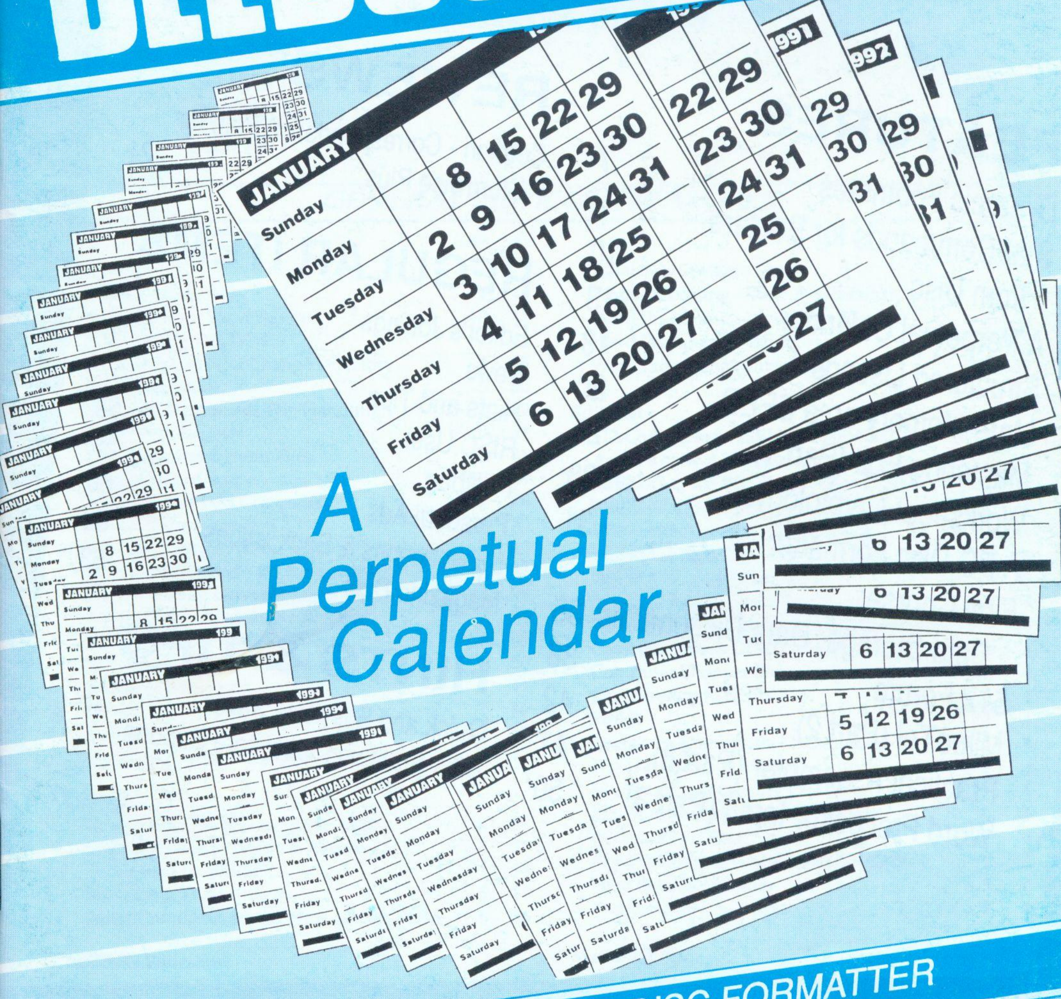
Vol.9 No.8 January/February 1991

BEEBUG

FOR THE BBC MICRO & MASTER SERIES



- PC DISC FORMATTER
- DYNAMIC FOOTNOTES
- MASTERING EDIT
- STRUCTURED LISTINGS

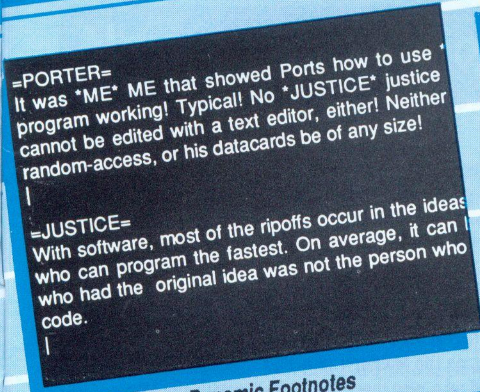
BEEBUG

Vol.9 No.8 January/February 1991

PROGRAM INFORMATION

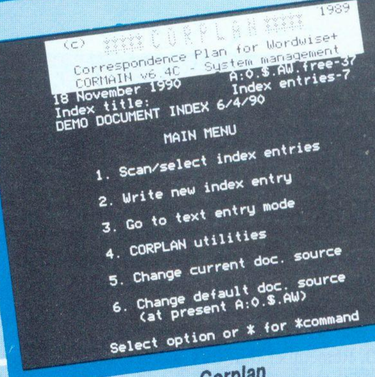
All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly 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



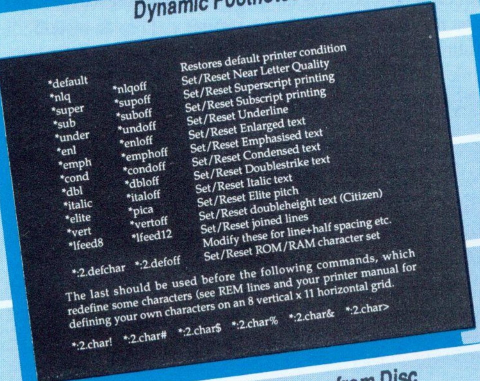
```
=PORTER=
It was *ME* ME that showed Ports how to use
program working! Typical! No *JUSTICE* justice
cannot be edited with a text editor, either! Neither
random-access, or his datacards be of any size!
|

=JUSTICE=
With software, most of the ripoffs occur in the ideas
who can program the fastest. On average, it can
who had the  original idea was not the person who
code.
|
```

Dynamic Footnotes



```
(c)        CORPLAN         1989
                Plan for Wordwise+
Correspondence Plan for Wordwise+
                 System management
CORMAIN v6.4C -          A:0.$.AW.free-37
18 November 1990         Index entries-7
Index title:
DEMO DOCUMENT INDEX 6/4/90

            MAIN MENU

   1. Scan/select index entries

   2. Write new index entry

   3. Go to text entry mode

   4. CORPLAN utilities

   5. Change current doc. source

   6. Change default doc. source
      (at present A:0.$.AW)

   Select option or * for *command
```

Corplan

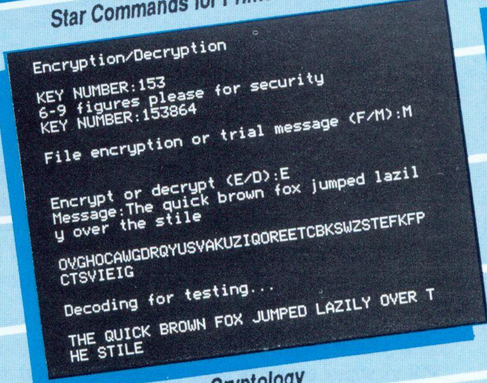


```
                             Restores default printer condition
*default     *nlqoff         Set/Reset Near Letter Quality
*nlq         *supoff         Set/Reset Superscript printing
*super       *suboff         Set/Reset Subscript printing
*sub         *undoff         Set/Reset Underline
*under       *enloff         Set/Reset Enlarged text
*enl         *emphoff        Set/Reset Emphasised text
*emph        *condoff        Set/Reset Condensed text
*cond        *dbloff         Set/Reset Doublestrike text
*dbl         *italoff        Set/Reset Italic text
*italic      *pica           Set/Reset Elite pitch
*elite       *vertoff        Set/Reset doubleheight text (Citizen)
*vert        *lfeed12        Set/Reset joined lines
*lfeed8                      Modify these for line+half spacing etc.
                             Set/Reset ROM/RAM character set
*:2.defchar  *:2.defoff
The last should be used before the following commands, which
redefine some characters (see REM lines and your printer manual for
defining your own characters on an 8 vertical x 11 horizontal grid.
*:2.char!  *:2.char#  *:2.char$  *:2.char%  *:2.char&  *:2.char>
```

Star Commands for Printers from Disc

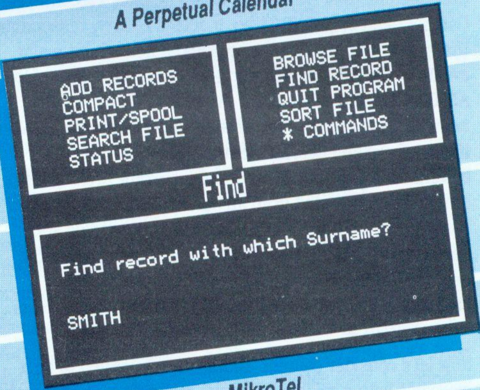
```
                          January

 M    T    W   Th    F    S   Su
                              7
 1    2    3    4    5    6   14
 8    9   10   11   12   13   21
15   16   17   18   19   20   28
22   23   24   25   26   27
29   30   31
```

A Perpetual Calendar



```
Encryption/Decryption

KEY NUMBER:153
6-9 figures please for security
KEY NUMBER:153864

File encryption or trial message (F/M):M


Encrypt or decrypt (E/D):E
Message:The quick brown fox jumped lazil
y over the stile

OVGHOCAWGDRQYUSVAKUZIQOREETCBKSWZSTEFKFP
CTSVIEIG

Decoding for testing...

THE QUICK BROWN FOX JUMPED LAZILY OVER T
HE STILE
```

Cryptology



```
ADD RECORDS        BROWSE FILE
COMPACT            FIND RECORD
PRINT/SPOOL        QUIT PROGRAM
SEARCH FILE        SORT FILE
STATUS             * COMMANDS

              Find

Find record with which Surname?

SMITH
```

MikroTel

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.
M	Program is for Master 128 and Compact only.

Editor's Jottings

Welcome to 1991, the year which will see the start of volume 10 of BEEBUG. Remember that this is one of our two month issues (for both January and February), and so the next issue will be that for March 1991, which we expect to send out towards the end of February.

BBC MICROS, THE MASTER SERIES AND THE ARCHIMEDES

From time to time I receive comments to the effect that 'these days BEEBUG seems to be all about the Master and the Archimedes', or 'BEEBUG has far too many games', and similar statements. So what does BEEBUG stand for?

First of all, BEEBUG is devoted to users of the BBC micro (model B etc.) and the Master series (Master 128, Master Compact and so on). Even so, it seems entirely reasonable that we should publish the occasional item about major events in the Archimedes world (such as the review of the A540 in the November issue - Vol.9 No.6), and smaller news items about the Archimedes where appropriate. After all, these are still Acorn machines (the A3000 is designated a *BBC micro*), and many current Archimedes owners have migrated from an earlier BBC micro (and no doubt there are many more who may choose to do so). Furthermore, the Archimedes is a significant part of the Acorn world these days, and news of Acorn and its products is likely to be of interest to all BBC micro users.

Now let's consider content. BEEBUG is intended to be a magazine with a relatively serious and informed approach to the use of BBC micros. Some readers who have been subscribers from the early days may recall that there was a time when we regularly published TWO games programs in every issue. Over the years that has diminished, and a game program is no longer a regular feature of the magazine. In any case, I prefer to think in terms of 'leisure' software, programs which entertain, inform, fascinate etc, though they may not be games as such. I feel that in this broader sense there is a good argument for publishing software in this category.

As far as the Master is concerned, our own surveys show that probably some 50% of readers nowadays have a Master series machine, and this proportion is likely to be growing, as the Master 128 is still manufactured by Acorn and sold as a new machine. In fact, when the Master 128 was first launched, we had a regular eight-page section entirely devoted to that model. That was dropped some time ago, but I feel that one or two items per month which are specifically for the Master range is fully justified.

We try to devote a major part of each issue to useful and worthwhile applications and utilities, and it is in these programs that I believe the strength of BEEBUG lies. Some readers ask, 'why can we not have more pages of reviews?' Well I for one agree, but the number of new products for the Beeb is now comparatively small. We aim to cover what does appear (see the Corplan review in this issue), but we have also tackled this area via the surveys which we have started and which we shall be continuing to publish.

The readers of any magazine no doubt have a wide variety of interests and tastes. It is unlikely that any publication can always satisfy fully the requirements of all readers all the time. We try to provide a variety of material in each issue which we believe will broadly appeal to most BEEBUG readers. If you feel that the magazine does not reflect your needs then do let us know - we might be able to do something about it - but also take an objective look at two or three issues and consider whether your criticisms really are justified. After all there are other readers to consider.

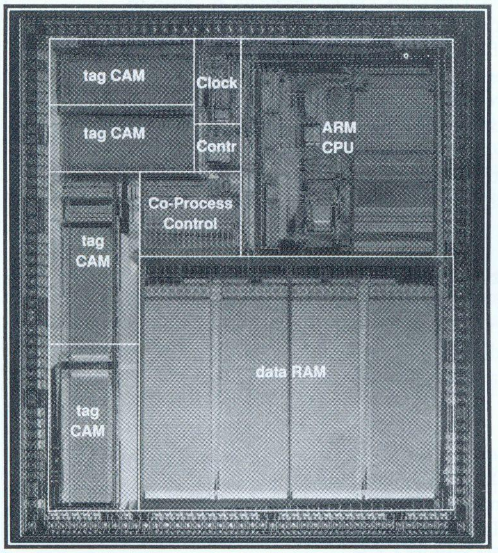
FILE HANDLING

We are very pleased with our first venture into book publishing, *File Handling for All* - see advertisement in this issue. This book provides information and examples suitable for all BBC micro owners from near beginners to more advanced programmers. The special offer price on the combined book and disc is available only to BEEBUG (and RISC User) members.
M.W.

News News News News News News

ADVANCED RISC MACHINES

In a move which saw Acorn's share price (not usually noted for its interest) rocket up by over 100% in a matter of days, the company announced on 27th November a new joint venture with Apple Computer (producers of the highly-regarded Macintosh) and VLSI (the manufacturers of Acorn's ARM chip set). A new company called *Advanced RISC Machines Ltd* (or ARM Ltd.), to be based in Cambridge, has been set up with equal shares owned by Apple and Acorn, and a smaller proportion by VLSI. The new company will be responsible for all future design and development of Acorn's RISC technology, and it is expected that most of the design team at Acorn will transfer to the new enterprise.



The VL86C020 Acorn RISC Microcomputer (ARM3), manufactured and marketed by VLSI Technology

The aim is intended to provide much better opportunities for Acorn's RISC chip set to compete in world markets, where Acorn (via VLSI) is already equal top in the league for RISC chip deliveries, alongside SPARC. Acorn itself will henceforth concentrate on systems design and manufacture (i.e. by developing its Archimedes range), leaving further development of the ARM processor to the new company. Apple refused to comment about its own future potential use of RISC, but it would appear that the company is looking at the market for embedded control devices, and mobile communications where the high performance and low power consumption of RISC is ideally suited.

The new company has already stated that its first new product will be the ARM600, to be released this year, though no further details were given. However, despite its growing importance, this latest in the ARM series seems unlikely to feature an on-board floating point processor, an area in which Acorn acknowledges the current ARM3 is weak.

Until ARM Ltd moves to new premises, all enquiries should be directed to Acorn Computers, Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN, tel. (0223) 245200.

MUSIC FOR SPECIAL NEEDS AND PRIMARY

The Beeb continues to be widely used for music applications, and no small part of this must be due to Hybrid Technology which continues to support the BBC micro with its Music 5000 and associated software.

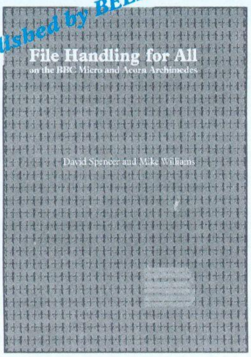
Hybrid has now published two new packages to support music education for young and handicapped children. *Soundshow* provides over an hour of pre-recorded music for listening exercises, and is particularly aimed at users whose handicaps prevent ready access to non-computerised music (such as a record collection). The package uses the concept of a musical slide show to represent each item of music pictorially, allowing it to be selected, played, stopped, or repeated under control of a single switch or touch screen. The package can be customised by the teacher to build up pop-music style charts of individual children's selections as an album of personal favourites.

The second new package, *Soundscore*, helps all children to compose music using a 'graphics score', a visual representation of the pitch and length of each note without the complexities of the customary musical staff. Music played on a musical keyboard is recorded as a scrolling colour graphics score on the computer, where it can then be stored, edited and replayed as required. Scores can also be printed out for reference and display.

Soundshow and *Soundscore*, for the model B or Master with a Music 5000 Synthesizer, each cost £29.00 plus VAT from Hybrid Technology Ltd, 273 The Science Park, Cambridge CB4 4WE, tel. (0223) 420360.

PROGRAMS FOR THE BBC MICRO

Micro-Aid is a company which still provides strong support for the BBC micro. Its various business programs (Cashbook, Account, Taxman etc. are well established as good-quality low-cost software. There is also a Family History System, Stockmarket and Cribbage games, plus multi-lingual Hangman and other educational software. For a copy of its latest catalogue contact Micro-Aid at 1 Kildonan Courtyard, Barrhill, S.Ayrshire, KA26 0PS, Scotland, tel. (0465) 82288.


Published by BEEBUG
File Handling for All
on the BBC Micro and Acorn Archimedes
David Spencer and Mike Williams

File Handling for All

on the BBC Micro and Acorn Archimedes

by David Spencer and Mike Williams

Computers are often used for file handling applications yet this is a subject which computer users find difficult when it comes to developing their own programs. This book *File Handling for All* aims to change that by providing an extensive and comprehensive introduction to the writing of file handling programs with particular reference to Basic.

File Handling for All, written by highly experienced authors and programmers David Spencer and Mike Williams, offers 144 pages of text supported by many useful program listings. It is aimed at Basic programmers, beginners and advanced users, and anybody interested in File handling and Databases on the Beeb and the Arc. However, all the file handling concepts discussed are relevant to most computer systems, making this a suitable introduction to the file handling for all.

The book starts with an introduction to the basic principles of file handling, and in the following chapters develops an in-depth look at the handling of different types of files e.g. serial files, indexed files, direct access files, and searching and sorting. A separate chapter is devoted to hierarchical and relational database design, and the book concludes with a chapter of practical advice on how best to develop file handling programs.

The topics covered by the book include:

Card Index Files	Serial Files
File Headers	Disc and Record Buffering
Using Pointers	Indexing Files
Searching Techniques	Hashing Functions
Sorting Methods	File System Calls
Networking Conflicts	Testing and Debugging

The associated disc contains complete working programs based on the routines described in the book and a copy of Filer, a full-feature Database program originally published in BEEBUG magazine.

File Handling for All Book	**£ 9.95**	Stock Code	**BK02b**
The disc supporting the book is	**£ 4.75**	Stock Code	**BK05a** (5.25" DFS 40/80T disc) or **BK07a** (3.5" ADFS disc)

SPECIAL OFFER to BEEBUG members *File Handling for All* book and disc **£ 11.95**
Stock Codes **BK04b** (5.25" DFS 40/80T disc) or **BK06b** (3.5" ADFS disc)

BEEBUG Ltd, 117 Hatfield Rd, St.Albans, Herts AL1 4JS. Tel (0727) 40303 Fax (0727) 860263

Dynamic Footnotes

Give added interest to text displays with Stephen Sexton's dynamic footnote system.

The purpose of the system described here is to enable keywords to be highlighted within a text display, such that selecting a keyword will result in a further display of text related to the chosen word. The system is quite powerful yet flexible, allowing footnotes to reference further footnotes, and so on, while different keyword labels can relate to the same footnote if required. As such the system has many applications from simple footnoting of articles, through appendices, diary systems, adventure games and much more.

The program given here as Listing 1 (fNOTES) uses simple ASCII text files (created with a word processor or text editor) as source - along with an index file for the keywords. By necessity, it requires disc files, as it uses random access techniques, or possibly a RAM disc for instant access (the best of all options).

Text needs to be entered in a special format (described below), such that when viewed, certain words are highlighted. Selecting a keyword then reveals a further text display which may itself have further keywords embedded in it. The program acts like a card index, with each card having pointers to other cards.

The fNOTES program requires each text file to have an index, and this is created by the program CREATE - see Listing 2.

CREATING FOOTNOTE FILES

This can be done with any word processor or text editor, but do make sure you do not include any embedded commands or other special characters. The text file structure is as follows (see figure 1 for part of a sample text file layout):

First line: Filename of the index file (to be created with Create and used as the index when fNOTES is run.

Then, the following format is used (pressing Return at the end of each line):

```
=KEYWORD=
text text text text text text *keyword2*
label text text text text *keyword3* label
text text text text text text text \ \
text text text etc etc
............................
until the end of the text area, then
|
=NEXT KEYWORD=
............................
```

and so on. Each definition of a keyword area is marked in the text by a prefix and suffix of '=' for a normal text area; or '==' for the start text area, and the title. The start text area can be anywhere within the main text area, but for ease of text creation, I usually put it at the start.

```
=PORTER=
It was *ME* ME that showed Ports how to use *PTR* PTR# to get his
program working! Typical! No *JUSTICE* justice  in the world! His data
cannot be edited with a text editor, either! Neither can his program totally
random-access, or his datacards be of any size!
|
=JUSTICE=
With software, most of the ripoffs occur in the ideas, then it's a race to see
who can program the fastest. On average, it can be said that the person
who had the  original idea was not the person who got the royalties for the
code.
|
```

Sample text showing title format

No linefeeds will be issued during text output, owing to the way the program works, so the character '\' should be used to force a new line. To put a blank line between paragraphs, '\ \ ', or:

`\<Return>\<Return>`

can be used. Spaces and Returns are not seen as different.

A reference to a keyword area is indicated by putting a single '*' character at the beginning and end of the keyword, (e.g. *keyword*), and the next word in the text will then also be highlighted. The keyword itself is not displayed. This means that keyword areas can be referenced by names different from the keyword itself, and that different highlights can point to the same keyword area of text. In this way, a great deal of control can be exercised over the selectable areas.

When keywords are highlighted, there is a potential problem if the keyword is followed by a punctuation symbol (e.g. a comma, full stop, etc.), as this punctuation will also be highlighted if it is included with the word. Punctuation can be treated as the next word by separating it from the preceding (highlighted) word with a space, and it won't then be highlighted. For example:

As an *INTRO* introduction, blah blah etc.

will result in the word "introduction," (including the comma) being highlighted. However, using:

As an *INTRO* introduction , blah blah etc.

will result in "introduction" only being highlighted.

Note: keywords and titles must not contain spaces (i.e. they must be single words). However, in the title, the character @ will act as a space - for example:

```
==1990@-@A@New@Decade==
```

will read as "1990 - A New Decade" in the title window.

CREATING AN INDEX

Once the text file has been created and saved, it needs to be indexed using the CREATE program. When this is run it simply prompts for the name of your text file.

USING THE FOOTNOTES PROGRAM

Once you have created an index to your footnotes file, you are ready to use fNOTES to browse through the text. The program prompts for the name of the text file to use. The start of the text will then be displayed in a three-window format described below, with any keyword labels highlighted.

The program is based around three windows: the top one is the title window, centred on the screen according to the length of the data title. The bottom one is the keyword selection window, and the main window is the text display window. Text is left-justified when displayed.

When a file is being interrogated, Space flips through the current keyword areas available for reading, Return selects a keyword and displays the related text, and Tab goes back to the start text area, (i.e. it quits a scan and begins a new one).

Note that each time a data file is changed, or edited, a new index file must be created for it, or the fNOTES program will start outputting text from the wrong position, and maybe even keyword labels and other garbage.

As listed, the program allows for a maximum of 100 keywords, and 20 keyword markers per keyword area. However, moving the program down in memory on a model B, or running it on a Master, would allow these limits to be expanded because more memory is then available. These limits can be changed in the DIM statements near the start of the listing.

