers to disc) £3, Book "Forth on the BBC Micro" £3, all complete with relevant manuals. Digimouse plus Grafik software £15. Tel. (0403) 55400.

BBC Master 512k twin Watford disc drives, Archimedes monitor, mouse, Epson RX80 FT+ printer, many utilities including Interword, Sheet, Chart, Spellmaster, boxed games and manuals £800. Tel. (0935)

BBC 32k computer, View Word Processor, Disc Doctor and DNFS, Taxan KP810 printer, Cumana 40/80 disc drive plus relevant manuals and software on disc £450. Viewstore £25, Viewsheet £20, Viewchart £7, Dabhand view and Dabhand Viewsheet/store £6 each, Viewstore User Guide £5, Into View £2, BEEBUG's Toolkit £10, Magscan and Magscan Update £8, Filer Disc £3, Bound vols. 1 to 6 + vol 7 unbound £25, Magazine discs £2 each, Acorn User USERdump £10, Calligraphy £7, Magazine tapes £1 each, Morley EpROM Programmer V2 £15, 20 single sided single/double density discs £6.50 postage extra. Tel. (0705) 731859.

BBC B twin D/D, modem, mouse, data recorder, 32k expansion board, business progs inc: Interword, Busicalc, Delta card-index, Disc Doctor. Many leisure progs inc: Revs 4 tracks, Elite, over 60 blank discs in storage case £350 o.n.o. Tel. (0933) 625313.

Master 128 turbo complete with reference manuals 1,2 and Advanced, View and Viewsheet, Sideways RAM guides, Philips 14" med res. colour monitor with green screen text mode. Watford CD800S dual 40/80T D/S D/D with PSU £650 o.r.o. Tel. (0933) 316050 after 6pm.

512 DOS+ co processor with DOS+ 2.1, welcome guide, GEM and mouse £80, Viewstore database £30. Intersheet spreadsheet £30, Watford Quest Paint & mouse £35 Tel. (0933) 316050 after 6pm.

Mitsubishi 40/80T switchable D/S D/D justbeen serviced, report sent with drive £60, Voltmace Delta 14b joystick BBC B/ Master 128 as new, boxed £10, 50 Reflex D/S D/D 5.25" (96TPI) discs, new and unformatted in plastic disc box, labels, lock tabs. £40. Tel 01 494 1365 (office hours only).

For Sale; Printwise, Spellcheck III, Wordease, Discmaster, Quickcalc, Forth cassette, Printmaster Epson, Toolkit, ROMit, Exmon II, T/D ROM, few 2764 EpROMs (new), Gould switched mode PSU. For more information. Tel. (0254) 706127 after 11am each day.

Shinwa CP-80 dot matrix printer (Epson compatible) £99, BBC Master double plinth (unused) £10, Supersoft 'Busicalc' spreadsheet disc (for BBC B or Master) £5, 'The Hobbit' adventure disc (for BBC B) £5, 'Brian Clough's Football Fortunes' cassette (for BBC or Electron) £5. Tel. (0344) 482826 eves.

Taxan KP915 136 column NLQ dot matrix printer with cable for BBC. Epson/IBM compatible, as new condition (about 5hrs use) £275. Tel. (0242) 36690.

Z88 computer £175, 32k RAM pack £15, 128k RAM pack £30, mains adaptor £8 or £200 the lot. Tel. (0734) 784897 or (0734) 771230.

Model B software many games and adventures on tape and disc, all originals £2 each. Tel. (0734) 784897 or (0734) 771230.

BBC B with TORCH ZDP Users pack, fitted with both the Acorn Basic chip and the Torch Z80 for CPM switchable on the side of the machine, Users Guide, Sanyo Green Hi Res monitor, Twin 360 drives with PSU, RAM ROM Sideways ROM box with the following ROM software titles installed; 2 blank RAM chips for sideways RAM, Volex Teletext, Solidisk ADFS, Solidisk DFS, Commstar, Wordwise Plus, Interchart, Printmaster, Solidisk Toolkit, BEEBUG Toolkit, MCP CPM chip, all manuals are included for each piece of software. Perfect Calc, Perfect Filer, Perfect Writer, Pace modem, Volex kits for teletext, all in working order, more software than shown, £250 plus £75 if Epson FX80 printer is required. Tel. Mr T Warren (0702) 431076 for more information.

512 users,

Solve the compatibility problems of your MASTER 512 or BBC with co-pro adaptor using DOS+ Problem Solver

It corrects hardware incompatibilities (such as programmable interrupts' speed, low level keyboard scanning, etc.) and operating system's bugs, enabling the 512 board to run most IBM-PC programs that otherwise wouldn't run.

Lots of programs like Cat, Jet, Digger, ARTWORX's Strip Poker, ELECTRONIC ART's Golf, Driller, Dark Side, Impact, Charlie Chaplin, Test Drive, Infiltrator, StarQuake, Bushido, Tennis, all versions of MicroSoft's Flight Simulator, Frogger, Osbit, 688 Atack Sub, Defender of the Crown, Quadralian, Yes Chancellor, Anciant Art of War, Adventure Writter, Dream House, AFT, Droege, Dream, Delux Paint 2, Fontasy 2.08, News, PC Tutor, Lotus 123 2.0, Turbo Calc, Mandelbrot Generator, Prospero Pascal, Turbo Pascal 4.0, Turbo C 1.5, Turbo Prolog, Prolog2, PC File+, Galaxy, Trendtex/2, Homebase 2.15, Mindreader, DBASE III plus, Pipedream, etc. will run like in an ordinary IBM PC.

The program provides the INS, DEL, PG-UP, PG-DOWN, HOME, END and SC-LOCK keys to users who don't have the numeric keypad.
It will also allow you to switch between colour and B&W modes, change the computer's speed and save the hi-res picture on the screen, during the execution of any program.

DOS+ Problem Solver works with all 80186 co-processor boards (with 512K or 1M bytes RAM) using DOS+ 1.2-XIOS 1.00, DOS+ 1.2-XIOS 1.01 or DOS+ 2.1-XIOS 1.03.
If you need any further informations about DOS+ Problem Solver, please contact Shibumi Soft.

PRICE: £24.95 inc. program, user manual & one year free user suport.
Price includes VAT. Please add £3 for postage and packing. Faulty disks will be replaced.
Cheques should be payable to Shibumi Soft.

Shibumi Soft: R. Prof. Camara Sinval, 138
4100 PORTO
PORTUGAL

Viglen model B Consul unit £30, AMX Stop Press (Master version) £30, Artist ROM £30, Quill Adventure Writing System £15, Watford's Dumpout 3 ROM £15, Watford's Print ROM £10, Clare's Replica III £8, BEEBUG's Sleuth ROM £20, BEEBUG's MuROM £10, Watford's TransfeROM £10, Penfriend 2 (WW+ Utility ROM) £15. Tel. (0734) 784897 or (0734) 771230.

BBC B, ROM RAM board, DFS twin D/S D/D, PMS56502 2nd processor, Digi mouse lots of software on ROM and disc. Tel. 01 388 0392 after 6pm with offer for all or part.

BEEBUG C including Stand Alone Generator £30, Wordwise Plus £15, Basic editor £10. Tel. (0202) 892904 after 6pm.

BBC B issue 7 with 32k RAM card and 12 ROM RAM board, and Wordwise plus, Interword, Intersheet, Vu-Calc, Novacad, and AMX art ROMs, with all manuals and AMX mouse. Opus DDFS/twin disc drive (8 vols. of 31 files per disc side) £450. Also Taxan KP810 printer £100, and Taxan Supervision 625 colour monitor £130 all in excellent condition. Tel. (0737) 221352.

Acorn Electron c/w ROMbox Plus, Joystick interface; Starword; EXP ROM 2.0; data recorder and computer desk, all leads and manuals, as new £100.Tel. 031 445 1802.

WANTED: Computer Concepts MEGA 3 ROM and BEEBUG Master ROM, manuals etc must be included. Tel. (0253) 67987.