Listing 1

```
   10 REM Program footNOTES
   20 REM Version B1.8
   30 REM Author  Stephen Sexton
   40 REM BEEBUG  Jan/Feb 1991
   50 REM Program subject to copyright
   60 :
  100 MODE0:VDU23,1,0;0;0;0;
  110 hS$=CHR$17+CHR$129+CHR$17+CHR$0
  120 hE$=CHR$17+CHR$128+CHR$17+CHR$1
  130 ON ERROR GOTO1580
  140 DIM mark%(100),mark$(100),cm$(1,20
)
  150 INPUT"File:"f$:CLS
  160 A%=OPENIN(f$):B%=OPENIN(FNgetword)
  170 mark%(0)=-1:E%=FALSE:Z%=0:REPEAT
  180 IF EOF#B% E%=TRUE
  190 IF NOT E% Z%=Z%+1:INPUT#B%,mark$(Z
%),mark%(Z%)
  200 UNTIL E%:N%=Z%:CLOSE#B%:A%=OPENIN(
f$)
  210 PROCheadings:area%=1:PROCwindow(5,
5,75,26):PROCwindow(3,28,23,30)
  220 REPEAT PTR#A%=mark%(area%)
  230 PROClayout:PROCselect:UNTIL FALSE
  240 END
  250 :
 1000 DEF FNgetword:Z$=""
 1010 REPEAT
 1020 Q%=BGET#A%:Z$=Z$+CHR$(Q%)
 1030 UNTIL Q%=13 OR Q%=32
 1040 =LEFT$(Z$,LEN(Z$)-1)
 1050 :
 1060 DEF PROClayout
 1070 VDU28,5,26,75,5:CLS:M%=0
 1080 REPEAT:key%=FALSE
 1090 test$=FNgetword
 1100 IF FNkeyword("*",test$) THEN M%=M%
+1:cm$(0,M%)=test$:test$=FNgetword:cm$(1
,M%)=test$:key%=TRUE
```

```
1110 IF NOT FNkeyword("=",test$) AND NO
T FNkeyword("==",test$) AND test$<>"|" T
HEN PROCoutput(test$)
1120 UNTIL test$="|"
1130 PRINT':ENDPROC
1140 :
1150 DEF FNkeyword(Q$,Z$)=(LEFT$(Z$,LEN
(Q$))=Q$ AND RIGHT$(Z$,LEN(Q$))=Q$)
1160 :
1170 DEF PROCselect:VDU28,3,30,23,28
1180 L%=M%:E%=FALSE
1190 REPEAT
1200 IF M%=0 L%=0 ELSE L%=L%MODM%+1
1210 CLS:PRINT:IF M%=0 THEN L%=0:PROCce
ntre("New Scan",20) ELSE PROCcentre(hS$+
cm$(1,L%)+hE$,20)
1220 g%=GET:IF g%=13 OR g%=9 E%=TRUE
1230 UNTIL E%
1240 IF L%=0 area%=1 ELSE area%=FNgetar
ea(cm$(0,L%))
1250 IF g%=9 OR (g%=13 AND M%=0) area%=
1
1260 ENDPROC
1270 :
1280 DEF PROCcentre(Z$,Z%):PRINT TAB((Z
%-LEN(Z$)-(L%<>0)*8)/2);Z$:ENDPROC
1290 :
1300 DEF PROCheadings:L%=0
1310 REPEAT:L%=L%+1
1320 UNTIL FNkeyword("==",mark$(L%))
1330 title$=MID$(mark$(L%),3,LEN(mark$(
L%))-4):St%=(79-LEN(title$))/2
1340 PROCwindow(St%,1,LEN(title$)+1+St%
,3):PRINT'" ";title$:VDU26:ENDPROC
1350 :
1360 DEF FNgetarea(Z$):Q%=0
1370 REPEAT:Q%=Q%+1
1380 UNTIL MID$(mark$(Q%),2,LEN(mark$(Q
%))-2)=MID$(Z$,2,LEN(Z$)-2)
1390 =Q%
1400 :
1410 DEF FNpunct(Z$):Q%=ASC(Z$)
1420 IF ((Q%>32 AND Q%<48) OR (Q%>57 AN
D Q%<64)) IF LEN(Z$)<3 =TRUE
1430 =FALSE
1440 :
1450 DEF PROCwindow(l%,t%,r%,b%)
1460 L=l%*16-2:R=(r%+1)*16
1470 B=1020-(b%+1)*32:T=1024-t%*32
1480 GCOL0,129:VDU24,L;B;R;T;:CLG
1490 VDU28,l%,b%,r%,t%:CLS:ENDPROC
1500 :
1510 DEF PROCoutput(Z$)
1520 IF FNpunct(Z$) THEN PRINT Z$;" ";:
ENDPROC ELSE PRINT" ";
1530 IF LEFT$(Z$,1)="\" PRINT:ENDPROC
1540 IF POS+LEN(Z$)>70 PRINT
1550 IF key%=TRUE Z$=hS$+Z$+hE$
1560 PRINT Z$;:ENDPROC
1570 :
1580 CLOSE#0:L%=0
1590 VDU28,5,26,75,5:CLS:PRINTTAB(0,10)
;
1600 IF ERR=222 PROCcentre(hS$+"File no
t found - check datafiles!"+hE$,80):IF G
ET:RUN
1610 IF ERR=19 OR ERR=15 PROCcentre(hS$
+"Datafile is not correct - check wordin
g and markers!"+hE$,80):IF GET:RUN
1620 MODE7:IF ERR<>17 REPORT:PRINT" at
line ";ERL
1630 END
```

Listing 2

```
  10 REM Index creator
  20 REM Version B1.0
  30 REM Author Stephen Sexton
  40 REM BEEBUG Jan/Feb 1991
  50 REM Program subject to copyright
  60 :
 100 MODE7:INPUT"File:"f$
 110 PRINT'"Defined Labels"'
 120 A%=OPENIN(f$):B%=OPENOUT(FNgetword
(A%))
 130 E%=FALSE:REPEAT
 140 IF EOF#A% THEN E%=TRUE
 150 IF NOT E% THEN PROCcheck
 160 UNTIL E%:CLOSE#0:END
 170 :
1000 DEF FNgetword(in%)
1010 Z$="":REPEAT
1020 Q%=BGET#in%:Z$=Z$+CHR$(Q%)
1030 UNTIL Q%=13 OR Q%=32:=LEFT$(Z$,LEN
(Z$)-1)
1040 :
1050 DEF PROCoutput
1060 a$=FNscrub(a$):PRINT a$
1070 PRINT#B%,a$,P%:ENDPROC
1080 :
1090 DEF PROCcheck
1100 a$=FNgetword(A%)
1110 IF LEFT$(a$,1)="=" AND RIGHT$(a$,1
)="=" THEN P%=(PTR#A%)-LEN(a$)-1:PROCout
put
1120 ENDPROC
1130 :
1140 DEF FNscrub(Z$):Q3$=""
1150 FOR Z%=1 TO LEN(Z$)
1160 IF MID$(Z$,Z%,1)="@" THEN Q$=Q$+"
" ELSE Q$=Q$+MID$(Z$,Z%,1)
1170 NEXT:=Q$
```

Star Commands for Printers from Disc

Derek Baron explains how you can create a series of star commands on disc to control all the features of your printer.

Not all owners of BBC micros have invested in sideways RAM, and cannot therefore use the Enhanced Printer Buffer (BEEBUG Vol.6 No.10) with its set of commands to control printer output. I was asked by one such owner who had invested in Interword how could he include these commands on his system. My answer was to write a short program which generated as many star commands as he wanted and saved them onto disc. This way he was able to look up in his printer manual, just once, the relevant codes for different effects, and can now call them by name at the appropriate points in his Interword text. I have modified the program slightly to automatically generate several useful star commands for BEEBUG readers.

CREATING THE STAR COMMANDS

Since you may decide to use a large number of commands, start with a blank disc if possible. Type in the program, and save it before running it. The commands will be saved to disc automatically. Should you exceed the number of files that will fit on one side of the disc (as does the present program) include ':2.' before the extra command names (as here), both in the program and when you use them, and bracket the name in the DATA statement between quotes (assuming the use of double-sided drives).

The first instructions (lines 150 - 160) check whether your machine is currently printing, setting a flag if it is. If not, then after enabling the printer and sending the commands, printer output will be disabled (line 240).

The data statements from line 410 onwards are paired; the first statement includes the command name and the number of printer codes that will be sent; the second statement gives the actual codes in decimal. Printer manuals are often a bit obscure here, sometimes giving characters rather than numbers, or supplying hexadecimal numbers. To find the decimal ASCII number of a character e.g. @ use:

```
PRINT ASC"@"
```

This example gives the value 64. Note that Escape has a code of 27, thus 'ESC @' which is the code to restore an Epson compatible printer to its default state is '27,64' in decimal. These numbers are sometimes given in hexadecimal (as 1B, 40), so use the computer to do the conversion to decimal:

```
PRINT &1B,&40
```

Delete any paired lines you do not want and add others as required.

The commands created by the program (see Table 1) are for a Citizen 120D or 180E printer, but most will work with Epson compatibles. If you have a different printer remember you only have to look up the codes in your manual once!

The commands given in Table 1 are only a start. Add your own DATA lines for a backspace character (decimal 8) allowing you to put that acute accent ' over the 'e' just like the French do. Combine the defined characters given into one command (the program will accept up to 200 printer codes in one command). Set up your own defined characters or use printer graphics for fancy borders.

USING THE COMMANDS

All the printer star commands created can be used at any time that it is appropriate to enter such a command. They can be used from the keyboard in immediate mode, or in the command mode of software like View, Interword, etc., or within your own programs.

*default		Restores default printer condition
*nlq	*nlqoff	Set/Reset Near Letter Quality
*super	*supoff	Set/Reset Superscript printing
*sub	*suboff	Set/Reset Subscript printing
*under	*undoff	Set/Reset Underline
*enl	*enloff	Set/Reset Enlarged text
*emph	*emphoff	Set/Reset Emphasised text
*cond	*condoff	Set/Reset Condensed text
*dbl	*dbloff	Set/Reset Doublestrike text
*italic	*italoff	Set/Reset Italic text
*elite	*pica	Set/Reset Elite pitch
*vert	*vertoff	Set/Reset doubleheight text (Citizen)
*lfeed8	*lfeed12	Set/Reset joined lines
		Modify these for line+half spacing etc.
*:2.defchar	*:2.defoff	Set/Reset ROM/RAM character set

The last should be used before the following commands, which redefine some characters (see REM lines and your printer manual for defining your own characters on an 8 vertical x 11 horizontal grid.

*:2.char! *:2.char# *:2.char$ *:2.char% *:2.char& *:2.char>

Table 1. Star commands provided by the program for printer control

For example, ViewStore and ViewSheet owners can use **elite* and **cond* together (with a Citizen printer) from the command screen to give reports with a printer width of 160 characters, enabling much more information to be printed than the restrictive 80 characters.

Interword does have a facility for entering printer codes directly in the text, but it does mean having the printer manual handy at all times. Using star commands like **nlq* or **elite* is much friendlier. Type in some text (in Interword) and then move the cursor to the point at which the printer effect is to start. Press f1 and an 'Embedded commands' Menu appears. Two presses of the cursor down key moves to a line starting with a star '*': type in just the name of the command e.g. *super*

Press the Escape key to return to the Edit screen, and you should see that the character position is now in inverse video. Move the cursor to the next position where you wish an effect to appear, or where you wish to cancel an effect, press f1 again and enter the appropriate command e.g. *supoff*

Make sure that the disc with all your star commands is present in the currently selected drive and use *Print text* (option 6) from the main menu.

The commands may also be used from the Interword menu screen before printing, as can all star commands. Finally, do check your Epson compatible printer for Epson compatibility!

```
   10 REM Program StarCode
   20 REM Version B1.0
   30 REM Author  Derek Baron
   40 REM BEEBUG  Jan/Feb 1991
   50 REM Program subject to copyright
   60 :
  100 CLS:start=&900
  110 REPEAT
  120 FOR pass=0 TO 2 STEP 2
  130 P%=start
  140 [OPTpass
  150 LDA #117:JSR &FFF4
  160 TXA:AND #1:STA onflag
  170 LDA #2:JSR &FFEE
  180 LDX #1
  190 .loop
  200 LDA #1:JSR &FFEE
  210 LDA data,X:JSR &FFEE
  220 INX:CPX data:BNE loop
  230 LDA onflag:BNE end
  240 LDA #3:JSR &FFEE
  250 .end RTS
  260 .onflag BRK
  270 .data
```

```
 280 ]
 290 NEXTpass
 300 READ f$,total
 310 IF f$="END" GOTO380
 320 ?P%=total+1
 330 FOR I%=1 TO total
 340 READ numbers:I%?P%=numbers
 350 NEXT
 360 PRINT"Saving "f$
 370 OSCLI("*SAVE "+f$+" "+STR$~start+"
"+STR$~(P%+I%))
 380 UNTIL f$="END"
 390 END
 400 REM Command name, Number of number
s
 410 DATA default,2
 420 REM Numbers to send to Printer
 430 DATA 27,64
 440 DATA nlq,3
 450 DATA 27,120,1
 460 DATA nlqoff,3
 470 DATA 27,120,0
 480 DATA super,3
 490 DATA 27,83,48
 500 DATA supoff,2
 510 DATA 27,84
 520 DATA sub,3
 530 DATA 27,83,49
 540 DATA suboff,2
 550 DATA 27,84
 560 DATA under,3
 570 DATA 27,45,49
 580 DATA undoff,3
 590 DATA 27,45,48
 600 DATA enl,3
 610 DATA 27,87,49
 620 DATA enloff,3
 630 DATA 27,87,48
 640 DATA emph,2
 650 DATA 27,69
 660 DATA emphoff,2
 670 DATA 27,70
 680 DATA cond,1
 690 DATA 15
 700 DATA condoff,1
 710 DATA 18
 720 DATA dbl,2
 730 DATA 27,71
 740 DATA dbloff,2
 750 DATA 27,72
 760 DATA italic,2
 770 DATA 27,52
 780 DATA italoff,2
 790 DATA 27,53
 800 DATA elite,2
 810 DATA 27,77
 820 DATA pica,2
 830 DATA 27,80
 840 DATA vert,2
 850 DATA 27,104
 860 DATA vertoff,2
 870 DATA 27,117
 880 DATA lfeed8,3
 890 DATA 27,65,8
 900 DATA lfeed12,3
 910 DATA 27,65,12
 920 DATA ":2.defchar",9
 930 DATA 27,58,0,0,0,27,37,1,0
 940 DATA ":2.defoff",4
 950 DATA 27,37,0,0
 960 REM Redefine ! CHR33 to full line
 970 DATA ":2.char!",17
 980 DATA 27,38,0,33,33,139,0,0,0,0,0,2
55,0,0,0,0,0
 990 REM Redefine & to alpha
1000 DATA ":2.char&",17
1010 DATA 27,38,0,38,38,139,0,24,36,66,
36,24,36,66,36,0,0
1020 REM Redefine $ to beta
1030 DATA ":2.char$",17
1040 DATA 27,38,0,36,36,139,0,0,255,255
,168,168,168,168,80,80,0
1050 REM Redefine # to Capital Delta
1060 DATA ":2.char#",17
1070 DATA 27,38,0,35,35,139,3,4,9,16,33
,64,33,16,9,4,3
1080 REM Redefine % to reversible react
ion sign
1090 DATA ":2.char%",17
1100 DATA 27,38,0,37,37,139,20,2,21,0,2
0,0,20,0,84,32,20
1110 REM Redefine > to forward arrow
1120 DATA ":2.char>",17
1130 DATA 27,38,0,62,62,139,24,0,24,0,2
4,0,24,129,90,36,24
1140 DATA END,0
```

A Perpetual Calendar

by Peter Brown

Although the subject of calendars has been touched on before in BEEBUG I've never really found a program which meets my needs. This one, hopefully, meets all requirements.

The internal calendar in the Master 128 is very accurate and useful in its own right as an instant-access calendar. However, it does not allow you to look at other dates or whole months at a time, and cannot cope with dates before 1900 or after 1999. The program listed here can display or print the calendar month by month for any year between 1753 and 5000 A.D. in the United Kingdom, or even earlier in other countries.

HOW TO USE THE PROGRAM

Type in the program as in the listing and then save it - the program is written entirely in Basic, so there should be no real problems, but watch out for the abbreviation of 'September' in line 1550 to 'Septemb.'. When the program is running, if you are using a Master 128 you will be faced with the calendar for the current month and year. If you are using a Model B or Compact then see later in this article for details of how to change the program for use with these models. From then on the program is the same for all machines - pressing the right cursor key will step up a month, the left key down a month - the calendar changing accordingly. Pressing the up and down keys will alter the years in exactly the same way.

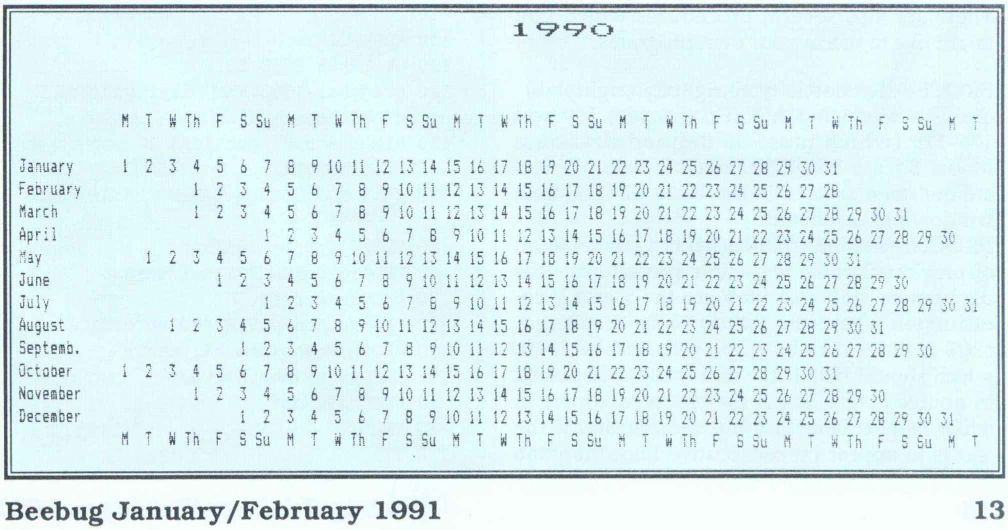
Pressing 'D' will let you enter the date you wish to display from the keyboard, 'P' will print the calendar for the current year in either planner format, or in monthly blocks. The 'Q' key will let you quit the program.

USING THE CALENDAR

The program assumes that you will be using calendars for this country. Although there is no actual difference between the calendars for various countries, the date when the Gregorian calendar was adopted differs from country to country, so you will need to change the variable called *gregyear* in line 120 according to table 1.

TECHNICAL NOTES

To adapt the program for use with the Model B or Compact you should change line 180 to PROCenterdate to make the program ask for a starting date rather than try to extract the date from the Master's TIME$.



```
                                                   1990

              M  T  W Th  F  S Su  M  T  W Th  F  S Su  M  T  W Th  F  S Su  M  T  W Th  F  S Su  M  T  W Th  F  S Su  M  T

January       1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
February               1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
March                  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
April                           1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
May              1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
June                      1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
July                            1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
August              1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Septemb.                     1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
October       1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
November               1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
December                     1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
              M  T  W Th  F  S Su  M  T  W Th  F  S Su  M  T  W Th  F  S Su  M  T  W Th  F  S Su  M  T  W Th  F  S Su  M  T
```

1583	Most of Western Europe
1587	Poland
1588	Hungary
1700	Denmark, the protestant states of Germany and Holland
1741	Sweden
1753	Great Britain and the American Colonies
1873	Japan
1913	China
1916	Bulgaria
1919	Russia and Turkey
1920	Yugoslavia and Romania
1924	Greece and the Eastern Church

Table 1

For the model B only you will also need to remove the REM from line 160 to activate PROCdefchars, and because of space considerations, add the following move-down routine to the start of the program:

```
      0 IF PAGE<&E01 THEN 100
      1 *KEY0
*T.|MFORA%=0TO(TOP-PAGE)STEP4:A%!&E00=A%!P
AGE:NEXT|MPAGE=&E00|MOLD|MRUN|M
      2 *FX138,0,128
```

A corollary of this is that Compact and model B users can omit PROCgettoday (lines 3310 to 3400), Compact owners can omit PROCdefchars (lines 3400 to 3520), and Master owners can omit both PROCdefchars and PROCenterdate (lines 2070 to 2260).

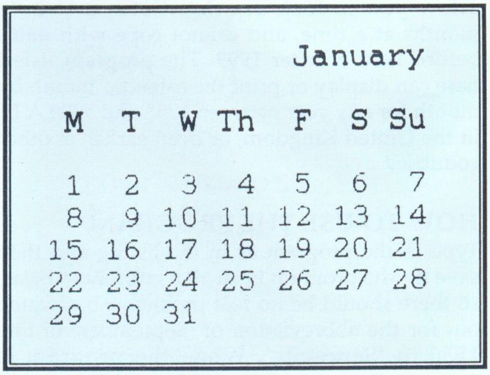
There are also several procedures which you might like to use in your own programs:

PROCbox(leftxtab,leftytab,rightxtab,rightytab) draws a box using ASCII codes 166, 169 and 176-179 (which must be defined first on a Model B - see PROCdefchars). The parameters are the same as used in the Basic VDU 28 (text windowing) command.

PROClarge(xtab,ytab,text$,dt) is a combination of previous double and triple procedures - the parameters xtab and ytab are the x and y co-ordinates where the text is to be displayed, text$ is the text to be displayed, and lastly dt, which should be 0 if the text is to be displayed in double height or 1 if the text is to appear in triple height. Bear in mind though that if the text is to appear on consecutive lines the y tab positions should be 2 apart if they are to appear in double height, or 3 apart if they are to appear in triple height.