BEEBUG MEMBERSHIP

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

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

BACK ISSUE PRICES (per issue)

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

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

POST AND PACKING

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

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

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

BEEBUG MAGAZINE is produced by BEEBUG Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Technical Editor: David Spencer
Technical Assistant: Glynn Clements
Advertising: Sarah Shrive
Production Assistant: Sheila Stoneman
Membership secretary: Mandy Mileham
Managing Editor: Sheridan Williams

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

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

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

In all communication, please quote your membership number.

BEEBUG Ltd (c) 1989
ISSN - 0263 - 7561
Printed by Newnorth-Burt Ltd (0234) 41111

Magazine Disc/Cassette

JUNE 1989 DISC/CASSETTE CONTENTS

FOREIGN LANGUAGE TESTER - test your knowledge of foreign languages with this program, plus character definers for French, German, Greek and Turkish, and sample vocabularies in French.

BASIC LINE EDITOR - a utility for Basic programmers allowing any line in a program to be easily edited.

FAST DFS BACKUP - speed up the backup of your DFS formatted discs with this intelligent utility.

FILE HANDLING FOR ALL (Part 12) - a utility for database developers allowing individual bytes in a file to be changed at will.

AN EXTENDED DISASSEMBLER - a comprehensive disassembler for the 6502, 65C02 and Rockwell versions - source code and ROM image both included.

FIRST COURSE - a program to demonstrate the use of mode 7 colour graphics control for large digits.

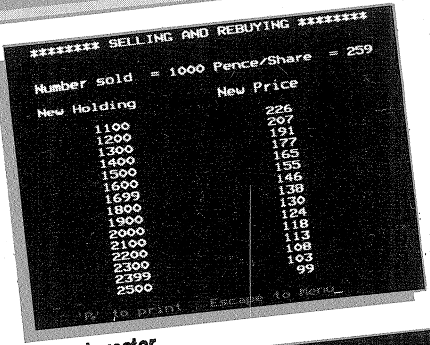
A CLOCK FACE FOR THE MASTER - a short program to provide a working clock face on a Master.

WORKSHOP - a general purpose disc access ROM in source code format.

SHARE INVESTOR - complete program for investors incorporating the latest features described in this issue.

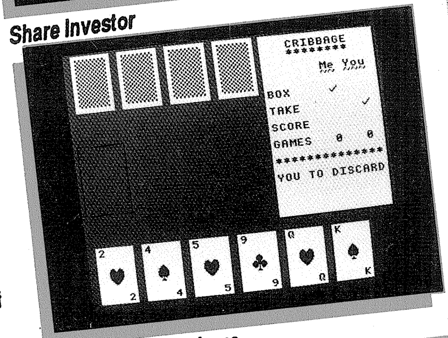
THE GAME OF CRIBBAGE - two programs which together provide an excellent implementation of the traditional card game, Cribbage, with the computer playing one of the two hands.

MAGSCAN - bibliography for this issue (Vol.8 No.2).



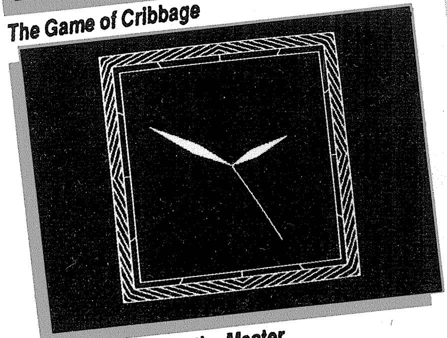
```
******** SELLING AND REBUYING ********
Number sold  = 1000 Pence/Share  = 259
                      New Price
New Holding
                         226
                         207
        1100             191
        1200             177
        1300             165
        1400             155
        1500             146
        1600             138
        1699             130
        1800             124
        1900             118
        2000             113
        2100             108
        2200             103
        2300              99
        2399
        2500
  P to print   Escape to Menu_
```

Share Investor



```
CRIBBAGE
********
         Me  You
BOX       ✓
TAKE          ✓
SCORE     0   0
GAMES     0   0
**************
YOU TO DISCARD
```

The Game of Cribbage



A Clock Face for the Master

All this for £3. 50 (cassette), £4.75 (5" & 3.5" disc) + 60p p&p (30p for each additional item).
Back issues (5.25" disc since Vol.3 No.1, 3.5" disc since Vol.5 No.1, tapes since Vol.1 No.10) available at the same prices.