EXTENDING THE PROGRAM

If your printer supports quadruple height printing then you could adapt the printing procedures to print only a whole month's calendar at a time, or you could print the dates in a long strip - leaving spaces for reminders etc. There are many other possibilities for extending the printing routines in this program.



```
                 January

 M   T   W  Th   F   S  Su

 1   2   3   4   5   6   7
 8   9  10  11  12  13  14
15  16  17  18  19  20  21
22  23  24  25  26  27  28
29  30  31
```

```
   10 REM Program PerpCal
   20 REM Version B1.20
   30 REM Author  Peter Brown
   40 REM BEEBUG  Jan/Feb 1991
   50 REM Program subject to copyright
   60 :
  100 MODE 129
  110 ON ERROR GOTO 5010
  120 gregyear=1753:REM First year of Gr
egorian calendar system
  130 DIM L 9:DIM monthlen%(11),monthnam
e$(11),monthno%(11)
  140 X%=L MOD 256:Y%=L DIV 256:A%=10:A=
10
  150 VDU23,1,0;0;0;0;
  160 PROCread:REM PROCdefchars
  170 PROCsetscreen
  180 PROCgettoday:REM PROCenterdate
  190 PROCnewdate(month%,year%)
  200 PROCprint(month%,year%)
  210 PROCcalendar
  220 END
  230 :
```

```
 1000 DEF PROCcalendar
 1010 IF INKEY(-26)PROCchangemonth(0)
 1020 IF INKEY(-122)PROCchangemonth(1)
 1030 IF INKEY(-58)PROCchangeyear(0)
 1040 IF INKEY(-42)PROCchangeyear(1)
 1050 IF INKEY(-51)PROCenterdate
 1060 IF INKEY(-17)PROCquit
 1070 IF INKEY(-56)PROCprintcal
 1080 GOTO 1010
 1090 ENDPROC
 1100 :
 1110 DEF PROCsetscreen
 1120 COLOUR 3
 1130 PROCbox(0,4,27,0)
 1140 PROCbox(28,4,39,0)
 1150 PROCbox(0,25,27,5)
 1160 PROCbox(28,25,39,5)
 1170 PROCbox(30,22,37,18)
 1180 PROCbox(29,14,38,10)
 1190 COLOUR 1
 1200 PROClarge(5,1,"Perpetual Calendar"
,1)
 1210 PROClarge(31,8,"Month",0)
 1220 PROClarge(32,16,"Year",0)
 1230 PRINT TAB(31,1)"By"
 1240 PRINT TAB(31,2)"Peter"
 1250 PRINT TAB(31,3)"Brown."
 1260 PROClarge(19,8," S Su",0)
 1270 COLOUR 3
 1280 PROCbanner
 1290 PROClarge(4,8," M  T  W Th  F",0)
 1300 ENDPROC
 1310 :
 1320 DEF PROCbox(lx%,ly%,rx%,ry%)
 1330 FOR A=lx%+1 TO rx%-1
 1340 VDU31,A,ly%,166,31,A,ry%,166
 1350 NEXT A
 1360 FOR A=ry%+1 TO ly%-1
 1370 VDU31,lx%,A,169,31,rx%,A,169
 1380 NEXT A
 1390 VDU31,lx%,ry%,176,31,rx%,ry%,177
 1400 VDU31,lx%,ly%,178,31,rx%,ly%,179
 1410 ENDPROC
 1420 :
 1430 DEF PROClarge(xtab%,ytab%,text$,dt
)
 1440 FOR i%=1 TO LEN(text$)
 1450 ?L=ASC(MID$(text$,i%,1))
 1460 CALL &FFF1
 1470 IF dt=0 THEN VDU23,224,L?1,L?1,L?2
,L?2,L?3,L?3,L?4,L?4:VDU23,225,L?5,L?5,L
?6,L?6,L?7,L?7,L?8,L?8:VDU31,xtab%+i%-1,
ytab%,224,8,10,225:NEXT
 1480 IF dt=1 THEN VDU23,224,L?1,L?1,L?1
,L?2,L?2,L?2,L?3,L?3:VDU23,225,L?3,L?4,L
?4,L?4,L?5,L?5,L?5,L?6:VDU23,226,L?6,L?6
,L?7,L?7,L?7,L?8,L?8,L?8:VDU31,xtab%+i%-
1,ytab%,224,8,10,225,8,10,226:NEXT
 1490 ENDPROC
 1500 :
 1510 DEF PROCread
 1520 FOR Z%=0TO11:READmonthlen%(Z%),mon
thname$(Z%),monthno%(Z%):NEXT
 1530 DATA 31,January,1,28,February,4,31
,March,4,30,April,0
 1540 DATA 31,May,2,30,June,5,31,July,0,
31,August,3
 1550 DATA 30,Septemb.,6,31,October,1,30
,November,4,31,December,6
 1560 FOR H=0 TO 11
 1570 monthname$(H)=monthname$(H)+STRING
$(8-LEN(monthname$(H))," ")
 1580 NEXT H
 1590 ENDPROC
 1600 :
 1610 DEF PROCnewdate(month%,year%)
 1620 days$=STRING$(20," ")+"1  2  3  4
  5  6  7  8  9 10 11 12 13 14 15 16 17 1
8 19 20 21 22 23 24 25 26 27 28 29 30 31"
 1630 Q%=year%DIV100:R%=year%MOD100:S%=R
%DIV12:T%=R%MOD12:U%=T%DIV4
 1640 yearno%=(S%+T%+U%+(19-Q%)*2-(Q%>19
))MOD7
 1650 leapflag%=TRUE
 1660 IF year%MOD400=0 GOTO 1700
 1670 IF year%MOD100=0 leapflag%=FALSE:G
OTO 1700
 1680 IF year%MOD4=0 GOTO1700
 1690 leapflag%=FALSE
 1700 dayno%=yearno%+monthno%(month%)+6
 1710 dayno%=(dayno%+((month%=0ORmonth%=
1)ANDleapflag%))MOD7
 1720 monthprint$=LEFT$(days$+STRING$(10
," "),18+monthlen%(month%)*3-(month%=1AN
Dleapflag%)*3)
 1730 monthprint$=MID$(monthprint$+STRIN
G$(30," "),19-dayno%*3):ENDPROC
 1740 :
 1750 DEF PROCprint(month%,year%)
 1760 midd=-20
 1770 FOR taby=12 TO 22 STEP 2
 1780 midd=midd+21
 1790 PRINT TAB(3,taby)MID$(monthprint$,
midd,21)
```

```
 1800 NEXT taby
 1810 PROClarge(30+((8-LEN(monthname$(mo
nth%))DIV2),11,monthname$(month%),1)
 1820 PROClarge(32,19,STR$(year%),1)
 1830 ENDPROC
 1840 :
 1850 DEF PROCchangemonth(direct%)
 1860 IF direct%=0 AND month%=0 AND year
%=gregyear THEN ENDPROC
 1870 IF direct%=1 AND month%=11 AND yea
r%=5000 THEN ENDPROC
 1880 IF direct%=0 AND month%=0 THEN mon
th%=11:PROCchangeyear(1)
 1890 IF direct%=1 AND month%=11 THEN mo
nth%=0:PROCchangeyear(0)
 1900 IF direct%=0 THEN month%=month%-1:
PROCreturn
 1910 IF direct%=1 THEN month%=month%+1:
PROCreturn
 1920 ENDPROC
 1930 :
 1940 DEF PROCreturn
 1950 PROCnewdate(month%,year%)
 1960 PROCprint(month%,year%)
 1970 PROCcalendar
 1980 ENDPROC
 1990 :
 2000 DEF PROCchangeyear(direct%)
 2010 IF direct%=0 AND year%=5000 THEN E
NDPROC
 2020 IF direct%=1 AND year%=gregyear-1
THEN ENDPROC
 2030 IF direct%=0 THEN year%=year%+1:PR
OCreturn
 2040 IF direct%=1 THEN year%=year%-1:PR
OCreturn
 2050 ENDPROC
 2060 :
 2070 DEF PROCenterdate
 2080 FOR G=27 TO 29:PRINT TAB(1,G)STRIN
G$(38," "):NEXT
 2090 REPEAT
 2100 PRINT TAB(9,27)"Year ?"
 2110 PRINT TAB(9,29)STRING$(29," ")
 2120 *FX 15,0
 2130 INPUT TAB(9,29)year%
 2140 UNTIL year%<5001 AND year%>gregyea
r-1
 2150 REPEAT
 2160 PRINT TAB(23,27)"Month ?"
 2170 PRINT TAB(23,29)STRING$(16," ")
 2180 INPUT TAB(23,29)month%
 2190 UNTIL month%>0 AND month%<13
 2200 month%=month%-1
 2210 PROCbanner
 2220 PROCnewdate(month%,year%)
 2230 PROCprint(month%,year%)
 2240 PROCcalendar
 2250 ENDPROC
 2260 :
 2270 DEF PROCbanner
 2280 FOR K=26 TO 29:PRINT TAB(0,K)STRIN
G$(40," "):NEXT
 2290 PROCbox(0,30,39,26)
 2300 VDU31,3,27,136,9,137
 2310 VDU31,22,27,139,9,138
 2320 VDU31,23,28,81,31,4,28,68,31,8,29,
80
 2330 COLOUR 1
 2340 PRINT TAB(7,27)"Change month"
 2350 PRINT TAB(26,27)"Change year"
 2360 PRINT TAB(7,28)"Enter new date"
 2370 PRINT TAB(26,28)"Quit program"
 2380 PRINT TAB(11,29)"Print Calendar"
 2390 COLOUR 3
 2400 ENDPROC
 2410 :
 2420 DEF PROCquit
 2430 FOR K=27 TO 29:PRINT TAB(1,K)STRIN
G$(38," "):NEXT K
 2440 PRINT TAB(2,27)"Are you sure that
you want to quit ?"
 2450 *FX 15,0
 2460 G$=GET$
 2470 IF G$="Y" OR G$="y" THEN CLS:END
 2480 PROCbanner
 2490 PROCcalendar
 2500 ENDPROC
 2510 :
 2520 DEF PROCprinter(min)
 2530 FOR line%=0 TO 105 STEP 21
 2540 FOR monthnm=min TO min+2
 2550 PROCnewdate(monthnm,year%)
 2560 PRINT ,MID$(monthprint$,line%+1,21
);STRING$(5," ");
 2570 NEXT monthnm
 2580 PRINT
 2590 NEXT line%
 2600 ENDPROC
 2610 :
 2620 DEF PROCprintcalblock
 2630 PROCprintmessage
 2640 @%=27
 2650 VDU2:VDU21
 2660 VDU1,27,1,120,1,49:REM Switch NLQ
on
 2670 PROCprintyear
 2680 FOR print=0 TO 9 STEP 3
 2690 VDU1,27,1,69
 2700 PRINT
 2710 PRINT STRING$(14," ")+monthname$(p
rint)+STRING$(14," ")+monthname$(print+1
)+STRING$(14," ")+monthname$(print+2)
 2720 PRINT
 2730 PRINT STRING$(3,"  M  T  W Th  F
S Su"+STRING$(6," "));
 2740 VDU1,27,1,70
 2750 PRINT
 2760 PROCprinter(print)
 2770 NEXT print
 2780 VDU1,27,1,120,1,48:REM Switch NLQ
off
 2790 VDU6
 2800 VDU3
 2810 PROCbanner
 2820 ENDPROC
 2830 :
 2840 DEF PROCprintmessage
 2850 FOR P=27 TO 29:PRINT TAB(1,P)STRIN
G$(38," "):NEXT P
 2860 PRINT TAB(5,27)"Printing Calendar
for ";year%
 2870 PRINT TAB(5,29)"Be Patient !!!"
 2880 ENDPROC
 2890 :
 2900 DEF PROCprintyear
 2910 VDU1,27,1,87,1,49:VDU1,27,1,69
 2920 PRINT' STRING$(18," ")+STR$(year%)
 2930 VDU1,27,1,87,1,48:VDU1,27,1,70
 2940 PRINT''
 2950 ENDPROC
 2960 :
 2970 DEF PROCprintcalplan
 2980 PROCprintmessage
 2990 VDU2:VDU21
 3000 PROCprintyear
 3010 PROCprintdows
 3020 FOR print=0 TO 11
 3030 VDU1,27,1,69
 3040 PRINT monthname$(print)+STRING$(3,
" ");
 3050 VDU1,27,1,70
 3060 VDU1,15
 3070 PROCnewdate(print,year%)
 3080 PRINT LEFT$(monthprint$,111)
 3090 NEXT print
 3100 PROCprintdows
 3110 VDU1,17
 3120 VDU6:VDU3
 3130 PROCbanner
 3140 ENDPROC
 3150 :
 3160 DEF PROCprintcal
 3170 FOR J=27 TO 29:PRINT TAB(1,J)STRIN
G$(38," "):NEXT J
 3180 PRINT TAB(5,27)"Print Calendar"
 3190 PRINT TAB(5,28)"Plan or Block ?"
 3200 *FX 15,0
 3210 G$=GET$
 3220 IF G$="P" OR G$="p" THEN PROCprint
calplan
 3230 IF G$="B" OR G$="b" THEN PROCprint
calblock
 3240 ENDPROC
 3250 :
 3260 DEF PROCprintdows
 3270 VDU1,15:PRINT STRING$(12," ")+STRI
NG$(5," M  T  W Th  F  S Su ")+" M  T":V
DU1,17
 3280 PRINT
 3290 ENDPROC
 3300 :
 3310 DEF PROCgettoday
 3320 month$=MID$(TIME$,8,3)
 3330 year%=VAL(MID$(TIME$,12,4))
 3340 num=-1
 3350 REPEAT
 3360 num=num+1
 3370 UNTIL MID$(monthname$(num),0,3)=mo
nth$
 3380 month%=num
 3390 ENDPROC
 3400 :
 3410 DEF PROCdefchars
 3420 VDU23,136,0,24,56,127,56,24,0,0
 3430 VDU23,137,0,24,28,254,28,24,0,0
 3440 VDU23,138,24,24,24,24,126,60,24,0
 3450 VDU23,139,0,24,60,126,24,24,24,24
 3460 VDU23,166,0,0,0,255,0,0,0,0
 3470 VDU23,169,24,24,24,24,24,24,24,24
 3480 VDU23,176,0,0,0,7,12,24,24,24
 3490 VDU23,177,0,0,0,192,48,24,24,24
 3500 VDU23,178,24,24,12,7,0,0,0,0
 3510 VDU23,179,24,24,48,192,0,0,0,0
 3520 ENDPROC
 5000 :
 5010 MODE7:REPORT:PRINT" at line ";ERL
 5020 END
```

Structured Listings

by Peter Hayes

This program was originally published in BEEBUG Vol.3 No.9. We have decided to repeat it for the benefit of newer readers, as the functions it provides are very useful. It has been updated to work with the Master.

One of the features of BBC Basic is the extent to which it allows the programmer to write well structured programs, more so than most other versions of Basic. However, the Beeb's limited memory often forces a program to be written in a style which obscures this structure, resulting in programs which are difficult to read. Although the LISTO7 command can be used when listing programs to provide a more structured result, it is quite limited in what it can achieve. The program given here, PRList, is a useful utility for producing much better structured listings than LISTO7 alone, and will enhance the appearance of your programs considerably.

The features of this program are:

1. Indentation of statements in the same way as LISTO7.
2. Splitting of multi-statement lines.
3. Splitting of IF-THEN-ELSE onto separate lines.
4. Displaying the hex codes of non-printable characters.

To start with, you will need to type in the program and save it (be careful when typing in the machine code section from lines 1430 to 1710). The PRList program can sit anywhere in memory, but for simplicity we shall refer to one method of using this program. Load the Basic program to be listed, and then type:

```
PRINT ~PAGE
PAGE=TOP+&100
LOAD "PRList"
```

and then run the program PRList. This will ask you four questions. The first question is about the PAGE value of your Basic program. This can be obtained from the first command that you typed above (normally &E00 for the Master and Compact, &1900 for the Model B). The computer will then ask you for a start line. This is the line number from which you wish to start the formatted listing. You are then asked if you wish to output codes as numbers. This refers to all teletext control codes and VDU codes below 32. If you wish to display (or print) the codes as spaces then answer 'N' to this question. If you wish to print the codes as condensed characters (Epson based printers) or display them as 2 digit hex bytes on the screen then answer 'Y'.

The final question will ask if you require output to the printer or to the screen alone. If you wish to display the program only on the screen, then the computer enters 'paged mode' which requires you to press the Shift key to continue listing the program.

Using this utility will enable you to produce program listings with a well structured format even though your working version may be highly condensed. It won't, of course, put back spaces that may have been 'compacted' out, but it will make your programs much easier to read and understand.

PROGRAM NOTES

The program is designed to produce a printout to an Epson (or compatible) printer, using condensed mode for any control characters as described above. The printer control codes for this are contained in lines 1210 and 1260. They can be readily changed to suit other printers (or omitted altogether, though this may well produce a slightly less readable program).

The outer loop of the main program runs from line 130 to 300 and controls the processing of a full line. The second loop, starting at line 160, scans each character of a line and takes appropriate action.

The procedures called are:

PROCassemble. This sets up the machine code to deal with tokenised Basic keywords. This procedure searches the Basic keyword table for a token to match the number stored in A% and then prints the keyword.

PROCinit. This sets up the global variables.

PROCnum. This prints out the line number.

PROCstr. This copies strings allowing for special characters.

PROCcode. This outputs the hexadecimal value of any special codes. If you have answered yes to both the printer and display code options it will set condensed printing to print the codes.

The program accesses the Basic ROM in order to read the token table, and will recognise versions of Basic supplied with the model B, B+, Master 128 (including those fitted with the new MOS) and Compact. An error is generated if an unrecognised version of Basic is encountered eg BASIC 1.

```
   10 REM Program PRLIST
   20 REM Version B1.4
   30 REM Author  Peter Hayes
   40 REM BEEBUG  Jan/Feb 1991
   50 REM Program subject to copyright
   60 :
  100 CLS:VDU15
  110 ON ERROR GOTO 1730
  120 PROCassemble:PROCinit
  130 REPEAT
  140 @%=5:PRINT ' 256*?(P%+1)+?(P%+2);:
FOR J%=1 TO S%:PRINT" ";:NEXT J%:@%=10
  150 I%=P%+4
  160 REPEAT
  170 C%=?I%
  180 IF C%<=&1F THEN PROCcode:GOTO 270
  190 IF C%=&22 THEN PROCstr:GOTO 270
  200 IF C%=&3A THEN PRINT'"      :";:FOR
J%=1 TO S%:PRINT" ";:NEXT J%:GOTO 270
  210 IF C%<&80 THEN PRINT CHR$(C%);:GOT
O 270
  220 IF (C%=&8B) OR (C%=&8C) THEN PRINT
 '"      ";:FOR J%=1 TO S%:PRINT" ";:NEXT
 J%:GOTO 260
  230 IF C%=&8D THEN PROCnum:GOTO 270
  240 IF (C%=&E3) OR (C%=&F5) THEN S%=S%
+2
  250 IF (C%=&ED) OR (C%=&FD) THEN S%=S%
-2
  260 A%=C%:CALL token
  270 I%=I%+1
  280 UNTIL I%>P%+?(P%+3)-1
  290 P%=P%+?(P%+3)
  300 UNTIL ?(P%+1)=&FF
  310 PRINT:VDU3
  320 END
  330 :
 1000 DEF PROCnum
 1010 LOCAL n%
 1020 n%=(?(I%+3) AND &3F)*256+(?(I%+2)
AND &3F)
 1030 IF (?(I%+1) AND &20)=&20 THEN n%=n
%+128
 1040 IF (?(I%+1) AND &10)=0 THEN n%=n%+
64
 1050 IF (?(I%+1) AND &4)=0 THEN n%=n%+1
6384
 1060 PRINT STR$(n%);
 1070 I%=I%+3
 1080 ENDPROC
 1090 :
 1100 DEF PROCstr
 1110 REPEAT
 1120 IF (C%<=&1F) OR (C%>=&7F) THEN PRO
Ccode ELSE PRINT CHR$(C%);
 1130 I%=I%+1:C%=?I%
 1140 UNTIL (C%=&22 AND ?(I%+1)<>&22) OR
 C%=&0D
 1150 IF C%=&22 THEN PRINT """";
 1160 ENDPROC
 1170 :
 1180 DEF PROCcode
 1190 IF OC%=0 VDU32:ENDPROC
 1200 LOCAL i%,c%,c$,h%:c%=C%:c$=""
 1210 PRINT CHR$(1)CHR$(15);
 1220 FOR i%=1 TO 2
 1230 h%=48+c% MOD 16:IF h%>57 THEN h%=h
%+7
 1240 c%=c% DIV 16:c$=CHR$(h%)+c$
 1250 NEXT
 1260 PRINT c$;CHR$(1)CHR$(18);
 1270 ENDPROC
 1280 :
 1290 DEF PROCinit
 1300 LOCAL l%,f%
 1310 S%=1
 1320 INPUT'"PAGE value of program =&"PG
$
 1330 P%=EVAL("&"+PG$)
 1340 INPUT'"Start line: "f%
 1350 l%=256*?(P%+1)+?(P%+2)
 1360 IF l%<f% THEN P%=P%+?(P%+3):GOTO 1
```

Continued on page 24

More Memory on a BEEB

Use this short function by Al Harwood to recapture user memory from the high resolution screen modes.

On the BBC micro much of the available program memory is taken up by the screen, especially in the large memory modes 0, 1, and 2 which use 20k). But by sacrificing a few screen lines, quite large amounts of memory can be regained - over 0.5K per line in these memory hungry modes.

This can be done by altering the 6845 CRTC registers to make the screen smaller and reposition the smaller screen in memory.

In more detail, this is done by decreasing the CRTC vertical displayed register (line 1090 in the demo program) by a specific number of lines, and moving the now smaller screen up in memory in order to position the regained memory to the top of the program memory (lines 1070, 1080). Set the text and graphic windows to the smaller size (lines 1050, 1060), and finally set HIMEM to the higher value (line 1100). The data at the end of the listing (lines 1110 - 1140) is used to set constants which vary with the screen mode in use.

The short listing following is basically a function which can do this. This function returns the new value of HIMEM, and so should be called by:

```
HIMEM=FNlose(n)
```

where 'n' is the number of lines to be lost. The function is self-contained, and works in all modes except mode 7.

```
   10 REM Program MORE MEMORY
   20 REM Version B1.2
   30 REM Author  Al Harwood
   40 REM BEEBUG  Jan/Feb 1991
   50 REM Program subject to copyright
   60 :
  100 MODE2
  110 HIMEM=FNlose(5)
  120 END
  130 :
```

```
 1000 DEF FNlose(n%)
 1010 LOCALa%,l%,w%,h%,b%,m%,s%,A%
 1020 A%=132:s%=((USR&FFF4)DIV256)AND&FF
FF
 1030 A%=135:m%=((USR&FFF4)AND&FF0000)DI
V&10000
 1040 FORa%=0TOm%:READl%,w%,h%,b%:NEXT
 1050 VDU28,0,l%-1,w%,n%
 1060 VDU24,0;0;1280;1024-h%*n%;
 1070 VDU23;12,(s%+b%*n%)DIV2048;0;0;0
 1080 VDU23;13,((s%+b%*n%)MOD2048)DIV8;0
;0;0
 1090 VDU23;6,l%-n%;0;0;0
 1100 =s%+&280*n%
 1110 DATA32,79,32,&280,32,39,32,&280
 1120 DATA32,19,32,&280,25,79,40,&280
 1130 DATA32,39,32,&140,32,19,32,&140
 1140 DATA25,39,40,&140
```

SAMOYED

Programmers. Do you have good game and educational game ideas can you write good animation, graphics, sound and music. Then if so send an Electron or BBC B version on tape to SAMOYED. Please include a listing if you can. Teachers. Do you programme or have good educational game ideas, then if so send your ideas or programmes to SAMOYED. Top rates will be payed if we use your idea or programme. NO copyright work will be accepted. Please send S.A.E. if you want your work returned.

SAMOYED
64 TOLPATH
COED EVA 3
CWMBRAN
GWENT
NP44 6UE

Corplan - Correspondence Plan for Wordwise Plus

Reviewed by Ian Waugh

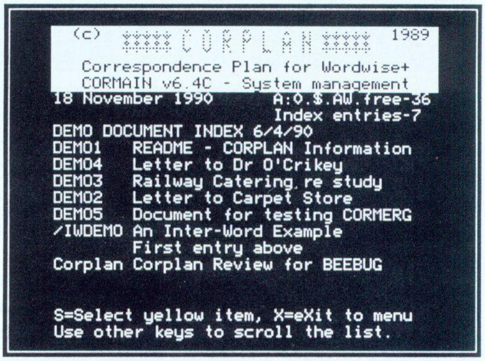
Product	**Corplan**
Supplier	**Corplan Computer Systems, Three Gables, 7A Talbots Drive, Maidenhead, Berkshire SL6 4LZ. Tel. (0628) 24591**
Price	**£19.50 UK post free**

In spite of its age - or perhaps because of its heritage - Wordwise Plus continues to be one of the most popular word processors for the range of BBC computers. It has spawned an entire army of support ROMs, and its in-built programming language - WPPL (Wordwise Plus Programming Language) - has led to the development of countless utility programs to perform every text-manipulatory function you can think of from mail-merge to random sentence generation.

After a quiet period on the support program front, Corplan appears. It describes itself as a Correspondence Planner and consists of a suite of disc-based programs (almost entirely written in WPPL) which provide document indexing, mail-merge and form selection facilities. It will run on virtually any BBC micro and with any filing system (including Watford and Opus), and makes use of the special features of the Master such as date stamping.

The full complement of Corplan programs numbers 18 (there are also additional demo files) although, depending on your application, you may not need all these. However, they are fully integrated and you'd be advised to keep them together. As there are so many files, twin drives are recommended although you could get by at a pinch with a single double-sided drive, but the manual advises against it and so would I. The current version isn't configured for use with a hard disc.

The first step is to produce a *working* disc from the *distribution* disc. 10 pages in the manual give detailed instructions and cover every possible variation of machine, disc drives and filing system. Later on you may want to alter some of the system's default settings. For example, the Setup program, which is chained by the !BOOT file, includes *TV0,1 while I use *TV255,1. It also includes function key definitions containing printer codes. A function key strip listing the definitions is supplied. However, the !BOOT file is arranged so that you can insert overriding definitions after Setup has been executed, so inexperienced users and non-programmers don't have to alter the file. The printer settings are for Epson compatibles but you can alter these, too. There is also provision for loading a SpellMaster dictionary.

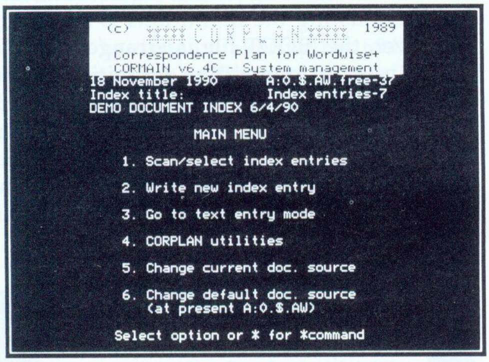


```
(c)  ::::: C O R P L A N :::::  1989
  Correspondence Plan for Wordwise+
  CORMAIN v6.4C - System management
18 November 1990        A:0.$.AW.free-36
                         Index entries-7
DEMO DOCUMENT INDEX 6/4/90
DEMO1   README - CORPLAN Information
DEMO4   Letter to Dr O'Crikey
DEMO3   Railway Catering re study
DEMO2   Letter to Carpet Store
DEMO5   Document for testing CORMERG
/IWDEMO An Inter-Word Example
        First entry above
Corplan Corplan Review for BEEBUG


S=Select yellow item, X=eXit to menu
Use other keys to scroll the list.
```

Scanning an Index to select a document

When you boot your work disc, the system runs through some initialising procedures, clears WW+ memory and presents you with the main menu which has six options. The !BOOT file will terminate if you try to boot from within WW+ which prevents losing text through a reboot should you hit Shift and Break. You can call the menu at any time by pressing Shift and f9, a process the manual calls recycling. When you enter the main text area, all ancillary programs are deleted from the segments to conserve memory.

The first menu option lets you scan a list of filenames and load one. Each entry can have a 30-character description - a boon when using DFS - although the maximum filename size is limited to seven characters even when using ADFS to maintain file compatibility. You can scroll through the indexed files, one way only, by pressing any key. The selected file can be loaded into the text area or into segments 0 or 1. You are warned if you are about to delete existing text.



```
(c)      C O R P L A N        1989
   Correspondence Plan for Wordwise+
   CORMAIN v6.4C - System management
18 November 1990       A:0.$.AW.free-37
Index title:           Index entries-7
DEMO DOCUMENT INDEX 6/4/90

              MAIN MENU

     1. Scan/select index entries

     2. Write new index entry

     3. Go to text entry mode

     4. CORPLAN utilities

     5. Change current doc. source

     6. Change default doc. source
        (at present A:0.$.AW)

   Select option or * for *command
```

Corplan Main Menu

The second option allows you to add a new filename and description to the Index. A warning is given if a file of the same name exists either in the Index or on the disc.

Option three takes you to text entry mode. You can go there directly, or via the Forms menu. The use of forms is one of Corplan's most interesting features. Forms are simply templates which define the layout of a letter or a report. They are especially useful in WW+ as the text entry screen is not WYSIWYG. The Forms menu can show up to 22 forms. Each option is lettered and followed by a description of the form.

The simplest type of form will only contain embedded commands for line length, page length, margin settings and so on. However, some forms can take you to an address file to load the address of the recipient. This includes search options (through an address file) with wildcards, and the ability to edit the address file. If you are preparing a business letter, some forms will prompt for 'Your' and 'Our' References. One really nice feature is the inclusion of the date. I'm lazy and use the PD command which is fine for the printout but the file itself doesn't tell you when it was written. Corplan actually inserts the date in words into the document. Finally, after loading an address into a form, pressing f9 will send the address to the printer (preferably to an envelope).

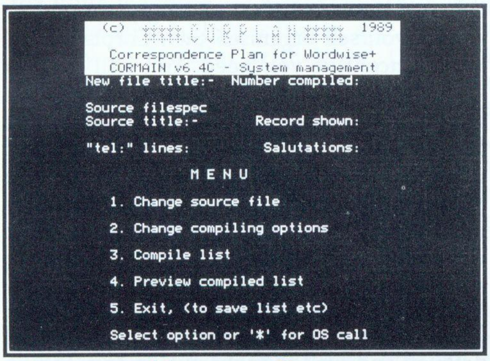
Option 4 accesses the Corplan Utilities Menu which contains eight options. The first one handles mail-merging. Obviously, the main text has to be formatted in a certain way and many of the demo files are already suitably formatted. The address file can include a salutation. The greeting is selected automatically according to the salutation. A Sir, Sirs or Madam results in "faithfully", anything else produces "sincerely".

You may have several address files on your work disc, and Corplan's mail-merge lets you select an individual address from a number of address files, so all the addresses need not be in the same file. Some checks are made to ensure that the document is suitable for merging. When all is well the Mailmerge menu appears. There is a preview mode which lets you run through all the options without wasting any paper - useful! There are also options for selecting single sheet and continuous stationery.

The next thing you need to do is print envelopes or labels for all the letters. The Address Label Printer handles this. It will format the addresses in one or two columns, optionally remove associated telephone numbers and let you set various margins. The Address List Compiler lets you make a new address list by copying addresses from one or more other address lists.

If you often load spooled ASCII text into WW+ for further editing you'll know that you usually have to remove a number of multiple spaces, Carriage Returns and pad characters - | - which WW+ substitutes for any 'illegal' characters which it does not recognise. The Text

Deformatting Utility will do this for you. It's not guaranteed to fully correct any errant text but it's a good start and you can tailor the routine to specific functions and use it in conjunction with formatting or deformatting routines of your own.



```
(c)   ***** C O R P L A N *****   1989
  Correspondence Plan for Wordwise+
  CORMAIN v6.4C - System management
New file title:-  Number compiled:

Source filespec
Source title:-       Record shown:

"tel:" lines:         Salutations:

            M E N U

    1. Change source file

    2. Change compiling options

    3. Compile list

    4. Preview compiled list

    5. Exit, (to save list etc)

    Select option or '*' for OS call
```

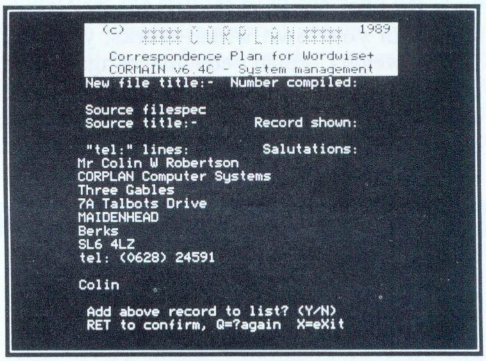
Address Compiler Menu

The Index to a group of files is held in a segment and as a file on disc. After you have been using the system for a while you may want to remove documents which are no longer required. The Index & Document Weeder Utility helps you do this. During the first part of the process it is simply taking instructions and it's not until you give it a final confirmation that it will make the changes. This should minimise the possibility of accidental deletions.

There are two utilities for initialising new document sources in DFS and ADFS formats. A document source is simply a directory in which documents are stored. Under DFS this is W. Under ADFS, 10 directories are created from AW to JW. It establishes INDX (sic) and ADDRESS files, and will copy the addresses in an existing file to it. INDX holds the names of the documents and the name of the Index. For example, you may call the AW directory *Business Letters*, BW may be *Aunt Harriet*, CW may be *Begging Letters* and so on. You must retain the two-character directory names so that the program can identify them and the Index titles must be changed in the INDX file manually and resaved to the disc.

The final utility is a Form Planner. It produces grids which help you identify character positions on a sheet of paper to make it easier to work out the correct formatting commands when designing your own forms. You can preview the grid before printing it. After designing your own forms, you can insert their names in the FMENU file in the root directory so that they appear when you select the forms option.

Corplan also has a limited degree of support for InterWord. In particular, you can enter it and load a form into it. You can select an address from an address file which is saved under the name of LABEL and can be loaded into InterWord by normal means. There is no facility to re-enter WW+, however, which is natural enough.

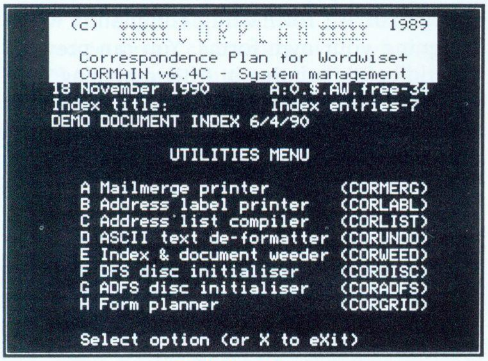


```
(c)   ***** C O R P L A N *****   1989
  Correspondence Plan for Wordwise+
  CORMAIN v6.4C - System management
New file title:-  Number compiled:

Source filespec
Source title:-       Record shown:

"tel:" lines:         Salutations:
Mr Colin W Robertson
CORPLAN Computer Systems
Three Gables
7A Talbots Drive
MAIDENHEAD
Berks
SL6 4LZ
tel: (0628) 24591

Colin

 Add above record to list? (Y/N)
 RET to confirm, Q=?again  X=eXit
```

Compiling Addresses

Corplan is completely menu-driven - although you have to save the files yourself - and generally one keypress is all that is required to move around. There are prompts and confirmations all over the system and you would have to be very careless indeed to perform a destructive act by mistake.

The 120-page manual covers each of Corplan's functions in detail. On a purely cosmetic note, a few more diagrams - essential or not - would break up the manual's layout and make it less daunting and less tiring on the eye. If you need further help, Corplan Computer offers user support should it be required. There are several demonstration forms and documents to help you gain experience of the system as quickly as possible.



```
(c)   ***** C O R P L A N *****   1989
      Correspondence Plan for Wordwise+
      CORMAIN v6.4C - System management
18 November 1990        A:0.$.AW.free-34
Index title:            Index entries-7
DEMO DOCUMENT INDEX 6/4/90

             UTILITIES MENU

   A Mailmerge printer          (CORMERG)
   B Address label printer      (CORLABL)
   C Address list compiler      (CORLIST)
   D ASCII text de-formatter    (CORUNDO)
   E Index & document weeder    (CORWEED)
   F DFS disc initialiser       (CORDISC)
   G ADFS disc initialiser      (CORADFS)
   H Form planner               (CORGRID)

   Select option (or X to eXit)
```

Utilities Menu

So is Corplan for you? If you already have a system of your own which works, you will need to weigh Corplan's advantages against the disadvantages of restructuring your system. I use ADFS with a different directory name for each individual or company I write to and WW+ II's cursor-driven menu selection system takes care of movement around the disc. I hold templates for all my standard letters and documents within each directory. These, of course, were constructed before Corplan came onto the scene.

But if you have numerous files scattered over several discs and just never got around to getting yourself organised then Corplan could be the incentive you need. You will have to spend a little time reading the manual and setting up the system to suit your application, but once that's done the hard work is over. If you need to produce several slightly different types of letter you'll find the forms options particularly useful. Corplan would also work well as the centre of a word processing/mailing system for a small club or business.

Corplan is a very clever set of programs, well detailed and well thought out. In fact, there's virtually nothing serious I can think of in the package to criticise (I even wrote this review under Corplan!). It's very reasonably priced and comes with a 14-day money-back guarantee.

Structured Listings (continued from page 19)

```
350
 1370 INPUT'"Output codes as numbers: "C
$
 1380 IF LEFT$(C$,1)="Y" OR LEFT$(C$,1)=
"y" OC%=1 ELSE OC%=0
 1390 INPUT'"Output to printer: "P$
 1400 IF LEFT$(P$,1)="Y" OR LEFT$(P$,1)=
"y" VDU2 ELSE VDU14
 1410 ENDPROC
 1420 :
 1430 DEF PROCassemble
 1440 REPEAT READ ver%,add%
 1450 UNTIL ver%=?&8008 OR ver%=256
 1460 IF ver%=256 PRINT"Unknown version
of Basic"'"Cannot continue with listing"
:END
 1470 DIM SP 100:FORZ=0 TO2 STEP2
 1480 P%=SP:[OPTZ
 1490 .token
 1500 STA &70:LDA#add%MOD256:STA&71
 1510 LDA#add%DIV256:STA&72
 1520 LDY#0:.loop:LDA(&71),Y
 1530 CMP&70:BEQ found
 1540 CLC:LDA&71:ADC#1:STA&71
 1550 LDA&72:ADC#0:STA&72:CLC
 1560 JMP loop
 1570 .found:SEC:LDA&71:SBC#1
 1580 STA&71:LDA&72:SBC#0:STA&72
 1590 CMP#add%DIV256:BNE notbeg
 1600 LDA&71:CMP#add%MOD256
 1610 BNE notbeg
 1620 DEY:JMP print
 1630 .notbeg LDA(&71),Y
 1640 CLC:CMP#&80:BCC found
 1650 INY
 1660 .print:INY:LDA(&71),Y
 1670 CLC:CMP#&7F:BCS end
 1680 JSR&FFEE:JMP print
 1690 .end:RTS:]
 1700 NEXT
 1710 ENDPROC
 1720 :
 1730 ON ERROR OFF
 1740 IF ERR<>17 THEN MODE 7
 1750 REPORT:PRINT" at line ";ERL
 1760 END
 1770 DATA 0,&806D,1,&8071,4,&8456,7,&85
13,64,&842F,256,0
```

Mastering Edit (Part 1)

by Mike Williams

If you own a Master 128 then you have access to *Edit*, one of the most underrated programs for your machine. The Master 128 is well provided with built-in software, and you may already be familiar with View, the word processor, and Viewsheet its spreadsheet complement. View is an excellent word processor, and if you are dealing with letters, reports and the like then it is ideally suited to this task. However, there are many situations in which View is less helpful, and some where the power and flexibility of Edit has no equal.

Edit is ideal for entering and editing programs, even programs in Basic. The EDIT keyword will quickly transform the current Basic program in memory into a format suitable for Edit, while Shift-f4 followed by 'B.' (for Basic) will leave you with the edited program ready to run. Extensive editing is then much easier as you no longer have to copy in its entirety any line being modified. There is one disadvantage - Edit has no knowledge of Basic, so if you make any mistake Edit will be unaware - and so will you until you try to run the program.

For example, Edit allows you to alter line numbers, but it won't automatically alter any other reference to the same line number. Changing line numbers can be useful. If you want to change the order of two lines in a Basic program just edit their line numbers to reflect correctly their desired positions, and when you return to Basic the two lines will be correctly ordered as specified.

Edit can also supplement View. There is nothing to stop you loading a View format file directly into Edit using function key f2 to select the *Load* option. Some of the file may look a little weird because of the ASCII codes (or characters) used by View to represent things like highlight markers, rulers, etc. But have you ever tried searching a View file for, say, two spaces to replace them by a single space? It won't work. In Edit this is no problem, and thus provides an easy way to search for all occurrences of two spaces, and to replace each found with a single space. There are other situations where Edit supplements the facilities of View.

Another example of Edit's use is in converting a file from one format to another, say Interword to View, or similar. For example, Wordwise represents Tab by the code for the Tab character (ASCII 9) but with 128 added (i.e. ASCII 137). View uses the standard Tab character (Ctrl-I). If you use Wordwise's spool option, then every Tab will be replaced by the corresponding number of spaces, which is not really the best solution. Read the Wordwise file into Edit, and you can easily replace Wordwise's Tab character with that used by View.

Other similar problems arise - for example, View Professional terminates every line with a space followed by a Carriage Return (ASCII 13). Leaving the space in can cause havoc when such a file is edited and reformatted within View. The solution is to use Edit to search for each occurrence of <space><Return> and replace by just <Return>. All this is trivial stuff as far as Edit is concerned, and it is capable of a whole lot more. This month I propose to examine the main features and characteristics of Edit, and then follow this next month with a look at some of its more advanced (and more powerful) features.

UNDERSTANDING EDIT

Edit treats every file simply as a sequence of bytes (as an ASCII file). Each byte is represented on screen by a single character, with the non-printing characters (ASCII codes 0 to 31) represented as inverse-video control characters. For example, the Return character (ASCII code 13) already referred to can also be entered from the keyboard by pressing Ctrl-M (where the ASCII code for 'M' is 77, i.e. 13 + 64). In Edit the Return character appears as an inverse-video 'M' (i.e. a black 'M' on a white background). Codes in the range 128 to 255 will be displayed as the characters of the Master's extended character, set as listed in the Master's User Guide.

The important thing to remember is that Edit works with ASCII files (or treats all files as ASCII). That's why you can't edit a Basic program by simply loading it directly into Edit. A Basic program is tokenised, with each keyword being replaced by a single token (or coded value). If loaded into Edit in this form, each token would appear as a corresponding character. To edit a Basic program properly it must be converted first from its more normal tokenised form into a simple ASCII format. That's what the EDIT keyword does. The alternative (which would work just as well, but which is more long winded) would be to spool the Basic program out to a file (this process saves the program as an ASCII text file), and then load this into Edit with the f2 function key command.

Let's suppose you have loaded a Basic program into memory. Type EDIT (followed by Return) and notice what happens. The screen should clear and be replaced by the 'long' form of the Edit display (with help information at the head of the screen), and a window below that in which the start of your program will appear.

The help display at the head of the screen should show the meanings of the function keys as used by Edit, and this serves as a useful reminder if you have lost or forgotten the supplied key-strip. If this is not visible on screen, press Shift-f5 (for SET MODE), and in response to the prompt at the foot of the screen enter 'D'. The screen display should then change to the format described above. You can also use SET MODE to select the screen mode - I usually use 131 (shadow mode 3). One useful feature of Edit is that it remembers such settings from one time to the next, so decide what your preferences are, and Edit will remember them.

USING EDIT

Once in Edit, with your program (or whatever) displayed, you can check out the basic editing features. Many controls are very similar to View: Ctrl- and Ctrl- to move to the start or end of the file, Shift- and Shift- to move up or down one screenful, and so on. If you use the cursor keys to scroll continuously up or down you will notice that you can never reach the very top or very bottom of the Edit window (unless at the very start or very end of the file respectively). Thus you can always see a few lines both above and below the current cursor position.

I normally find that having the details of the function keys displayed at the head of the screen more than compensates for the reduced size of the edit window, but that is personal preference. Although cursor movement is very much as for View there are some differences in the way in which things are controlled. The Delete key operates as normal, deleting the character to the left of the current cursor position, but in Edit it is the Copy key which deletes the character at the cursor, not f9 as in View.

The other difference is that of dealing with blocks of characters. In View you mark the start and end of a block. This is not always true in Edit. To delete a block move to the start and press f6 (MARK PLACE), then move to the end of the block and press Shift-f8 (without placing another marker). To copy or move blocks of characters, you mark both start and finish with f6 before pressing f7 (MARKED COPY) or Shift-f7 (MARKED MOVE).

Another useful key is Shift-f1 (INSERT/OVER) which toggles between insert mode and overwrite mode. Again, whatever setting is current at the end of a session is remembered by Edit for the next one. There is, unfortunately, no case change key, so if you do need to change from upper to lower case, or vice versa, the best solution is to select overwrite mode and carefully type over the top of what is already there.

As far as loading and saving of files is concerned, I have already described how a Basic program in memory can be transferred into Edit, and a return to Basic made at the end. If you are dealing with ASCII files to start with, then f2 and f3 will prompt for the name of a file to load or to save respectively. Note the use of Shift-f2 to load and insert a file at the current cursor position (setting a marker will not work, and will prevent the Insert command from having effect).

Finally, before getting to the heart of Edit, note too the use of f1. This gives a star prompt ready for any star command to be typed in, useful for cataloguing a disc or whatever without leaving Edit. Escape will always cancel command mode. You can also use command mode to leave Edit by typing Basic (or just B.) in response to the prompt, but do remember that any Basic program you have been editing (or any other file) will be immediately lost unless saved in ASCII format first. Incidentally, one of the few irritating traits I have found in Edit, is that using this method to exit from Edit leaves a screen window set unless you subsequently change mode.

SEARCH AND REPLACE

Amongst the most powerful features of any editor are those to search for, and optionally replace, any specified character string. And it is in this area that Edit is particularly versatile and powerful.

Two function keys only do all the work, f4 which performs a selective search and replace, and f5 which performs a global search and replace. Using f4, Edit will search from the current cursor position for the first occurrence of the specified string and pause when this has been found. It will then prompt you to *continue* (to search for the next occurrence) or *replace* (the target string with a new string). The keys with which to respond are 'R' for replace and 'C' for continue (not View's 'Y' or 'N'). The selective search may or may not specify a replacement string at the outset.

However, the global search and replace is only valid if both target and replacement are specified, and all occurrences will be replaced (very quickly) without any further response from the user. This makes the use of f5 very powerful, and you should be cautious about using it unless you are quite sure what you are doing. It can be fatally easy to change one thing for another throughout a file only to find the wrong string has been replaced. And trying to change back to the original state may be impossible.

For example, if you decide, when editing a program, to change the name of a variable, do a search first to establish that you haven't already used the new name for something else. If you have, and you make all the changes you will have used the same name for two different purposes, but at this stage you will be quite unable to separate one from the other, except by individual inspection.

It is often preferable, particularly with more complex search strings, to use the selective search first to ensure that you have got it right, and only then use the global search and replace.

In both cases, a replacement string is specified by following the search string by a '/' and then the replacement string. Putting '/' followed by nothing (except Return) will result in the target string being deleted (replaced by 'nothing').

Escape can be used to terminate a selective search prematurely, and once a search has been made, a further press of f4 immediately followed by Return will use the same search string as the last time.

So far we have really only scratched the surface of the search and replace facilities. To cover this in some detail will require a further article in its own right, and that must wait for the next issue. In the meantime, if you have not used Edit before, or only a little, I suggest you give it try - you might be surprised at just how much you can achieve.

Cryptology

by Bernard Hill

When we were small children I suspect that we all enjoyed making secret messages which we could send to our friends and have them decode them. This children's pursuit clearly has a very important military value and is the subject of much classified research, but I propose in this article to introduce the elements of this subject and to produce a simple code-making program.

CAESAR'S ALGORITHM

Our alphabet has 26 characters, plus a space. When a message was required to be relayed, Caesar (using the Latin alphabet!) reputedly shifted all letters cyclically three places to the right (counting space as the first letter of the alphabet) so that:

```
BUY BEEBUG
```

becomes:

```
EXACEHHEXJ
```

This is not very profound, and very easy to break even if you don't know the shift distance: you merely need to try all the 26 possible shifts!