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

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

Cassette subscriptions can be commuted to a 5.25" or 3.5" disc subscription on receipt of £1.70 per issue of the subscription left to run. All subscriptions and individual orders to:
BEEBUG, Dolphin Place, Holywell Hill, St.Albans, Herts. AL1 1EX.

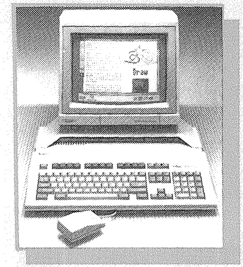
Acorn

Archimedes

BEEBUG the Leading Archimedes Specialists

BEEBUG

Acorn's NEW A3000 Computer



ACORN A3000 SERIES		
0255G	A3000 Entry System	649.00
0256G	A3000 Colour System	869.00
ARCHIMEDES 300 SERIES		
0193G	310 Entry System	899.00
0195G	310 Colour System	1119.00
0257G	310M Entry System	959.00
0259G	310M Colour System	1179.00
ARCHIMEDES 400 SERIES		
0260G	410/1 Entry System	1199.00
0261G	410/1 Colour System	1419.00
0262G	420/1 Entry System	1699.00
0264G	420/1 Colour System	1919.00
0275G	440/1 Entry System	2499.00
0276G	440/1 Colour System	2719.00
0263G	440 Entry System	1929.00
0265G	440 Colour System	2149.00

PRICES EXCLUDE V.A.T.

BEEBUG offers you:

Technical Support Department
We don't lose interest as soon as you've made your purchase. You can always telephone our Technical Support Department or Showroom for some friendly and impartial advice.

Archimedes Magazine - RISC USER
Subscription is just £14.50 a year (UK), for this 56 page magazine packed full of useful information, programs, news, hints and tips etc.

Repairs and Service
If things do go wrong, you can be confident of our repair and service department. We will get your computer repaired as soon as possible, and return it to you free of charge, for a full year from the date of purchase.

Price
Our prices are competitive, but if you see a better current deal from another Acorn dealer, we will be pleased to match it. In that way you can be sure of the best price as well as outstanding support and service.

Speedy Service
We have a showroom in St Albans where you can see and try any of the systems. They are always on display and in stock. Our mail order service is also very efficient, with over 80% of orders going out the same day. 48 hour delivery charges on a complete system are just £7 and 24 hour delivery £11.50.

Payment and Finance
We accept cheques, Connect, Access and Visa, and are able to offer various credit options including 0% finance over 9 months or 13.75% flat rate (typical APR 28%) over 12, 24 or 36 months.

Archimedes Price Guarantee

We will match any current price or offer available from any other reputable Official Acorn Dealer.

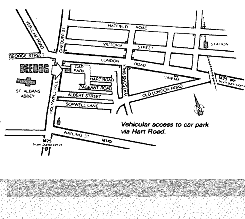
MEMBERS SPECIAL OFFER - WHEN BUYING AN ARCHIMEDES SYSTEM WE WILL GIVE YOU ABSOLUTELY FREE:
PC Emulator, Printer Lead, Ten 3.5" discs & Lockable disc box. In addition with 410s you will receive 1st Word Plus and with 420s & 440s, the Software Developer's Toolbox.
OR
A choice of goods up to 10 % of the value of your Archimedes.
Please enquire for details of member's offers with 0% finance.

FURTHER DETAILS

Please telephone for more information on:

- 0% Finance over 9 months
- 12/24/36 Month Finance
- Trade-ins for Archimedes

Finance arranged within 48 hours and all orders for Archimedes despatched within 24 hours.



BEEBUG Ltd

Dolphin Place, Holywell Hill, St. Albans, Herts., AL1 1EX
Opposite Clarks Camera Shop on Holywell Hill
Showroom open
9am - 5.30 pm Mon - Sat Thursday late 'til 8pm
☎ (0727) 40303 Fax (0727) 60263
(Answering machine from 5pm)