SUBSTITUTION CODES

The obvious step onwards from this is to employ a code whereby every letter of the alphabet is obtained from another by a unique letter, so that A,B,C,etc might be replaced by B,E,U, etc. It is common for this sequence to be obtained from the first by a code-word or phrase: such as "READ BEEBUG WORKSHOPS" which might be embedded in the substitution list as:

```
READ BUGWOKSHPCFIJLMNQTVX
```

(using each letter once only, and then following the (if necessary truncated) code-word with the rest of the alphabet) giving a substitution table of:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
READ BUGWOKSHPCFIJLMNQTVXYZ
```

and remembering to provide a substitution for 'space'. The message:

```
BUY BEEBUG
```

when encoded then becomes:

```
AQYRABBAQG
```

Sadly any decent code-breaker can handle a simple substitution scheme by looking at the frequency of letters and diagrams (two-letter combinations). Certain diagrams occur commonly in English (ER), some are impossible (QT), and repetitive ones can only be possible for a handful of combinations (EE, LL etc.). Provided the message is long enough, it's easy to crack.

VIGNERE CIPHERS

An improved coding method is based on shifting letters within the alphabet. Here the key word gives a distance through which we move a letter (rather like the Caesar cipher). Suppose our key word is BBC (normally a longer key is used), then counting B as the second letter of the alphabet we would move the letters alternately 2,2 and 3 places and repeat:

```
BBCBBCBBCB..
BUY BEEBUG then becomes:
DWABDHGDXI
```

This method of coding is called a *Vignere* cipher, and the key to the encoding and decoding is the keyword "BBC". The longer the word the better, of course, and we can even extend the key to phrases such as "READ BEEBUG WORKSHOPS" giving letter shift distances of 19,5,1,4,0,2 etc. Now when the key phrase is the same size as the message (or longer) then the code is provably unbreakable, in the sense that a code-breaker would have to attempt all possible keys of the length of the message - a process just as long as trying all possible messages and guessing the right one! This type of cipher is called a 'one-time pad' or *Vernam* cipher and is (was?) reportedly used for the Moscow-Washington hot line.

We have considered that all possible messages can be represented with the characters upper-case A to Z and space, but for the more efficient coding of binary data we can obviously extend this to the full ASCII set, or all byte values 0 to 255. In this case, the shift distances (as in "READ BEEBUG WORKSHOPS") could be the ASCII values of the characters: 82,69,65,68,32 etc. But instead of this addition algorithm, we can use the exclusive-OR operation on the source data so that the key using the 82 would make an ASCII 233 into 233 EOR 82, or 187. This also has the advantage that the decryption algorithm is identical to the encryption algorithm (187 EOR 82=233), and the interesting by-product that the EOR of the source and answer is the key! (233 EOR 187=82).

Clearly the security of any such system is completely dependent on the security of the key, and the longer the key the better the security. But sadly, long keys are less memorable and tend therefore to be written down: less security again! What we want is a memorable way of generating a long sequence of random-looking numbers from a short key (which we will call a proto-key), and we would use the new sequence as our actual key.

A suitable random sequence is actually a good random number generator: and the Beeb contains precisely that. Recall from last month that starting the RND function with a negative argument gives a repeatable random sequence, so we can use a proto-key of a (say) 9-figure number N, and successive calls to RND(256) to produce a binary sequence or calls of RND(27) to produce an alphabetic rotation key. The key thus produced is thus a true Vernam cipher, and the only decryption method open to the code breaker is the testing of each of the possible 9-figure proto-keys in turn. If we also keep the details of the random-number generator secret then even this method breaks down.

Listing 1, at the end of this article, is a program which allows the encoding and decoding of simple messages (converting to upper case) with alphabetic source, or the coding of complete binary files (program, data, text etc.) by use of a particular numeric proto-key. In the case of a message the shift method is used, but in the case of file encryption the source is EORed with the data.

PUBLIC-KEY ENCRYPTION

The secrecy of any message is only as good as the secrecy of the key, and long keys are vulnerable to mistakes and short keys are vulnerable to guessing. It has been a goal since the 1970s to produce an encryption system in which the encryption key was specific to each user and yet publicly known, so that messages to X could be encoded by anyone by looking up X's (public) encryption key. The decoding however would be private to X as it would require a decryption key quite unique to him and known by no-one else.

Such a system was devised in 1976 by Rivest, Shamir and Adleman and so is called the RSA public-key cryptosystem. The encryption key is essentially a pair of integers N and P. The decryption key consists of N and a further number S, kept secret. N, P and S must satisfy certain conditions:

(i) N=XY where X and Y are primes
(ii) S is prime
(iii) P is a solution of the equation:

```
PS MOD ((X-1)*(Y-1))=1
```

Now with these conditions satisfied, it can be mathematically proved that for any number M (essentially our message):

```
M^(PS) MOD N = M
```

or, equivalently:

```
(M^P)^S MOD N = M.
```

To illustrate this we will consider X=47, Y=79 and S=97 so that N=3713 and P=37 (found by trial and error - see algorithm below).

Now to encode "BUY BEEBUG" we can break it into two-digit sized pieces using the alphabetic position again:

```
0221 2500 0205 0502 2107
B U  Y    B E  E B  U G
```

Now each of these numbers, x, will be less than 3713 so we encode one each as:

```
x^37 MOD 3713
```

i.e.:

```
3372 1277 2265 3621 0702
```

(221^37 MOD 3713=3372 etc.).

Note that we don't need to work out 221^37 - it's too large for BBC Basic - we can perform the MOD 3713 at every calculation:

```
ANS=1
FOR i=1 TO 37:ANS=(ANS*221) MOD 3713:NEXT
```

To decode the message, the secret number S is used, and each section is raised to the Sth power MOD 3713 in an exactly similar way: 3372^97=221 etc.

Now this whole method of encryption depends on the fact that it must be difficult or impossible to deduce S from P. Now of course finding S from P is the same problem as finding P from S, the algorithm we used above to find an S, and here is one simple possibility:

```
10 INPUT X,Y,S
20 D=(X-1)*(Y-1)
30 N=1
40 REPEAT
50    N=N+D
60 UNTIL N MOD S=0
70 PRINT "P=";N DIV S
```

so that it would appear that we have no chance of keeping S secret, knowing P.

But this algorithm depends upon the knowledge of X and Y, and here is the crux of the matter: rather than the 2-figure example we used, X and Y are chosen to be 100 (or so)-digit prime numbers and kept quite secret. N is thus a 200-digit number and it is thought that a 200-digit number would take millions of years to factorise into two 100-digit primes by current machines. So publish N and a P for each user, keep your factorisation into X and Y secret and you have a workable public-key encryption system.

All you have to do is to implement a 100-digit integer arithmetic system for the Beeb. It's actually quite feasible but I leave that to you and your back copies of BEEBUG (see Vol.3 No.1 and Vol.6 Nos. 5 & 6).

Listing 1

```
 10 REM Program Encoding
 20 REM Version B1.0
 30 REM Author  Bernard Hill
 40 REM Beebug  Jan/Feb 1990
 50 REM Program subject to copyright
 60 :
100 DIM alf 27,D 255,M 255
110 $alf=" ABCDEFGHIJKLMNOPQRSTUVWXYZ"
120 INPUT "KEY NUMBER:" K%
130 dummy=RND(-K%)
140 IF NOT FNin(1E5,K%,1E9) THEN PRINT
"6-9 figures please for security":GOTO 1
20
150 PRINT "File encryption or trial me
ssage (F/M):";
160 ans$=GET$
170 IF INSTR("FfMm",ans$)=0 THEN 160
180 PRINTans$
190 file=INSTR("FfMm",ans$)<3
```

Continued on page 34

A PC Disc Formatter

by Kate Crennell

Here is a program which will format a PC compatible disc on a BBC micro. It is most useful for those who have no access to a PC and who wish to send out files on a PC disc, or for people like me who want to take files from a BBC at home to a PC at work and forget to bring home a PC formatted disc.

In either case the excellent programs of Bernard Hill (BEEBUG Vol6 No.10 & Vol.7 No.1) must be used to transfer files between Beeb and PC, but these are of no use if you have no PC formatted disc.

PC discs are recorded in double density with 9 sectors of 512 bytes on each track, and hence require the 1770 series disc interface to access them in your micro. The B+ and Master series computers have this as standard; if you have a BBC micro fitted with the older 8271 disc controller it will not be able to access PC discs (though you can upgrade your machine quite simply).

There are two standard PC formats (see ref.1) which can be written with this program: a version for 5.25" disc which uses 40 tracks on both sides (the 360K format), and a version standard on 3.5" discs which again uses both sides, but with 80 tracks (720K format). Hence you must also have a double sided drive. If you are using a 5.25" 80 (or 40/80 switchable) track drive, it is definitely best to format a completely blank disc so that there is nothing written on the intermediate tracks.

When you have typed in the program, there are two options coded in which you might wish to change:

1. If you have a 40 (only) track 5.25" drive, change line 230 to read U%=1, otherwise the program will expect an 80 track drive and use only every other track.

2. If your drive has trouble moving from track to track at the highest speed, change line 810 to give a larger value of V%: V%=&51 allows an additional 6ms for a track step, &52 and &53 give progressively longer times.

PROGRAM STRUCTURE

The main program calls PROCinit and PROCmenu which initialise your micro to access the 1770 disc controller, and control program flow. PROCmenu calls PROCdisktype, PROCdrive and PROCtitle to perform their simple and obvious functions.

The real work is done in PROCformat. Here the first ten lines set up an image of a complete PC disk track with its preamble, nine sectors with their ID fields, and the trailing gap. The PC format differs in several important ways from the BBC ADFS format described by David Spencer in the last of his Spin-a-Disc articles (see BEEBUG Vol.7 No.10 to Vol.8 No.3). The most obvious difference is that here there are 9x512 byte sectors whereas the ADFS has 16x256 byte ones. Other differences are a more complex preamble and 80 bytes inter-sector spacing on the PC disc. The track image is set up in the array buf% using PROCword and PROCbyte.

There follow nested loops over the two sides and the tracks to write the initial sectors to the disc. Here the 1770 controller is asked to perform tasks by storing commands in ?cmd%, and then PROCwait checks for a successful completion. Details of the commands, and how to access the 1770 have been described in David Spencer's articles and are not repeated here.

Finally, PROCformat writes fixed information into the first few sectors of the disc. The first sector contains 11 identifying bytes followed by 19 bytes describing how the disc is laid out (number of tracks, sectors/track etc.). Usually this would be followed by a boot program to load the PC system, but because there is no system on the disc, the rest of this sector is set to zero. Then follow two copies of the File Allocation Table (FAT) which will describe which sectors belong to each file, but are now zero (except for the first two special entries) because there are no files on the disc. A FAT is 2 (3) sectors long for 360K (720K) discs. Finally, the program writes 7 sectors of blank top directory. The optional title is put into the first pseudo-directory entry.

PROCverify simply reads all the sectors on both sides of the disc for all the tracks using PROCchk to read and check each sector.

PROCq terminates the program nicely, returning control of the 1770 to the original disc filing system. It is also called if the program finds an error or Escape condition.

RUNNING THE PROGRAM

After loading the formatter, take your program disc out (at least the first time you run!) and insert the blank disc to be formatted in drive 0 or 1. From the menu page select the type of disc and drive number using the cursor keys. The next item on the menu is the disc title. This is not displayed by Bernard Hill's programs, but will be shown on the PC when you ask for a directory. It is optional (carry on down the menu if you don't want one) but must not be longer than 11 characters terminated with Return. The last three lines control the action of the program: select the required line and press Return. The "Verify" function just checks that all sectors can be read; it does not check their contents.

REFERENCE

Details of the format of PC discs are given in the "MS-DOS Encyclopedia" Article 3; published by Microsoft Press; ISBN 1-55615-049-0.

```
   10 REM Program PCFormat
   20 REM Version B1.0
   30 REM Authors K.M. & D.J. Crennell
   40 REM BEEBUG  Jan/Feb 1991
   50 REM Program subject to copyright
   60 :
   70 MODE7:PROCinit
   80 ONERROR REPORT:PRINT" at line ";ER
L:PROCq
   90 PROCmenu
  100 :
  110 DEF PROCbyte(I%,N%):REM add N% byt
es of I% to buf%
  120 FORJ%=1TON%:?B%=I%:B%=B%+1:NEXT:EN
DPROC
  130 :
  140 DEF PROCchk:REM check sector
  150 cmd%?2=S%:?cmd%=&88:REPEATUNTIL(?c
md%AND1)=0:IF(?cmd%AND24)=0 ENDPROC
  160 PRINTTAB(0,16)"Error &";~?cmd%", t
rack ";J%", side ";F%", sector ";S%
  170 E%=E%+1:ENDPROC
  180 :
  190 DEF PROCdisktype:REM select and se
t up for 3.5 or 5.25 discs
  200 U%=T%DIV40-1:REPEAT:PRINTTAB(14*(1
-U%),4)n$;TAB(U%*14,4)i$;:K%=GET
  210 IF K%=136 OR K%=137 U%=1-U%
  220 UNTIL K%=138:T%=40*(U%+1):H%=2*T%D
IV3
  230 REM set U%=1 here for 40-track dri
ve ***
  240 ENDPROC
  250 :
  260 DEF PROCdrive:REM select disc driv
e- D%=0 or 1
  270 REPEAT:PRINTTAB(5*(1-D%),7)n$;TAB(
5*D%,7)i$;:K%=GET
  280 IF K%=136 OR K%=137 D%=1-D%
  290 UNTIL K%=138 OR K%=139
  300 FOR I%=0TO1:fc%(I%)=(fc%(I%)AND&FC
)+D%+1:NEXT
  310 ENDPROC
  320 :
  330 DEF PROCformat:REM formats disc
  340 PRINTTAB(0,24)r$"One moment please
...";n$;
  350 B%=buf%:C%=&4E4E4E4E:REM first set
 up track data
  360 PROCword(C%,19):PROCword(0,3):PROC
word(&FCF6F6F6,1)
  370 PROCword(C%,12):PROCbyte(C%,2):int
ro%=B%
  380 FOR S%=1TO9:PROCword(0,3)
  390 PROCword(&FEF5F5F5,1):PROCword((S%
+512)*65536,1):PROCbyte(&F7,1)
  400 PROCbyte(C%,2):PROCword(C%,5):PROC
word(0,3):PROCword(&FBF5F5F5,1)
  410 PROCword(&5A5A5A5A,128):PROCbyte(&
F7,1):PROCword(C%,20):NEXT
  420 slen%=(B%-intro%)DIV9:PROCword(C%,
256)
  430 FOR F%=0TO1:?ctrl%=fc%(F%):?cmd%=V
%AND3:REM reset to track 0
  440 PROCsetbyte(17,F%):REM set side no
. in buf%
  450 PROCwait
  460 PRINTTAB(3,24)"Side ";F%", track
";n$;
  470 FOR J%=0TOT%-1:PRINTTAB(17,24);J%;
  480 PROCsetbyte(16,J%):REM set track n
umber in buf%
  490 PROCnmi:IFJ%>H% ?cmd%=&F0 ELSE ?cm
d%=&F2:REM format track
  500 PROCwait
  510 REM move in to next track (twice o
n 80-track 5.25" drive)
  520 IF J%<T%-1 ?cmd%=V%:PROCwait:IFU%=
0 ?cmd%=V%:PROCwait
```

```
  530 NEXT,
  540 PRINTTAB(3,24)"Writing fixed secto
rs";n$;
  550 ?ctrl%=fc%(0):?cmd%=V%AND3:REM see
k track 0 side 0
  560 REM meanwhile, set up OEM id and B
IOS parameter block
  570 !buf%=&9034EB:K%=buf%+3:$K%="BEEBU
G01":K%=K%+8
  580 !K%=512:K%?2=2:K%!3=1:K%?5=2:K%!6=
112
  590 IF T%=40 K%!8=720:K%?10=&FD:K%!11=
2 ELSE K%!8=1440:K%?10=&F9:K%!11=3
  600 K%!13=9:K%!15=2:FORI%=28TO508STEP4
:buf%!I%=0:NEXT
  610 PROCwait:PROCwrite(1):REM header s
ector
  620 N%=K%?11:REM # sectors/fat
  630 !buf%=&FFFF00+K%?10:REM first 2 en
tries are reserved
  640 FOR I%=4TO24STEP4:buf%!I%=0:NEXT
  650 PROCwrite(2):PROCwrite(N%+2):!buf%
=0
  660 FORJ%=3TON%+1:PROCwrite(J%):PROCwr
ite(J%+N%):NEXT
  670 REM and now the blank top director
y
  680 K%=2*N%+2
  690 FOR J%=K%+1TOK%+6:PROCwrite(J%):NE
XT
  700 REM insert title (if any)
  710 IF title$<>"" $buf%=title$+STRING$
(11-LENtitle$," "):buf%?11=8
  720 PROCwrite(K%)
  730 ENDPROC
  740 :
  750 DEF PROCinit:REM initialize run co
nstants
  760 DIMfc%(1),buf% &1C00:D%=0:T%=40:M%
=INKEY(-256)
  770 IF M%=251 ORM%<1   cmd%=&FE84:ctrl
%=&FE80:M%=FALSE:fc%(0)=33:fc%(1)=37
  780 IF M%=245 ORM%=253 cmd%=&FE28:ctrl
%=&FE24:M%=FALSE:fc%(0)=5 :fc%(1)=21
  790 IF M% PRINT"sorry, no set up for t
his computer":END
  800 osbyte=&FFF4
  810 V%=&50:REM set V% to &51, &52, or
&53 for slower drives.
  820 c$=CHR$131+CHR$157+CHR$129:i$=CHR$
129+CHR$157+CHR$134
  830 r$=CHR$135+CHR$157+CHR$132:n$=" "+
CHR$156+CHR$135
  840 A%=143:X%=12:Y%=(USR(osbyte)DIV655
36):REM request NMI
  850 ENDPROC
  860 :
  870 DEF PROCmenu:REM control actions w
ith menu
  880 FOR I%=0TO1
  890 PRINTTAB(0,I%)CHR$132+CHR$157+CHR$
131+CHR$141+"      Formatting PC Disc"
  900 NEXT:*FX4,1
  910 PRINT'"   Disc type"n$'"    5{"" 36
0K "n$"  3\"" 720K "n$
  920 PRINT'"   Disc drive"n$'"    0 "n$"
1 "n$
  930 PRINT'"   Title:"n$:title$=""
  940 PRINT''"    Format "n$'''"    Verify
 "n$'''"   Quit "n$
  950 PRINTTAB(6,22)c$"Select with curso
r keys "n$
  960 L%=3:REPEAT:PRINTTAB(0,L%)r$;
  970 IF L%=3 PROCdisktype
  980 IF L%=6 PROCdrive
  990 IF L%=9 PROCtitle
 1000 IF L%>9 PRINTTAB(0,24)c$"Press RET
URN for action "n$;:K%=GET
 1010 IF K%=13 AND L%=12 PROCformat
 1020 IF K%=13 AND L%=15 PROCverify
 1030 IF K%=13 AND L%=18 PROCq
 1040 PRINTTAB(0,24)SPC(38);TAB(0,L%)n$;
 1050 IF K%=138 AND L%<18 L%=L%+3
 1060 IF K%=139 AND L%>3 L%=L%-3
 1070 UNTIL FALSE
 1080 :
 1090 DEF PROCnmi:REM initialise NMI to
write from buf%
 1100 FOR I%=0TO2STEP2:P%=&D00
 1110 [OPTI%
 1120     PHA:LDAcmd%:AND#31:CMP#3:BNEa2
 1130 .a1 LDAbuf%:STAcmd%+3:INCa1+1:BNEa
2:INCa1+2
 1140 .a2 PLA:RTI
 1150 ]:NEXT
 1160 ENDPROC
 1170 :
 1180 DEF PROCq:REM return NMI to BBC fi
ling system and end.
 1190 ?&D00=64:X%=11:CALL osbyte
 1200 *FX4,0
 1210 END
 1220 :
 1230 DEF PROCsetbyte(K%,J%):REM set byt
e K% in each sector to J%
 1240 FOR I%=1TO9:intro%?K%=J%:K%=K%+sle
n%:NEXT
 1250 ENDPROC
 1260 :
 1270 DEF PROCtitle:REM get optional tit
```

```
le for disc
 1280 PRINTTAB(0,24)c$;"Enter optional t
itle <12 chars";
 1290 a$=title$+STRING$(11-LENtitle$," "
)
 1300 PRINTTAB(6,10)i$;a$;CHR$156;TAB(9,
10);
 1310 K%=GET:IFK%>127 AND title$="" PRIN
TTAB(6,10)n$
 1320 IF K%>127 ENDPROC
 1330 OSCLI"FX138,0,"+STR$K%
 1340 INPUT""a$:IFLENa$<12 title$=a$:K%=
138 ELSE VDU7
 1350 PRINTTAB(9,10)SPC(29);TAB(9,10)tit
le$+" "+n$
 1360 ENDPROC
 1370 :
 1380 DEF PROCverify:REM verify all sect
ors can be read
 1390 E%=0:?cmd%=V%AND3:?&D00=64:PROCwai
t
 1400 PRINTTAB(0,24)r$"track   "n$;SPC(1
5);
 1410 FOR J%=0TOT%-1:cmd%?1=J%:PRINTTAB(
9,24);J%;
 1420 FOR F%=0TO1:?ctrl%=fc%(F%)
 1430 FOR S%=1TO9STEP2:PROCchk:NEXT
 1440 FOR S%=2TO8STEP2:PROCchk:NEXT
 1450 NEXT
 1460 IF J%<T%-1 ?cmd%=V%:PROCwait:IFU%=
0 ?cmd%=V%:PROCwait
 1470 NEXT:?cmd%=V%AND3
 1480 PRINTTAB(0,16)SPC(38);TAB(3,16)i$;
 1490 IF E%=0 PRINT"No"; ELSE PRINT;E%;
 1500 PRINT" errors";n$:PROCwait
 1510 K%=GET:PRINTTAB(0,16)SPC(39)
 1520 ENDPROC
 1530 :
 1540 DEF PROCwait:REM wait for end of d
isc access
 1550 REPEATUNTIL(?cmd% AND1)=0:IF(?cmd%
AND24)=0ENDPROC
 1560 PRINT"Error ";?cmd%''"Track ";J%",
  Sector ";S%:PROCq
 1570 :
 1580 DEF PROCword(I%,N%):REM add N% wor
ds of I% to buf%
 1590 FOR J%=1TON%:!B%=I%:B%=B%+4:NEXT:E
NDPROC
 1600 :
 1610 DEF PROCwrite(S%):REM write to sec
tor S%
 1620 IF S%>9 cmd%?2=S%-9:?ctrl%=fc%(1)
ELSE cmd%?2=S%:?ctrl%=fc%(0)
 1630 PROCnmi:?cmd%=&A6:PROCwait
 1640 ENDPROC
```

BEEBUG Workshop - Cryptology (continued from page 30)

```
  200 IF file THEN PROCfile ELSE PROCmsg
  210 END
  220 :
 1000 DEF PROCfile
 1010 INPUT" Input file name:"in$
 1020 f=OPENINin$:IF f=0 THEN PRINT"Not
found":GOTO 1010
 1030 INPUT"Output file name:"out$
 1040 g=OPENOUTout$:PRINT"Encoding file.
.."
 1050 REPEAT
 1060 X%=BGET#f:BPUT#g,X% EOR RND(256)
 1070 UNTIL EOF#f:CLOSE#0
 1080 PRINT"The same key value will deco
de the file"
 1090 ENDPROC
 1100 :
 2000 DEF PROCmsg
 2010 PRINT "Encrypt or decrypt (E/D):";
 2020 ans$=GET$
 2030 IF INSTR("EeDd",ans$)=0 THEN 2020
 2040 PRINTans$
 2050 IF INSTR("EeDd",ans$)<3 THEN sign=
1 ELSE sign=-1
 2060 INPUT LINE "Message:"$M
 2070 $M=FNuppercase($M)
 2080 PROCencode:PRINT $M
 2090 PRINT"Decoding for testing..."
 2100 sign=-sign:dummy=RND(-K%)
 2110 PROCencode:PRINT$M
 2120 ENDPROC
 2130 :
 3000 DEF PROCencode
 3010 FOR i=0 TO LEN$M-1
 3020 p=(INSTR($alf,CHR$M?i)+26+sign*RND
(27)) MOD 27 + 1
 3030 M?i=ASCMID$($alf,p,1)
 3040 NEXT:ENDPROC
 3050 :
 4000 DEF FNuppercase($&A00)
 4010 LOCAL i,P:P=&A00
 4020 FOR i=0 TO LEN$P-1
 4030 IF FNin(97,P?i,122) P?i=P?i-32
 4040 IF NOT FNin(65,P?i,90) AND P?i<>32
THEN P?i=32 : REM other symbols=space
 4050 NEXT:=$P
 4060 :
 5000 DEF FNin(a,b,c)=a<=b AND b<=c
```

1st course

Data Storage, Data Dictionaries

By Mike Williams

Many programs (I might almost claim all programs) store and process data. In many cases too, data used when a program is run is often needed again the next time the program is used. Sometimes the data will remain unchanged; sometimes the data will need to be modified. What I want to do in this article is to consider some of the ways in which data can be stored (and modified) so that it can be used by a program whenever it is run.

USING DATA STATEMENTS

One of the most obvious and widely used ways of supplying data to a program is through the use of Basic's DATA statement. This is ideal for relatively small amounts of data, particularly where that data is constant and not subject to possible amendment. For example, a program might need to know the names and lengths (in terms of days) of each month in the year. This could be easily accommodated by writing:

```
DIM Mname$(12),Mlen(12)
FOR I=1 TO 12
READ Mname$(I),Mlen(I)
NEXT I
DATA January,31
DATA February,28
DATA March,31
DATA April,30
DATA May,31
DATA June,30
etc.
```

Even here there is a small problem, because the number of days in February is not fixed. Provided the program using this information knows the year number, it can calculate whether or not a leap year is involved and add one on to the number of days in the DATA statement for this month if necessary.

It is often in the use of small amounts of constant data like this that the DATA statement is most useful. Notice, though I am sure you are already aware of this, that it is READ which is used to read the data, not INPUT which specifically refers to keyboard input.

USING RESTORE

In any program using DATA, there is a data pointer, which is always initialised to point to the very first item in the complete set of DATA statements. As the data is read so this pointer is moved through the data items.

Sometimes you may need to move this data pointer yourself to a specific item. This is accomplished with the RESTORE statement which specifies a new line number for the pointer. Personally, I have found little use for this. Since the constant data of my example is only intended to be read once by the program, there is no need for any RESTORE instruction, to tell Basic where to start reading the DATA.

Of course, if the data is to be read more than once in the course of running a program, then RESTORE will be needed each time to move the data pointer back to the startof the data, but such an occurrence would seem unlikely.

A circumstance where RESTORE might be useful would be a program which contained, say, three alternative sets of similar data, one set being chosen for use each time the program is run depending on other information supplied at the time by the user. For example, you could write:

```
IF Set=1 THEN RESTORE 3000 ELSE
IF Set=2 THEN RESTORE 4000 ELSE
IF Set=3 THEN RESTORE 5000
```

assuming that equivalent sets of DATA statements begin at lines 3000, 4000, and 5000 respectively. For example, a program might work in either imperial or metric units, and need to read a corresponding set of constants at the outset.

There is an alternative way of handling this requirement, but one which is not without its dangers, and this is the computed RESTORE. For example, the IF statement given above could be replaced by:

```
RESTORE 1000*(Set+2)
```

This is much shorter, but the danger with all computed RESTOREs is that if you renumber the program at all, Basic cannot cope with changing the expression following RESTORE. If you reference a single line number after a RESTORE then renumbering will adjust this correctly. My recommendation would be to avoid using RESTORE in the first place, unless really essential, and to think very carefully indeed before using a computed RESTORE. If you do, remember that every time you renumber your program you *must* manually adjust any computed RESTOREs to compensate or the program will fail (as we have sometimes found when editing programs for the magazine).

What if your data requirements are not constant? What if you do need to modify the data from time to time? Well to some extent you can still cope with this using DATA, though it is not my ideal way of doing it. A problem which often arises with storing data for use by a program, is how you cope with updating that data. Although it is not too difficult to write a program to create a data file, it is a much more demanding task to write a program which will update a data file (see our new book *File Handling for All* for a more detailed discussion of this and other problems with file handling).

One of the apparent beauties of using DATA statements in a program, is that they can be edited and the data changed just as easily as with the program itself. Indeed I have seen a data handling application, where all the data records were kept as DATA statements at the end of the program, adding, amending and deleting data records as required. It works, but once again it does have its dangers.

Editing data in this way is quite unstructured. The facilities which you are taking advantage of, i.e. the Basic line editor, has no knowledge of your data. If you make any mistake this will only come to light when the program is run. There is also the danger that other data might be accidentally modified at the same time, or even that the program itself might become corrupted.

In general, DATA statements are ideal for constant data which is needed every time a program is run. They are also best suited to relatively small amounts of data, because there is another disadvantage. The data included with a program in the form of DATA statements clearly uses up some of the computer's memory when the program is loaded. That is to be expected. However, when the program is run, and the data is read from the DATA statements into variables or arrays, even more memory is then required. In effect, any data supplied in DATA statements and read into a program then uses up double the amount of memory space it really needs. For small amounts of data the convenience outweighs any other consideration, but with large amounts of data, this is really quite wasteful.

For example, suppose you are writing a game program and want to display the instructions on screen. Such instructions could be quite extensive (think of adventure games for example). It is true, of course, that you are only likely to read a section at a time, in order to display it on the screen, so you won't be doubling up on all the text data as far as storage is concerned. Indeed, you might not choose to use DATA statements at all, but simply include the text in a lengthy sequence of PRINT statements. Often this will work, and if it does, fair enough. But the model B, in particular, has only a limited amount of memory, particularly in the higher resolution screen modes. Apart from potentially doubling memory requirements when held as DATA statements, extensive displays of text may simply occupy more memory space than you can afford if included with the program.

The solution, of course, is to save such text separately from the main program. With many simpler games, one program can display the instructions on the screen before chaining to a second program which plays the game. This is also useful with things like character definitions, because the BBC micro has a reserved area of memory in which to store such definitions. This does not detract in any way from user memory, but the program to create the definitions can need quite a lot of space. Once again, include these definitions in a first program which chains the main program after the required characters have been defined.

However, incorporating text in a separate program which is run first is not always a feasible answer to the problem. One solution to consider is to use one of the many word processors or editors available for the BBC micro. Most users have such an application on their machine, and this provides an ideal way to create *and update* a data file which can then be read by your own program. Indeed some commercial database programs adopt this approach, to create templates for screen displays and printed reports where the concept is called a *data dictionary*.

DATA DICTIONARIES

This approach can be excellent for 'constant' data which will be needed by your program. The contents can be updated as required with the word processor or editor with which it was created, but it is normally less satisfactory for the application program to attempt to update this file itself. As far as this is concerned the data file is 'read only'.

All you need to do is to decide upon a suitable and simple format for your data file, and make sure that when you create it (or edit it) you don't insert any embedded commands or other information. You also need to remember that such files are really text files, so any numeric data will have to be read as character strings initially and then converted to numbers inside your own program.

The simplest format is to have just one data item per line, terminated by a Carriage Return (ASCII code 13). This does mean that a Return character cannot be included as part of any data, but since you know this, you can easily cope with any consequential problems.

You will need to include a statement in your program to open the file ready to be read, for example:

```
F1=OPENUP("MyData")
```

You could then use the following function to read the next data item from the file:

```
1000 DEF FNread_data(F)
1010 LOCAL char$, string$:string$=""
1020 REPEAT
1030 char$=BGET#F
1040 string$=string$+char$
1050 UNTIL char$=CHR$(13)
1060 =string$
```

In a program you could then write:

```
text$=FNread_data(F1)
```

for reading text (character strings), and:

```
number=VAL(FNread_data(F1))
```

for reading numeric data.

What I have written here is just by way of example. You determine the format of the data file, so you can write your program to fit in with this and handle the data as you wish. My routine includes the last character, the Return itself in the string
- you might not wish to do this, in which case replace line 1040 with:

```
1040 IF char$<>CHR$(13) string$=string$+c
har$
```

In the system I have described, our data pointer again starts at the beginning of the text data file, and moves through the file as the data is read. There is no direct equivalent of the RESTORE instruction to use here, though there are ways of moving to any particular location in the file (using the instruction PTR#). However, we do not have the space to discuss this for now. Incidentally, the *Dynamic Footnotes* program published in this issue is a good example of the technique described here, and uses an index to locate individual sections of text.

One final point needs to be made. Most word processors and editors, like Wordwise, View and Master Edit, are suitable tools for this purpose. If in doubt, create a short sample data file and then use the *DUMP command to examine the layout of its contents.

Overall, a text data file created with your familiar word processor is an ideal solution to the problems of limited memory space coupled with ease of editing, and is a technique which is often overlooked by many users.

512 Forum

by Robin Burton

I thought this month I'd deal with a number of points which, according to my mail, have caused a bit of trouble for quite a few of you from time to time.

FILE ATTRIBUTES

Most of us know what file attributes mean in the 512, but even so it appears that there are a few common misunderstandings for new users, particularly when they are carrying out certain operations.

File attributes in most operating systems share common ideas, but there are of course also differences. For example, in the Beeb's DFS a file can be locked or not, and that's it. In ADFS this idea is extended, so a file can be set to read only, write only, read and write or execute. In addition the first three can also have the lock attribute set too. (I've never quite been able to work out what use a locked write-only file is! Does anyone have any ideas?)

DOS too uses file attributes, again with similarities but also with differences of its own. Before I go on let me say that I know that most of these facts will be known to many of you, but as I've said before they're not all obvious to new users of the 512.

SYSTEM FILES

The first attribute which can cause some consternation is the 'system' attribute, because in DOS it makes a file invisible on the normal directory display and it also prevents it from being copied in the usual manner.

The problem for new 512 users is of course that there's nothing like it in BBC micro filing systems so it's a completely new idea. Strictly speaking, the system bit doesn't actually prevent files from being copied, so this is where the confusion arises.

The system attribute actually prevents any file with this bit set from being included in normal directory searches, hence a straightforward 'COPY' command for a file set to 'system' produces a simple 'file not found' instead of something more helpful.

One occasion when this can be particularly annoying is when you use wild cards in a copy command. DOS will copy all the normal files, omitting the system files, but it won't report anything amiss because there's no error. Only later might you realise that some of the files have been missed, particularly because they're not displayed normally anyway. Of course it's easy to avoid the problem by using a modifier with the copy command, so if you thought you had to set all the system files to 'DIR' before they could be copied (and perhaps set back to 'system' later on the new disc) you'll be pleased to know there's a much easier way.

If you want to copy all the system files from one disc to another, the copy command can have a '/S' modifier added onto the end, just like the directory command, so that, assuming drive A: is the current drive:

```
COPY *.* /S B:
```

will copy all the system files in the current directory to the current directory of drive B:, and there's no need to change the file attributes at all. However, bear in mind that this modified command will copy only system files, so you'll still need a normal 'COPY *.* B:' as well.

FSET

Another attribute which has frustrated some users is the read-only setting, shown as RO for short in DOS. There's nothing mysterious about it, it means exactly the same as 'locked' in DFS or ADFS. However, whereas these filing systems will helpfully report 'File locked' when you try to overwrite a locked file, DOS is less informative. DOS reports instead 'Access denied', which means exactly the same as 'File locked', but which is by no means as clear about the precise situation as it could be.

Suffice it to say that if you get the 'Access denied' message it always means that you're trying to modify a file (or disc) which is set to read only; there is no other cause. There's only one course of action to take and that is to set the target file to read-write. This sounds simple enough, but it has caused a bit of hair-pulling for a few users. 'FSET.CMD' is the utility you need and it's quite simple to use so long as you are aware of its peculiarities.

I have to admit that until a few months ago I hadn't thought about one of FSET's oddities. It had never bothered me simply because of the way my system is set up, so perhaps quite a number of you are in the same situation without knowing it. I always have FSET in a current PATH setting, so it can always be called from any directory on any drive in my system, and normally it's within the current working directory that I wish to amend file attributes.

However, if this isn't the case FSET simply will not work. Suppose you're in a directory called 'SPRDSHT' and you want to copy your updated masterpieces to a directory called 'SHEETBAK', but all the files in the backup directory are set to read-only. Obviously they must be set to read-write before the copy can take place. You'll find however, that in this situation you cannot enter, for example:

```
FSET B:SHEETBAK\*.* [RW]
COPY *.* SHEETBAK
FSET B:SHEETBAK\*.* [RO]
```

because FSET doesn't allow directory names to be included in a file specification. This looks like a bug, but it's not. It's caused by the fact that CMD files, (including FSET.CMD) are CP/M utilities, and CP/M simply does not have subdirectories. The problem for DOS Plus users is that the program wasn't updated for DOS use and it still doesn't understand directories, so you can only run it on files in the current directory. To carry out the above operation, therefore, the commands needed are actually:-

```
B:
CD \SHEETBAK
FSET *.* [RW]
COPY A:SPRDSHT\*.*
FSET *.* [RO]
A:
```

assuming that you want to finish up in the original directory on the original drive, hardly convenient! Unfortunately this time there's no simple solution, you must just be aware of and allow for FSET's limited intelligence.

While we're looking at FSET there's one more point of which I've realised many people are not aware. Given FSET's limitations anything that makes life a bit easier is welcome, so I hope this saves one or two of you a bit of typing from time to time.

FSET can be used to change a file to read-only from read-write (or vice-versa) or from system to directory and back. The read-only attribute can be set on a file along with the read-only attribute at the same time, so when you want to copy such files to a different disc which already contains old versions of the same files with these attributes set you might think that you have two attributes to alter before starting the copy.

I heard from a member a while back about this very subject, and it was obvious from his letter that to perform this sort of operation he was issuing two separate FSET commands, when one can be used. To set a read-only-system file to read-write and directory the two changes can be specified together in one command, so using COMMAND.COM as an example the entry would be:

```
FSET COMMAND.COM [RW/DIR]
```

or changing it back again:

```
FSET COMMAND.COM [RO/SYS]
```

There's no need to issue separate commands for two changes, just separate the attributes by a '/' and FSET will apply them both in the one operation.

One more point frequently not realised is that 'SYS' and 'RO' do not need to be set together, nor do 'DIR and 'RW'. It's quite possible to have a 'RO/DIR' or a 'RW/SYS' file. Use 'SDIR' if you want to see all the different types of file attributes used in any directory.

By the way, have you noticed that you can miss off the trailing bracket from FSET commands? It seems that, having found a valid set of parameters preceded by a left square bracket FSET isn't bothered whether a right square bracket or a RETURN follows.

MOVE

MOVE.EXE is another command which seems to cause more than its fair share of problems for some people. The command is actually quite simple if you understand it, though it must be acknowledged that it can require a pretty lengthy command line at times.

From what I hear of users' problems with MOVE it seems again that several particular problems crop up more often than others. The first of these we've indirectly covered, that is that no matter which filing systems are involved and in which direction the transfer is to be, the target files must not be locked or read-only. That's pretty obvious, but it's also very easy to overlook, it seems.

The second most common difficulty is also self-inflicted, but this time there's more of an excuse. From my experiences it appears that some of you try to make the command more complicated than it needs to be, and not surprisingly you get problems.

For illustration let's suppose that you want to transfer a DFS file called 'LETTER' to DOS so you can import it into a DOS text file. The first point is that MOVE.EXE itself must be available in a current path. Again this is simple enough, but I frequently hear of users trying to copy files from DFS or ADFS drive :0 when the currently selected DOS drive is A: (think about it!)

While I remember it I'll mention another situation which has foxed a few. A problem for floppy disc only systems is copies between DFS and ADFS, when of course there isn't a 'spare' drive for DOS. Well actually there is, but you need to set up a RAM disc in this case, then select that as your current DOS drive, ensuring that both MOVE.EXE and COMMAND.COM have been copied to it. You can then use both floppy drives for BBC micro filing system transfers with no trouble.

Back to the unnecessary complexities. By far the most frequent is that users include the root directory in a file specification. MOVE frequently doesn't like it!

OK, this is less obvious, but with a bit of thought it's clear that in DFS, ADFS or DOS everything is subordinate to the root directory, whether it's a file or another directory. Once you've accepted that point it's obvious that there's no need to specify the root directory in the command unless it is the only parameter. Remember in MOVE the source specification can be either a list or a file, so if you MOVE a list of files from the root or the current directory, only the file list spec. is needed, '*' or '*.*' as appropriate.

Using our example file, to copy it to the current directory of drive A: in DOS the source disc must obviously be drive :1 (or perhaps :3 for DFS), so assuming that the file is in the root directory of the BBC ADFS disc, the command would be:

```
MOVE :1.LETTER -ADFS A: -DOS
```

or, even shorter:

```
MOVE :1.LETTER -ADFS . -DOS
```

so the command can be extremely simple (if you're not familiar with the use of the '.' in the above command look back over the last few Forums for an explanation). If we had a list of files called 'LETTER1', 'LETTER2' and so on the command simply changes to:-

```
MOVE :1.LETTER* -ADFS . -DOS
```

If there is a need to supply a directory path in either part of the command just specify it in the usual way for the filing system concerned, for example:

```
MOVE :1.LETTER -ADFS \DOCS -DOS
```

will move all the 'LETTER' files to directory 'DOCS' on the current DOS drive, instead of using the current directory. Likewise:

```
MOVE :1.TEXT.LETTER -ADFS A: -DOS
```

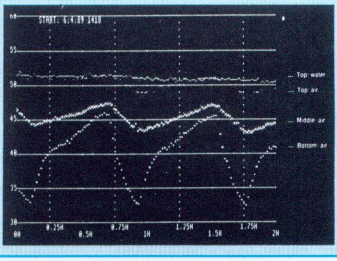
would move all the 'LETTER' files from directory 'TEXT' of ADFS drive 1 to our current DOS directory in drive A:. There's one other item which I must mention, since it has tripped up a surprisingly large number of people, and that is the filing system should always be specified in both the source and the target portion of the command, there are *no defaults*.

The final problem I'll mention with MOVE is really trivial, but it is certainly one of the most frequent mistakes I hear about. Please note, the filing system identifier for transfers to or from DFS is '-DISC'. It is NOT '-DFS'!

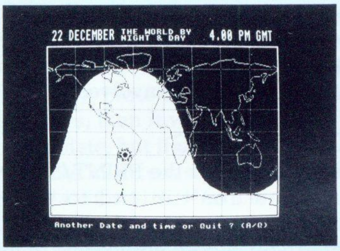
Special Offers!
Best of BEEBUG software

Applications II Disc

Monthly Desk Diary - a month-to-view calendar which can also be printed
Crossword Editor - for designing, editing and solving crosswords
Real Time Clock - a real time digital alarm clock displayed on the screen
Running Four Temperatures - calibrates and plots up to four temperatures
Label Processor - for designing and printing labels on Epson compatible printers
3D Landscapes - generates three dimensional landscapes.
Foreign Language Tester - foreign character definer and language tester
Julia Sets - fascinating extensions of the Mandelbrot set
Share Investor - assists decision making when buying and selling shares.



Applications I Disc



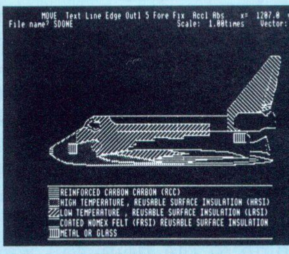
22 DECEMBER THE WORLD BY NIGHT & DAY 4.00 PM GMT
Another Date and time or Quit ? (A/Q)

Business Graphics - for producing graphs, charts and diagrams
Video Cataloguer - catalogue and print labels for your video cassettes
World by Night and Day - a display of the world showing night and day for any time and date
Phone Book - an on-screen telephone book which can be easily edited and updated
Page Designer - a self contained page-making package for Epson compatible printers
Personalised Letter-Headings - design a stylish logo for your letter heads
Appointments Diary - a computerised appointments diary
Mapping the British Isles - draw a map of the British Isles at any size
Selective Breeding - a superb graphical display of selective breeding of insects
The Earth from Space - draw a picture of the Earth as seen from any point in space
Personalised Address Book - on-screen address and phone book

ASTAAD

Enhanced ASTAAD CAD program for the Master, offering the following features:

✷ full mouse and joystick control
✷ built-in printer dump
✷ speed improvement
✷ STEAMS image manipulator
✷ Keystrips for ASTAAD and STEAMS
✷ Comprehensive user guide
✷ Sample picture files



MOVE Text Line Edge Outl 5 Fore Fix Accl Abs x= 1207.0 y= 606.0
File name? SDONE Scale: 1.00times Vector: 145.7
REINFORCED CARBON CARBON (RCC)
HIGH TEMPERATURE, REUSABLE SURFACE INSULATION (HRSI)
LOW TEMPERATURE, REUSABLE SURFACE INSULATION (LRSI)
COATED NOMEX FELT (FRSI) REUSABLE SURFACE INSULATION
METAL OR GLASS

EDIKIT ROM

An indispensible utility ROM for all Basic programmers, containing the following commands:

***FTEXT** (find text) ***FBASIC** (find Basic) ***FPROCFN** (find procedure/function)
***LPROC** (list procedure) ***LFN** (list function) ***LFROM** (list 8 lines of a program)
***RTEXT** (replace text) ***RBASIC** (replace Basic) ***SYSINF** (system information)
***VARLIST** (list program variables) ***FKDEFS** (function key definitions)

Incorporating the updated Basic Booster utilities:

SUPER SQUEEZE PARTIAL RENUMBER PROGRAM LISTER
RESEQUENCER SMART RENUMBER TEXTLOAD AND TEXTSAVE

General Utilities Disc

✷ Printer Buffer * ✷ Sprite Editor/Animator
✷ Mode 7 Screen Editor
✷ Epson Character Definer
✷ ROM Filing System Generator
✷ Multi-Column Printing
✷ Multi-Character Printer Driver for View
✷ ROM Controller ✷ BEEBUG MiniWimp †

*** Master series only. † Requires sideways RAM.**

	Stock Code	Price		Stock Code	Price
ASTAAD (80 track DFS)	1407A	£5.95	**ASTAAD** (3.5" ADFS)	1408A	£5.95
EDIKIT (EPROM)	1451A	£7.75			
EDIKIT (40/80T DFS)	1450A	£5.75	**EDIKIT** (3.5" ADFS)	1452A	£5.75
Applications II (80 track DFS)	1411A	£4.00	**Applications II** (3.5" ADFS)	1412A	£4.00
Applications I Disc (40/80T DFS)	1404A	£4.00	**Applications I Disc** (3.5" ADFS)	1409A	£4.00
General Utilities Disc (40/80T DFS)	1405A	£4.00	**General Utilities Disc** (3.5" ADFS)	1413A	£4.00

Please add p&p - 60p for the first item and 30p for every additional item.

BEEBUG Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS. Tel. (0727) 40303 Fax. (0727) 860268

MikroTel (Part 2)

Mike Bryant describes the remaining features of his powerful and versatile telephone database.

The last issue presented the main program and the basic features of MikroTel. In this issue we cover some of the more detailed features that make MikroTel complete.

You will need to enter Listing 1 and append this to last month's program. The best way to do this is to type the new coding in and save it separately. Now create a spooled version (using the *SPOOL command), load in the original program and *EXEC in this month's additions. In all of this keep strictly to the line numbers as printed. Save the completed version as *MikroTel*. You will also need to type in Listing 2. Save this before running it to assemble a machine code routine called *Compact* which is needed by MikroTel

THE NEW OPTIONS - FIND

All the options displayed in the main menu are now available. The FIND option can be run from either the main menu or from BROWSE, and enables you to locate a particular record, provided you know one of the fields. When you run Find, you will be asked for the surname of the record that you want to find. Either enter this directly, or, if you don't know the surname, press Return and you will be asked if you know the next field. Just keep pressing Return until you come to the field you want. As with everything in this program, if you make a mistake that you can't see how to correct, just press Escape and try again. Note: due to the storage of surnames in memory and all other data on disc, it is far quicker to find a record by surname than by any other field.

THE SEARCH COMMAND

MikroTel also incorporates a wildcard search, called SEARCH. This, too, can be run from either the main menu or via BROWSE. The SEARCH option searches through the entire hash table and data file to try and find, anywhere, a match to the entered string. It is thus quite a lot slower than the FIND option, which will search only the specified field. Note also that the SEARCH option will, if you're searching for 'HIGH' for example, consider '48 HIGH ST.' a match, whereas FIND would not.

COMPRESSING DATA

COMPACT, quite simply, compresses the data file so that it takes up less disc space. It does this by ensuring that all records follow each other consecutively on disc, eliminating any gaps between that may have been caused by deletion. Due to the slowness of Basic in handling this kind of operation, most of this option is performed in assembler, but it is still fairly slow because of the disc access time involved.

PRINTING/SPOOLING

By far the largest option is the PRINT/SPOOL option, and if you don't intend to use it, MikroTel has been designed such that it will run without it. If you choose not to type in this option, then you will not require PROCselectorder or PROCselectrecords either. This option is, by necessity, intricate due to the large number of different ways you may require your data to be printed, and when running it you will be asked a number of questions. These are explained below.

Firstly, you are asked whether to spool (create a text file for loading into a word processor, etc.) or print. This should be fairly straightforward. Next you will be asked which records you want printed (or spooled). If you want them all, then all well and good, else you will then have to go through each record one by one (using the cursor keys in much the same way as for the BROWSE option), and press 'P' (or 'S') to select the record displayed in the box at the top of the screen. If you make a mistake, just go back and press 'D' to de-select.

Once this is over, you next need to define which fields you want to be printed, and in what order. Thus, if all that you wanted were first names and then surnames, you would type '21', and then press Return. If you want all of the fields to be printed in their normal order, then you do not have to deal with this. Lastly come the questions on spacing. You have two choices: to print records with a certain number of spaces between them (e.g. for a telephone address book), or to print blank lines between them such that each record takes up a certain number of lines (e.g. for address labels). Only after all questions have been answered does any printing commence, so that if you enter something incorrectly you have only to press Escape and try again.

SORTING RECORDS

The SORT option will sort the file into alphabetical order using whichever field you specify (note, though, that the positions of the data on disc do not change, merely the order of the surnames in the hash table - this takes far less time). Note that if you sort into order of surnames, then it will take about a second or so to complete, but if you sort by any other field, it may take in excess of an hour, with around 100 records or so, due to the different ways in which surnames are stored compared with all other fields. Also note that, normally, records are automatically sorted into surname order whenever they become out of order, because of the auto-sort flag (see STATUS for more information). This happens so quickly that it is barely noticeable. As such, there is little value in sorting records via this option, but it is included anyway for the sake of completeness.

CHECKING STATUS

Lastly, the STATUS option tells you about some of the more technical parts of your current data file, and enables you to turn on/off (by the cursor keys) the 'Duplicate warning' operation, which will warn you if you try to ADD a record with the same surname as one already stored in the file. The other information shown is how many records are stored on disc at the moment, how much disc space this takes up, and how much memory is still free within the computer.

Listing 1

```
   10 REM Program Mikrotel2
 1290 DEF PROCcompact
 1300 LOCAL A%,I%,J%,O%,A$
 1310 !&70=0:!&74=0
 1320 PROChead("COMPACT DATA FILE")
 1330 PRINT''"This option tidies up the
stored record data so that it takes  up
 less disc"'"space."'"This may take a li
ttle while."
 1340 PRINTTAB(0,9)"Press C to compact,
or R to return to the main menu."
 1350 A$=FNget("CR")
 1360 IF A$="R"THENENDPROC
 1370 PRINT''"Compacting"STRING$(NR%,"."
);
 1380 O%=!(FNh(NR%+1)+14)AND&FFFF
 1390 X%=NR%MOD256:Y%=NR%DIV256
 1400 A%=X:*FX200 1
 1410 CALL &900
 1420 REM Update freespace pointer
 1430 ?(FNh(NR%+1)+14)=?&74
 1440 ?(FNh(NR%+1)+15)=?&75
 1450 *FX200 0
 1460 PRINTCHR$11'"Bytes saved :"O%-(!&7
4)AND&FFFF
 1470 PROCsavehash
 1480 ENDPROC
 1490 :
 1540 DEF PROCprint
 1550 LOCAL I%,J%,P%,rl%,rlt%,bl%,B$
 1560 PROChead("PRINT / SPOOL RECORDS")
 1570 PRINT''CHR$130"Do you want to dump
 records to a"'CHR$130"printer or to an
ASCII file?  (P/F)"CHR$135;
 1580 A$=FNget("PF")
 1590 PRINTA$
 1600 IF A$="P"THENP%=TRUE:B$="Print"ELS
EB$="Spool":PRINT'CHR$130"Enter filename
 :"CHR$135;:INPUT""C$
 1610 PRINT'CHR$130B$;" every record?  (
Y/N)"CHR$135;
 1620 A$=FNyn
 1630 IF A$="Y"THENFORI%=&5A00TO&5BFCSTE
P4:!I%=TRUE:NEXTELSEPROCselectrecords(B$
)
 1640 PRINT'CHR$130B$;" every field?   (
Y/N)"CHR$135;
 1650 A$=FNyn
 1660 IF A$="Y"THENPRINT'CHR$130B$;" in
normal field order?  (Y/N)"CHR$135;:A$=F
Nyn:IF A$="Y"THEN FORI%=0TO11:fo%(I%)=I%
:NEXT ELSE PROCselectorder(B$)
 1670 PRINT'TAB(2)CHR$130"Fixed number o
f "B$"ed lines per"'TAB(2)CHR$130"record
?             (Y/N)"CHR$135;
 1680 A$=FNyn
 1690 IF A$="Y"THENPRINT'CHR$130"How man
y lines per record?   "CHR$135;:INPUT""b
l%ELSEPRINT'CHR$130"How many blank lines
 between records?":INPUT""bl%
 1700 PRINT'CHR$129CHR$136B$"ing."
 1710 IF P%THENVDU2ELSEOSCLI("SPOOL "+LE
FT$(C$,7))
 1720 FORI%=1TONR%
 1730 IF I%?&5A00<255THEN1790
 1740 rlt%=rl%:PROCload(I%)
 1750 FOR J%=0TO11
 1760 IF fo%(J%)<0THEN1770ELSEIForig$(fo
%(J%))>""THENPRINTorig$(fo%(J%)):rlt%=rl
t%-1
 1770 NEXT
 1780 IF bl%>0THENFORJ%=1TObl%:PRINT:NEX
TELSEIFrlt%>0THENFORJ%=1TOrlt%:PRINT:NEX
T
 1790 NEXT
 1800 VDU3:*SPOOL
 1810 PRINT''"Press any key to return to
 menu..."
 1820 IF INKEY350:ENDPROC
 1830 :
 1940 DEF PROCsearch(quiet%)
 1950 LOCAL I%,F%,start%,A%
 1960 PROCbotf("Search")
 1970 PRINTTAB(3,15)"Search for what str
ing?"
 1980 PROCbig_cur
 1990 INPUTTAB(3,17)""$T%
 2000 PROCno_cur
 2010 REM Search first the surnames...
 2020 FOR I%=1TONR%
 2030 IF INSTR($FNh(I%),$T%)=0THEN2040 E
LSE IF NOTF%THENPROCshow(I%):PROCbotf("S
earch"):F%=TRUE ELSEPROCanytab:IFGET=9TH
EN2220 ELSEPROCshow(I%):PROCbotf("Search
")
 2040 NEXT
 2050 REM Then search the rest...
 2060 REM $T% has to be reversed, due to
 internal format of data on disc...
 2070 $U%=""
 2080 FOR I%=1TOLEN$T%
 2090 $U%=$U%+MID$($T%,LEN$T%-I%+1,1)
 2100 NEXT
 2110 $T%=$U%:PTR#X=0
 2120 FOR I%=1TONR%
 2130 REPEAT:A%=BGET#X:UNTILA%=60
 2140 start%=PTR#X-1:$U%=""
 2150 REPEAT
 2160 A%=BGET#X:$U%=$U%+CHR$A%
 2170 UNTIL A%=62ORLEN$U%=255
 2180 IF INSTR($U%,$T%)=0THEN2190 ELSE I
F NOTF%THENPROCshow(FNrecno(start%)):PRO
Cbotf("Search"):F%=TRUE ELSE PROCanytab:
IFGET=9THEN2220 ELSE PROCshow(FNrecno(st
art%)):PROCbotf("Search")
 2190 IF A%=62THENNEXTELSEGOTO2150
 2200 REM Then print messages...
 2210 IF NOTF%THENPROCnomatch:IFINKEY350
 2220 IF NOTquiet%THENPRINTTAB(3,20)SPC3
"Search for another string?"SPC4TAB(3,21
)SPC17:IF FNget("YN")="Y"THEN I%=0:F%=0:
GOTO1960
 2230 ENDPROC
 2240 :
 2250 DEF PROCsort
 2260 LOCAL I%,J%,K%,temp%,A$
 2270 temp%=&70
 2280 PROChead("SORT RECORDS")
 2290 PRINT''"Which field do you want to
 sort under?"
 2300 FOR I%=0TO11
 2310 PRINT~(I%+1);". ";field$(I%)
 2320 NEXT
 2330 PRINT'"Selection - ";
 2340 PROCbig_cur
 2350 A$=FNget("123456789ABCH")
 2360 PROCno_cur
 2370 PRINTA$
 2380 IF A$="1"THENPROCsorthash:PROCsave
hash:ENDPROC
 2390 A%=EVAL("&"+A$)-1
 2400 PTR#X=0
 2410 FOR I%=1TONR%
 2420 FOR J%=1TONR%-I%
 2430 PTR#X=1+!(FNh(J%)+14)AND&FFFF
 2440 FORK%=1TOA%
 2450 INPUT#X,$T%
 2460 NEXT
 2470 PTR#X=1+!(FNh(J%+1)+14)AND&FFFF
 2480 FORK%=1TOA%
 2490 INPUT#X,$U%
 2500 NEXT
 2510 IF $T%>$U%THEN OSCLI("FX200 1"):FO
R K%=0TO12STEP4:!temp%=!(FNh(J%)+K%):!(F
Nh(J%)+K%)=!(FNh(J%+1)+K%):!(FNh(J%+1)+K
%)=!temp%:NEXT:*FX200 0
 2520 NEXT:NEXT
 2530 PROCsavehash
 2540 ENDPROC
 2550 :
 2560 DEF PROCstatus
 2570 LOCAL I%,G%,J%,@%
 2580 PROChead("CURRENT STATUS")
 2590 PRINT'''"Press arrow keys to toggle
 on/off, or RETURN to return to main men
u."
 2600 PRINT''''"Duplicate warning"
 2610 @%=6
 2620 PRINT'''CHR$134,NR%;"  records on
```

```
file"
 2630 PRINTCHR$134,FNfilesize;"K record
data on disc"
 2640 PRINTCHR$134,INT(FNdiskfree/1024);
"K disc space left"
 2650 PRINTCHR$134,FNmemfree;"K free mem
ory"
 2660 REPEAT
 2670 PRINTTAB(28,8)CHR$157CHR$132;:IFD%
THENPRINT"ON ";ELSEPRINT"OFF";
 2680 PRINTSPC2CHR$156CHR$132
 2690 G%=GET
 2700 IF G%=136THEND%=FALSE
 2710 IF G%=137THEND%=TRUE
 2720 UNTILG%=13
 2730 ENDPROC
 2740 :
 3070 DEF PROCselectf(B$)
 3080 LOCAL I%
 3090 PROCbotf("Select records to "+B$)
 3100 PRINTTAB(3,15)"Press ";LEFT$(B$,1)
;" to ";B$;" this record"
 3110 PRINTTAB(9,16)"D to de-select"
 3120 PRINTTAB(9,17)"[ for previous reco
rd"
 3130 PRINTTAB(9,18)"] for next record"
 3140 PRINTTAB(9,19)"F to finish selecti
on."
 3150 ENDPROC
 3160 :
 3490 DEF PROCexists
 3500 VDU7
 3510 PRINTTAB(5,11)"WARNING! A record a
lready exists"'TAB(10)"with this ";field
$(0);"."
 3520 IF INKEY500
 3530 PRINTTAB(0,11)SPC79
 3540 ENDPROC
 3550 :
 4100 *LOAD MCCMPCT
 4510 DEF PROCfind(quiet%)
 4520 LOCALS%,F%,R%
 4530 IF NOTquiet%THENREPEAT
 4540 F%=0
 4550 PROCbotf("Find")
 4560 S%=-1
 4570 REPEAT
 4580 S%=S%+1
 4590 PRINTTAB(3,15)"Find record with wh
ich ";:IFLEN(field$(S%))<12THENPRINTfiel
d$(S%);"?";SPC(38-POS):PRINTTAB(3,16)SPC
14ELSEPRINTSPC12:PRINTTAB(3,16)field$(S%
);"?"SPC4
 4600 PROCbig_cur
 4610 INPUTTAB(3,18)""$T%
 4620 UNTIL$T%>""ORS%=11
 4630 PROCno_cur
 4640 IF S%>0THEN PROCfindother($T%,S%):
GOTO4700
 4650 R%=FNfindsurname($T%)
 4660 IF LEFT$($(FNh(R%)),LEN($T%))<>$T%
THEN4680ELSEIFNOTF%THENPROCshow(R%):PROC
botf("Find"):F%=TRUE ELSEPROCanytab:IFGE
T=9THENGOTO4700 ELSEPROCshow(R%):PROCbot
f("Find")
 4670 R%=R%+1:IFR%<=NR%THENGOTO4660
 4680 IF NOTF%THENR%=0:PROCnomatch:IFINK
EY350
 4690 R%=R%-1
 4700 IF NOTquiet%THENPRINTTAB(7,20)"Fin
d another record? (Y/N)";:UNTILFNyn="N"
 4710 ENDPROC
 4720 :
 4730 DEF FNfindsurname($T%)
 4740 LOCALJ%,R%,I%
 4750 REM To find a surname is quick...
 4760 $T%=LEFT$($T%,13)
 4770 J%=INT((NR%+1)/2):R%=INT((NR%+1)/2
)
 4780 FOR I%=1TOINT(SQR(NR%))+1
 4790 J%=INT((J%-1)/2)+1
 4800 IF LEFT$($(FNh(R%)),LEN($T%))<$T%T
HEN R%=R%+J% ELSE IF LEFT$($(FNh(R%)),LE
N($T%))>$T%THENR%=R%-J%
 4810 NEXT
 4820 REPEAT
 4830 R%=R%-1
 4840 UNTIL LEFT$($(FNh(R%)),LEN($T%))<$
T%
 4850 R%=R%+1
 4860 IF LEFT$($(FNh(R%)),LEN($T%))<>$T%
THENR%=0
 4870 =R%
 4880 :
 4890 DEF PROCfindother($T%,S%)
 4900 LOCALI%,J%,F%
 4910 REM (Variables: S%=field no.;F%=fo
und flag;$T%=search string)
 4920 FOR I%=1TONR%
 4930 PTR#X=(!(FNh(I%)+14)AND&FFFF)+1
 4940 FOR J%=1TOS%
 4950 INPUT#X,$U%
 4960 NEXT
 4970 IF LEFT$($U%,LEN($T%))<>$T%THEN498
0 ELSE IF NOTF%THEN PROCshow(I%):PROCbot
f("Find"):F%=TRUE ELSE PROCanytab:IFGET=
9THEN ENDPROC ELSE PROCshow(I%):PROCbotf
("Find")
```

```
 4980 NEXT
 4990 IF NOTF%THEN PROCnomatch:IFINKEY35
0
 5000 ENDPROC
 5010 :
 5020 DEF FNmemfree:=INT((HIMEM-(!2 AND
&FFFF))/1024)
 5030 :
 5080 DEF PROCselectorder(B$)
 5090 LOCAL I%,A%,A$,C$
 5100 PROChead("SELECT FIELDS")
 5110 FOR I%=0TO11
 5120 PRINTTAB(pos%(I%,0)-4,pos%(I%,1)+2
)CHR$135;~I%+1"."CHR$134field$(I%);
 5130 NEXT
 5140 PRINTTAB(0,14)CHR$130"Enter whiche
ver field numbers / letters"CHR$130"you
want, in the order you want them,"'CHR$1
30"then press RETURN."
 5150 INPUT""C$
 5160 FOR I%=0TOLENC$-1
 5170 fo%(I%)=EVAL("&"+MID$(C$,I%+1,1))-
1
 5180 NEXT
 5190 FOR I%=LENC$TO11
 5200 fo%(I%)=TRUE
 5210 NEXT
 5220 PROChead(B$+" Records")
 5230 PRINT''
 5240 ENDPROC
 5250 :
 5260 DEF PROCselectrecords(B$)
 5270 LOCAL I%,A$
 5280 PROCselectf(B$)
 5290 FOR I%=&5A00TO&5BFC:!I%=0:NEXT
 5300 I%=1
 5310 REPEAT
 5320 PROCshow(I%)
 5330 IF I%?&5A00=255 THENPRINTTAB(23,9)
CHR$129B$" selected";
 5340 A$=FNget("PDSF"+CHR$136+CHR$137)
 5350 IF A$=LEFT$(B$,1)THEN I%?&5A00=TRU
E:PRINTTAB(23,9)CHR$129B$" selected";:I%
=I%+1
 5360 IF A$="D"THENI%?&5A00=FALSE:PRINTT
AB(23,9)SPC15;
 5370 IF A$=CHR$136THENI%=I%-1:IFI%<1THE
NIA%=1
 5380 IF A$=CHR$137THENI%=I%+1
 5390 UNTIL I%>NR%ORA$="F"
 5400 PROChead(B$+" Records")
 5410 PRINT''
 5420 ENDPROC
 5430 :
```

Listing 2

```
   10 REM Program Data Compacter
   20 REM Version B1.0
   30 REM Author  Mike Bryant
   40 REM BEEBUG Jan/Feb 1991
   50 REM Program subject to copyright
   60 :
  100 read%=&70
  110 write%=&74
  120 compare=&78
  130 jump=&7A
  140 returnH=&7C
  150 buffer_length=&7D
  160 last_char=&7E
  170 buffer_start=&7F
  180 loop_count=&80
  190 channel=&82
  200 t=&5B00
  210 :
  220 osargs=&FFDA:osbget=&FFD7
  230 osbput=&FFD4:addr%=&900
  240 FOR pass%=0 TO 3 STEP 3:P%=addr%
  250 [OPTpass%
  260 :
  270 STX loop_count:STY loop_count+1
  280 STA channel:TAY
  290 .next_record
  300 LDX #read%:LDA #1:JSR osargs
  310 .find_a_less_than_sign
  320 JSR osbget:CMP #60
  330 BNE find_a_less_than_sign
  340 LDA #0:JSR osargs
  350 LDA read%:SEC:SBC #1:STA compare
  360 LDA read%+1:SBC #0:STA compare+1
  370 BCS dont_DEC_compareH
  380 DEC compare+1
  390 .dont_DEC_compareH
  400 CMP write%:BNE end:LDA compare+1
  410 CMP write%+1:BNE end
  420 .find_a_greater_than_sign
  430 JSR osbget:CMP #62
  450 BNE find_a_greater_than_sign
  460 LDA #0:JSR osargs
  470 LDX #write%:JSR osargs
  480 JMP end_of_main_loop
  490 .end
  500 LDA #&5C:STA jump+1
  510 LDA #14:STA jump
  520 LDX #0:STX returnH
  530 .next_pointer
  540 INX:TXA:BNE not_big
  550 LDA returnH:CLC:ADC #1
  560 CMP #2:BNE not_too_big_yet
```

Continued on page 55

Practical Assembler (Part 8)

by Bernard Hill

Last month we looked at the assembler equivalent of the *FX command, the OSBYTE call. In common with a large number of assembler routines (including those you write yourself) the parameters which it requires are loaded into the A, X and Y registers before the JSR into the routine. All well and good, but when more than three one-byte parameters are needed, then 6502 assembler subprograms have to find another means of passing parameters.

The most popular of these is to pass them in a parameter block. Sometimes these are at a fixed address (rather like the CALL-with-parameters which we explored in BEEBUG Vol.9 Nos.4-6), but in order to maintain flexibility the system routines use a variable address. The location of this parameter block is passed to the routine by using the X and Y registers to hold the low and high bytes of its address. That leaves the A register free to hold a 'function number', and this is how the system OSWORD routine works.

OSWORD is used for a few functions (by no means as many as OSBYTE) such as reading the Master's real-time clock or reading a character definition (a sort of inverse of VDU23). The simplest example of an OSWORD command is perhaps the assembler equivalent of the SOUND command, OSWORD function number 7.

Let's suppose we want to issue the instruction:

```
SOUND 1,-15,100,20
```

from assembler. This is the fragment of code which would do the trick:

```
LDX #pblock MOD 256 \ pass address of
LDY #pblock DIV 256 \ parameter block
LDA #7              \ SOUND is function 7.
JSR &FFF1           \ call OSWORD
....
.pblock EQUW 1
        EQUW -15
        EQUW 100
        EQUW 20
```

It's that easy. The parameter block in this case is eight bytes (or four "words") long, one for each of the four parameters. Incidentally, do remember the '#' characters in the first two lines: omission of these causes a large fraction of beginners' bugs - I know from experience! If you're interested in following up any of the other OSWORD calls then you can look in any of the Advanced User Guides or even the standard User Guide for the Model B. This will tell you clearly the function number and the parameter block structure.

Basic	Assembler Name	Function No
(A, Y are registers)		
A=OPENIN		A=&40
A=OPENOUT	OSFIND	A=&80
A=OPENUP	(&FFCE)	A=&C0
CLOSE#Y		A=0
*SAVE		A=0
*LOAD	OSFILE	A=&FF
*DELETE	(&FFDD)	A=6
A=BGET#Y	OSBGET(&FFD7)	
BPUT#Y,A	OSBPUT (&FFD4)	
x=PTR#Y	10	A=0
PTR#Y=x	OSARGS	A=1
x=EXT#Y	(&FFDA)	A=2

Table 1. Assembler equivalents of Basic filing system calls

FILING SYSTEM CALLS

Many people have a lot of trepidation about handling filing system calls from assembler, but the process is remarkably easy. The key to the problem is to have a clear view of the filing system routines in Basic, as most (not INPUT# and PRINT#) have equivalents in assembler. These equivalents are to be found in the Advanced User Guide where the full details are listed, and Table 1 contains some of the commoner calls. Of the other calls not listed in this table, some have Basic equivalents (such as EOF#) and some do not (such as writing a load address onto an existing file), but once the principle of the parameter block is understood then a perusal of one of the Advanced User Guides shows the correct call, function number and parameter block structure.

*FCOPY

In order to illustrate the simplicity of these filing system calls I am going to introduce a program which copies a file. Those readers who have used other filing systems (such as MS-DOS) will have been frustrated by the Beeb's inability to make another identical copy of a file but under another name. The *COPY command only transfers the file to another disc, and I have had to resort to:

```
*COPY 0 2 oldname
*RENAME :2.oldname newname
*COPY 2 0 newname
*DELETE :2.newname
```

in order to duplicate a data file on drive 0.

Before getting into assembler, let's look at the Basic program which would do the job for us:

```
 10 DIM old 20, new 20
 20 INPUT "Old name:" $old
 30 INPUT "New name:" $new
 40 handle1=OPENIN($old)
 50 IF handle1=0 THEN STOP
 60 handle2=OPENOUT($new)
 70 REPEAT
 80    A=BGET#handle1
 90    BPUT#handle2,A
100 UNTIL EOF#handle1
110 CLOSE#handle1:CLOSE#handle2
```

I'm using static strings - *$old* - rather than dynamic - *old$* - for reasons which will be apparent later. The 20 bytes allocated for the name may need to be increased for full path names if the ADFS is to be used).

Note that if the input file does not exist then *handle1* will be set to 0 at line 40, and line 50 will stop the program. As it stands, this program will duplicate the contents of the file, but will not copy the execution and load address of the new file to be the same as the old. For this we need to use an OSFILE call with values of A which are not contained in Table 1 as they have no Basic equivalent. These are A=5 to read the file load/execution addresses, and A=1 to write the same to an existing file. The actual parameter block is 18 bytes long (numbered 0 to 17) and is listed in Table 2, but in fact we shan't be using the parameter block beyond byte 9 for these two functions.

OSFILE has an 18-byte parameter block as follows:

Bytes 0, 1	: address of the characters of the filename, ending in &0D
Bytes 2 -5	: load address of the file
Bytes 6 -9	: execution address of the file
Bytes 10-13	: length of file or start address
Bytes 14-17	: end address of file if saving.

OSARGS has a 4-byte zero-page parameter block (address in X) to contain the 4-byte integer variable x. Y contains the handle as usual.

Table 2. OSFILE and OSARGS parameter

Luckily we can use the same parameter block for both reading the old exec/load addresses and writing the new. But let's first allocate the parameter block, set up the filename pointer (this is why we used fixed strings - *$old* - and they always end in &0D as required) and read the addresses from the source file:

```
110 DIM pblock 17
120 ?pblock=old MOD 256
130 pblock?1=old DIV 256
140 X%=pblock MOD 256
150 Y%=pblock DIV 256
160 A%=5
170 CALL &FFDD
```

We don't even need to look at these load and execution addresses, we can just copy them to the target file by replacing the filename pointer and changing A to 1. The X and Y registers were not changed by the call:

```
180 ?pblock=new  MOD 256
190 pblock?1=new DIV 256
200 A%=1
210 CALL &FFDD
```

And that's it.

ASSEMBLER IMPLEMENTATION

This is printed at the end of this article as Listing 1. If we agree that the syntax of the command to be executed by the assembler program is:

```
*FCOPY oldfile newfile
```

then we can obtain the source and target file names as we did in last month's article when we saw that (&F2),Y points to the command string tail (though this is only valid for the DFS, not the ADFS). Lines 130-310 of Listing 1 are concerned with copying both filenames to their places and putting the terminating &0D on the end of each. An error (see BEEBUG Vol.9 No.2) is of course generated if either filename is missing from the command.

Now to open the input and output files (OPENIN/OUT) we use the OSFILE function as in Table 1. The only 'parameter block' the OSFILE uses is the filename, and X and Y must be set up to point to this, and the correct value of A set up for OPENIN or OPENOUT (&40 and &80 respectively). After the call to open the file the A register contains the file handle, and as in Basic, if it is zero the file could not be opened. Lines 330-450 cover the opening of these two files.

Then it is simply a matter of copying the bytes of the file one by one until the end of file 1 is reached. For reading, the OSBGET function is used, and here the Y register contains the handle. On return from OSBGET, the A register contains the byte, and if the carry flag is set then the end of file has been reached and the byte returned is invalid. OSBPUT performs the BPUT# function, A is the byte to write, Y contains the handle, and so lines 460-490 perform this simple process. With a final call to OSFIND with A=0 (and Y having the file handle) to close each file (lines 510-520), we can allocate the load and execution addresses as in the Basic program above (lines 540-630).

CONCLUSION

Although this byte-by-byte copying doesn't disturb any programs in RAM, it is very slow, but it does illustrates many assembler filing system functions very well. But a faster method would be to *LOAD and *SAVE the file if it will fit in the available RAM. Next month we will produce a program to perform this and at the same time show how this sort of routine can be run from sideways RAM.

Listing 1

```
   10 REM Program FCOPY
   20 REM Version B1.0
   30 REM Author Bernard Hill
   40 REM BEEBUG Jan/Feb 1991
   50 REM Program subject to copyright
   60 :
  100 FOR opt=0 TO 3 STEP 3
  110 P%=&900:Q%=P%
  120 [OPT opt
  130 DEY \ first find file names
  140 .skipsp INY:LDA (&F2),Y
  150 CMP #32:BEQ skipsp:LDX #0
  160 .file1 STA filein,X
  170 INX:INY:LDA (&F2),Y
  180 CMP #13:BEQ nofile2
  190 CMP #32:BEQ endfile1:BNE file1
  200 .nofile2 BRK:EQUB 67
  210 EQUS "Filename missing":EQUB 0
  220 \ place &0D on end of name
  230 .endfile1 LDA #13:STA filein,X
  240 \ second file
  250 .loop INY:LDA (&F2),Y
  260 CMP #32:BEQ loop:LDX #0
  270 .file2 STA fileout,X
  280 INX:INY:LDA (&F2),Y
  290 CMP #13:BEQ endfile2
  300 CMP #32:BNE file2
  310 .endfile2 LDA #13:STA fileout,X
  320 \ now open both files
  330 LDX #filein MOD 256
  340 LDY #filein DIV 256
  350 LDA #&40:JSR &FFCE \ osfind
  360 CMP #0: BNE okfile1
  370 BRK:EQUS "File 1 not found":EQUB 0
  380 .okfile1 STA handle1 \save handle
  390 LDX #fileout MOD 256
  400 LDY #fileout DIV 256
  410 LDA #&80:JSR &FFCE \ osfind
  420 CMP #0: BNE okfile2
  430 BRK:EQUS "Cannot open file 2"
  440 EQUB 0
  450 .okfile2 STA handle2
  460 .rp LDY handle1:JSR &FFD7 \ osbget
  470 BCS eof \end of file,byte no good
  480 LDY handle2:JSR &FFD4 \ osbput
  490 JMP rp
  500 .eof \close all files
  510 LDA #0:LDY handle1:JSR &FFCE
  520 LDY handle2:JSR &FFCE
  530 \ now set the exec/load addrs
  540 LDA #filein MOD 256:STA pblock
  550 LDA #filein DIV 256:STA pblock+1
  560 LDX #pblock MOD 256
  570 LDY #pblock DIV 256
  580 LDA #5 \ read cat info
  590 JSR &FFDD \ osfile
  600 LDA #fileout MOD 256:STA pblock
  610 LDA #fileout DIV 256:STA pblock+1
  620 LDA #1 \ write cat info
  630 JSR &FFDD \ osfile
  640 RTS \ finished
  650 \ data area....
  660 .handle1 EQUB 0
  670 .handle2 EQUB 0
  680 .pblock  EQUS STRING$(18,CHR$0)
  690 .filein  EQUS STRING$(20,CHR$0)
  700 .fileout EQUS STRING$(20,CHR$0)
  710 ]:NEXT
  720 c$="SAVE FCOPY "+STR$~Q%+" "+STR$~
P%
  730 PRINTc$:OSCLIc$
```

Word Processor Input (Part 3)

by Andrew Rowland

In this article, the last of the present series, we will put the character string editor introduced in Part 1 to serious uses - a function key editor for the Master, and a Basic line editor. Neither of these is a new idea, but the brevity of the code required and their uniformity with the input routine make them very attractive.

Before I start, I will mention that the utilities presented here use all or part of the area &900 to &CFF. Master users on Econet should be aware of this, as must model B owners, who will lose their user definable characters when using Edlin.

MASTER FUNCTION KEYS

The Master has 1K of memory set aside for the function keys, compared with 256 bytes in the Model B. While there have been function key editors for the Model B (e.g. Vol.3 No.1), none has been published for the Master, despite the greater scope that this machine gives you.

In listings 1 and 2 we have two utilities, FSHOW and FEDIT, with WPinput providing the editor for the latter.

A NEW *SHOW

The first utility prints the contents of a function key, rather as *SHOW does, or the recent *FKDEFS in Bill Hine's EdiKit ROM (Vol.8 No.10), but differs in the way it displays the key definition.

Firstly, both these commands use a format where the whole string is enclosed in quotes, quotes within the string are preceded by a vertical bar and codes above 127 are shown as a combination of vertical bars, exclamation marks and printable characters. Personally, I prefer to see it displayed as I would type it in: only surrounded by quotes if necessary and only control codes (less than 32) preceded by a vertical bar. Codes above 128 are shown normally using the Master's extended font or as teletext control codes in mode 7. If this isn't your cup of tea, I refer you to the above alternatives, but I find it less cluttered and more readable. In any case, the routine is a useful half-way house to FEDIT.

Secondly, you may either specify a particular key number or type *FSHOW without a parameter and see all 16 keys displayed.

Type in and run Listing 1, adhering to the line numbers. The assembled version, FSHOW, will be saved to disc, and can be called by typing *FSHOW <number> or just *FSHOW for all the keys. Unlike *SHOW, the key is displayed complete with *KEY so that the current function key definitions can be spooled to disc. To save them type:

```
*SPOOL Keys
*FSHOW
*SPOOL
```

To redefine the function keys, type:

```
*EXEC Keys.
```

THE FUNCTION KEY EDITOR

Now we can display the key definitions, the next stage is to edit them. Firstly, we must store the characters as displayed (control codes, you will recall, have been expanded by preceding a printable character with a vertical bar). This is achieved by the subroutine *writechar* (line 2660) which has a different function in FEDIT to FSHOW. Secondly, we will use the technique described in last month's issue to edit the buffer. And thirdly, we will issue a *KEY command to redefine the key in question.

Listing 2 shows all the changes you need to make to Listing 1 in order to create FEDIT. Enter the instructions exactly as printed, taking care with line numbers. If you have the monthly disc, you will find the program complete on disc. If you are using InputROM remember to modify line 2860 as shown. When run, FEDIT is assembled and saved to disc; it is called with *FEDIT <key number>. Full details of how to use the string editor are in Part 1 of the series. You will find you can alter the key number if you want to copy one key definition (possibly modified) to another function key without losing the original.

The new key definition is programmed when you press Return, or aborted if you press Escape.

BASIC LINE EDITOR

With WPinput you are able to enter lines of Basic with great ease and comfort. What has been lacking is the ability to edit lines later without copying them entire from the screen. Basic line editors have been published before, the most recent being Paul Pibworth's (Vol.8 No.2). This was an excellent utility, but the approach adopted here results in a very short program, always an important consideration with the memory constraints of the Beeb, and offers powerful editing facilities which are utterly consistent with the environment used to enter the line in the first place.

Enter and run Listing 3. Edlin will be assembled and saved to disc, and should be used by entering *EDLIN <line number> from Basic's command mode (not from within a program). If you give a non-existent line number, by the way, you will find yourself editing a blank line - just press Escape. You may rename Edlin to something shorter if you wish, or program Key 0 "*EDLIN " as an aid. It is useful as a debugging tool too. Make the last line in your error handler:

```
  OSCLI "EDLIN "+STR$(ERL):END
```

Remove it when the program is complete.

HOW EDLIN WORKS

The great economy of code is achieved by an interesting technique which involves no calls into the Basic ROM, so it can be used on any machine. Consider what happens when you want to change a line of Basic in the normal way. First, the > prompt appears and you type LIST 10 or some such. The line concerned is listed on your screen. Then the > prompt reappears and you enter a new line 10 using screen copying and typing. You press Return and it becomes part of the program.

This is stating the obvious, but Edlin uses this sequence of events in a special way. The 'main' part of Edlin is from line 1040 to 1290. Let us simplify and say it does nothing and exits. We now start the sequence described above - '>' appears and Basic calls Osword 0 for its next line of input. When Osword 0 exits, Basic will expect to find a string in its input buffer (which is memory page 7). And it will! Edlin has actually copied "L." and the line number you specified into page 7, and altered the Osword vector to a dummy routine, newWORD1 (line 1350). Basic now dutifully lists the line. But nothing appears on the screen - Edlin has also altered the OSWRCH vector to redirect screen output to a buffer. This is handled by *vduentry* (line 1550), which stores each character in page &C instead of displaying it.

When it realises the end of a line has been reached (Basic sends a line feed (ASCII 10) at the end of each line), it calls WPinput to let you edit it - this is the first time something visible happens. The final stage is when the > prompt appears again. You would normally enter your modified line here, but again, Edlin has diverted the Osword vector, this time to *newWORD2* (line 1430), which copies the string from page &C to page 7 and exits. Basic accepts this and inserts the line in its program. By now, all vectors have been restored to their original state and we are back to normal.

If you got lost in all that, don't worry. Edlin is far simpler to use than to describe. One hint: if you want to repeat a line in a program, say if line 240 is the same as line 20, type *EDLIN 20 and change the line number to 240. A copy of the line is produced, leaving line 20 unchanged.

ROM BONUS

The three utilities presented this month are all disc based, which slows down their operation. As a bonus, on this month's disc, you will find a ROM image called InputRom, which includes WPinput, an enhanced *HELP and the three new star commands, all of which work on the Model B as well as the Master. A number of additional editing functions are included, such as *delete to end of line*, and *delete word*, and there is a feature which lets you recall a specific string by typing a few of the letters it contains, then pressing Ctrl-H. Together, these features make a comfortable and powerful environment for your work.

I gratefully acknowledge the use of David Spencer's routine to print decimal numbers (BEEBUG Vol.7 No.4).

Listing 1

```
   10 REM Program .>FSHOWbas
   20 REM Version B1.00 (M)
   30 REM Author  Andrew Rowland
   40 REM BEEBUG  Jan/Feb 1991
   50 REM Program subject to copyright
   60 :
  100 mc%=&900:zp=&70:fn=&F2
  110 oswrch=&FFEE:osnewl=&FFE7
  120 oscli=&FFF7:osword=&FFF1:osbyte=&F
FF4
  130 buffer=&C00:transfer=&134B
  140 PROCass
  150 a$="SAVE FSHOW "+STR$~mc%+" "+STR$
~P%
  160 PRINT a$:OSCLI a$
  170 END
  180 :
 1000 DEF PROCass
 1010 FOR pass=0 TO 3 STEP 3
 1020 P%=mc%
 1030 [OPT pass
 1040 LDA &F4:ORA #&80
 1050 \ access private RAM
 1060 STA &F4:STA &FE30
 1070 :
 1080 LDY #0 \ decode parameter
 1090 .param INY
 1100 LDA (fn),Y:CMP #13:BEQ noparam
 1110 CMP #32:BNE param
 1120 .space
 1130 INY:LDA (fn),Y
 1140 CMP #32:BEQ space
 1150 CMP #13:BEQ noparam
 1160 JSR showkey:BRA end
 1170 :
 1180 .noparam \ show all Fkeys
 1190 LDX #0
 1200 .lp JSR show
 1210 CPX #16:BNE lp
 1220 :
 1230 .end \ page out private ram
 1240 LDA &F4:AND #&7F
 1250 STA &F4:STA &FE30
 1260 RTS
 1270 :
 1280 .showkey
 1290 \ return rest command line in zp
 1300 TYA:CLC:ADC fn:STA zp
 1310 LDA #0:ADC fn+1:STA zp+1
 1320 .numeric
 1330 LDY #0 \ push decimal param
 1340 .readparamlp
 1350 LDA (zp),Y
 1360 CMP #13:BEQ readall
 1370 CMP #32:BEQ readall
 1380 INY:PHA:BRA readparamlp
 1390 \ Y = length of string
 1400 .readall  \ retrieve backwards
 1410 LDX #0:STX accum
 1420 .numlp
 1430 PLA
 1440 CMP #ASC"0":BCC paramerror
 1450 CMP #ASC"9"+1:BCS paramerror
 1460 SEC:SBC #ASC"0"
 1470 CPX #0:BEQ numover
 1480 STX count
 1490 .nlp1 STA byte
 1500 LDA #10:STA multiplicand
 1510 STZ multiplicand+1
 1520 JSR mult
 1530 LDA zp+1:BNE paramerror
 1540 LDA zp
 1550 DEC count:BNE nlp1
 1560 .numover
 1570 CLC:ADC accum:BCS paramerror
 1580 STA accum
 1590 INX:DEY:BNE numlp
 1600 LDX accum:CPX #16:BCS paramerror
 1610 JSR show:RTS
 1620 :
 1630 .paramerror
 1640 BRK:EQUB 251
 1650 EQUS "Bad key":BRK
 1660 :
 1670 .error BRK:EQUB 251
 1680 EQUS "String too long":BRK
 1690 :
 1700 .show STX key:LDY #0
 1710 .lp1
 1720 LDA string,Y:BEQ over
 1730 JSR writechar
 1740 INY:BRA lp1
 1750 .over
 1760 LDY #0:JSR prdec
 1770 LDA #32:JSR writechar
 1780 LDX key
 1790 LDA &8000,X:STA zp
 1800 LDA &8011,X:STA zp+1
 1810 INX
 1820 SEC:LDA &8000,X
 1830 SBC zp:STA length
 1840 LDA &8011,X:SBC zp+1
 1850 BNE error
1860 LDA length:BEQ nul
1870 LDY #0:STY flag
1880 LDA (zp),Y:CMP #ASC" "
1890 BNE loop:INC flag
1900 JSR quote
1910 .loop
1920 LDA (zp),Y:JSR prch
1930 INY:CPY length
1940 BNE loop
1950 LDA flag:BNE exit
1960 JMP osnewl \ RTS
1970 :
1980 .prch
1990 CMP #ASC" ":BCS notctrl
2000 ORA #&40:BCC ctrl
2010 .notctrl
2020 CMP #ASC"""":BNE out
2030 LDX flag:BEQ out
2040 .ctrl
2050 PHA:LDA #ASC"|":JSR writechar
2060 PLA
2070 .out JMP writechar
2080 :
2090 .nul JSR quote
2100 .exit
2110 JSR quote:JMP osnewl
2120 .quote
2130 LDA #ASC"""":JMP writechar
2140 :
2150 .prdec
2160 STX num:STY num+1
2170 .nprt
2180 CLC:PHP:LDY #4
2190 .nprt1 LDX #&30
2200 .nprt2 SEC:LDA num
2210 SBC tenlo,Y:PHA:LDA num+1
2220 SBC tenhi,Y:BCC nprt3
2230 STA num+1:PLA:STA num
2240 INX:BCS nprt2
2250 .nprt3 PLA:TXA
2260 CMP #&30:BEQ nprt4
2270 PLP:SEC:BCS nprt5
2280 .nprt4 PLP:BCS nprt5
2290 PHP:CPY #0:BNE nprt6
2300 PLP
2310 .nprt5 PHP:JSR writechar
2320 .nprt6 DEY:BPL nprt1
2330 PLP:RTS
2340 :
2350 .tenhi  EQUW 0:EQUB 0
2360 EQUB 3:EQUB &27
2370 .tenlo  EQUB 1:EQUB 10:EQUB 100
2380 EQUB &E8:EQUB &10
2390 :
2400 .mult
2410 STZ zp:STZ zp+1
2420 .mull
2430 LDA byte:BEQ mulend
2440 LSR A:BCC notadd
2450 LDA multiplicand:CLC
2460 ADC zp:STA zp
2470 LDA multiplicand+1:ADC zp+1
2480 STA zp+1
2490 .notadd
2500 ASL multiplicand
2510 ROL multiplicand+1
2520 LSR byte:BRA mull
2530 .mulend
2540 RTS
2550 :
2560 .multiplicand EQUW 0
2570 .byte   EQUB 0
2580 .count  EQUB 0
2590 .accum  EQUB 0
2600 .length EQUB 0
2610 .num    EQUW 0
2620 .key    EQUB 0
2630 .flag   EQUB 0
2640 .string EQUS "*KEY ":BRK
2650 :
2660 .writechar JMP oswrch
2670 ]NEXT
2680 ENDPROC
```

Listing 2

```
  10 REM Program .>FEDITbas
 150 a$="SAVE FEDIT "+STR$~mc%+" "+STR$~
P%
1080 LDY #0:STY index
1160 JSR showkey:JSR edit
1165 JSR star:BRA end
1180 .noparam
1190 BRK:EQUB 251
1200 EQUS "Syntax: FEDIT <key no.>"
1210 BRK
1730 JSR writechar:JSR oswrch
1960 RTS
2650 .index  EQUB 0
2660 .block  EQUD 0:EQUW 0
2670 :
2680 .writechar PHX
2690 LDX index:STA buffer,X
2700 INC index:PLX:RTS
2710 :
 2720 .edit
 2730 LDA #(buffer+5) MOD &100:STA block
 2740  LDA  #(buffer+5)  DIV  &100:STA
block+1
 2750 LDA #240:STA block+2
 2760 LDA #32 :STA block+3
 2770 LDA #255:STA block+4
 2780 LDA index:SEC:SBC #5
 2790 CMP #240:BCC edover
 2800 LDA #244
 2810 .edover STA block+5
 2820 LDX #block MOD &100
 2830 LDY #block DIV &100
 2840 \ for ROM vs, following line is
 2850 \ LDA #16:\JSR osword
 2860 LDA block+5:JSR transfer
 2870 BCS escape:RTS
 2880 :
 2890 .escape LDA #126:JSR osbyte
 2900 BRK:EQUB 17:EQUS "Escape":BRK
 2910 :
 2920 .star
 2930 LDX #buffer MOD &100
 2940 LDY #buffer DIV &100
 2950 JMP oscli
 2960 ]NEXT
 2970 ENDPROC
```

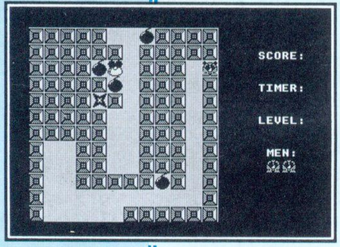
Listing 3

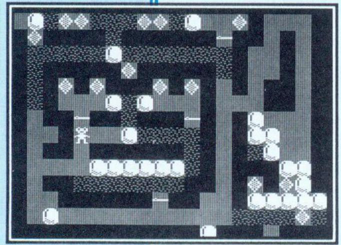
```
   10 REM Program .>EDLINbas
   20 REM Version 1.00
   30 REM Author  Andrew Rowland
   40 REM BEEBUG  Jan/Feb 1991
   50 REM Program subject to copyright
   60 :
  100 mc%=&900
  110 zp=&70:fn=&F2:block=&70
  120 oswrch=&FFEE:oscli=&FFF7
  130 osword=&FFF1:osbyte=&FFF4
  140 wrchv=&20E:wordv=&20C
  150 buffer=&700:store=&C00:transfer=&1
34B
  160 PROCass
  170 a$="SAVE EDLIN "+STR$~mc%+" "+STR$
~P%
  180 PRINT a$:OSCLI a$
  190 END
  200 :
 1000 DEF PROCass
 1010 FOR pass=0 TO 2 STEP 2
 1020 P%=mc%
 1030 [OPT pass
 1040 LDY #0:STY index
 1050 .param INY
 1060 LDA (fn),Y:CMP #13:BEQ noparam
 1070 CMP #32:BNE param
 1080 .space
 1090 INY:LDA (fn),Y
 1100 CMP #32:BEQ space
 1110 CMP #13:BEQ noparam
 1120 \ return rest command line in zp
 1130 TYA:CLC:ADC fn:STA zp
 1140 LDA #0:ADC fn+1:STA zp+1
 1150 \ set VDU vector
 1160 OPT FNcopy1(wrchv,oldwrchv)
 1170 OPT FNcopy2(vduentry,wrchv)
 1180 \ set WORDV
 1190 OPT FNcopy1(wordv,oldwordv)
 1200 OPT FNcopy2(newWORD1,wordv)
 1210 \ poke "L." into buffer
 1220 LDX #0:LDA #ASC"L":STA buffer,X
 1230 INX   :LDA #ASC".":STA buffer,X
 1240 \ poke command tail into buffer
 1250 LDY #0
 1260 .taillp
 1270 LDA (zp),Y:INX:STA buffer,X
 1280 INY:CMP #&0D:BNE taillp
 1290 STX length:RTS
 1300 :
 1310 .noparam BRK:BRK
 1320 EQUS "Syntax: EDLIN <line no.>"
 1330 BRK
 1340 :
 1350 .newWORD1
 1360 CMP #0:BEQ forme
 1370 JMP (oldwordv)
 1380 .forme
 1390 \ restore WORD vector
 1400 OPT FNcopy1(oldwordv,wordv)
 1410 LDY length:CLC:RTS
 1420 :
 1430 .newWORD2
 1440 CMP #0:BEQ forme2
 1450 JMP (oldwordv)
 1460 .forme2
 1470 \ restore WORD vector
 1480 OPT FNcopy1(oldwordv,wordv)
 1490 LDA #127:JSR oswrch \ delete '>'
 1500 LDY #0
 1510 .loop LDA (block),Y:STA buffer,Y
 1520 INY:CPY length:BEQ loop:BCC loop
 1530 LDY length:CLC:RTS
 1540 :
 1550 .vduentry
1560 CMP #10:BEQ end:STX temp
1570 LDX index:STA store,X
1580 INC index:BEQ end:LDX temp
1590 .vduout RTS
1600 :
1610 .end PHA:TXA:PHA:TYA:PHA
1620 \ restore VDU vector
1630 OPT FNcopy1(oldwrchv,wrchv)
1640 LDY #0
1650 .loop INY:LDA store,Y
1660 CMP #32:BEQ loop
1670 STY length:LDA index:CLC
1680 SBC length:STA block+5
1690 TYA:CLC:ADC #store MOD &100
1700 STA block
1710 LDA #store DIV &100:STA block+1
1720 LDA #238:STA block+2
1730 LDA #32 :STA block+3
1740 LDA #255:STA block+4
1750 LDX #block MOD &100
1760 LDY #block DIV &100
1770 \ for ROM vs, following line is
1780 \LDA #16:\JSR osword
1790 LDA block+5:JSR transfer
1800 BCS escape:STY length
1810 OPT FNcopy2(newWORD2,wordv)
1820 PLA:TAY:PLA:TAX:PLA:RTS
1830 :
1840 .escape LDA #126:JSR osbyte
1850 BRK:EQUB 17:EQUS "Escape":BRK
1860 :
1870 .temp     EQUB 0
1880 .index    EQUB 0
1890 .length   EQUB 0
1900 .oldwrchv EQUW 0
1910 .oldwordv EQUW 0
1920 ]NEXT
1930 ENDPROC
1940 :
1950 DEF FNcopy1(A%,B%)
1960 [OPT pass
1970 LDA A%:STA B%
1980 LDA A%+1:STA B%+1
1990 ]=pass
2000 :
2010 DEF FNcopy2(A%,B%)
2020 [OPT pass
2030 LDA #A% MOD &100:STA B%
2040 LDA #A% DIV &100:STA B%+1
2050 ]=pass
```

MikroTel (continued from page 46)

```
  570 BRK
  580 EQUB 200
  590 EQUS "Can't match pointer with Sur
name"
  600 BRK
  610 .not_too_big_yet
  620 STA returnH
  630 .not_big
  640 LDA jump:CLC:ADC #16:STA jump
  650 BCC no_carry:INC jump+1
  660 .no_carry
  670 LDY #0:LDA (jump),Y
  680 CMP compare:BNE next_pointer
  690 INY:LDA (jump),Y
  700 CMP compare+1:BNE next_pointer
  710 LDY #0:LDA write%:STA (jump),Y
  720 INY:LDA write%+1:STA (jump),Y
  730 LDY channel
  740 LDX #read%:LDA #0:JSR osargs
  750 LDA #60:STA t
  760 LDA #0:STA buffer_start
  770 .over_255_characters_long
  780 LDX #read%:LDA #1:JSR osargs
  790 LDX buffer_start
  800 .read_byte
  810 INX:JSR osbget:STA t,X
  820 CMP #62:BEQ finish_reading
  830 CPX #255:BNE read_byte
  840 .finish_reading
  850 STX buffer_length
  860 LDX #read%:LDA #0:JSR osargs
  870 LDX #write%:LDA #1:JSR osargs
  880 LDX #&FF
  890 .write_byte
  900 INX:LDA t,X:JSR osbput
  910 CPX buffer_length:BNE write_byte
  920 STA last_char
  930 LDX #write%:LDA #0:JSR osargs
  940 LDA #&FF:STA buffer_start
  950 LDA last_char
  960 CMP #62:BNE over_255_characters_lo
ng
  970 .end_of_main_loop
  980 LDA loop_count:SEC:SBC #1
  990 STA loop_count
 1000 LDA loop_count+1:SBC #0
 1010 STA loop_count+1:BNE next_record_j
ump
 1020 LDA loop_count:BNE next_record_jum
p
 1030 RTS
 1040 .next_record_jump
 1050 LDA #127:JSR &FFEE
 1060 JMP next_record
 1070 ]NEXT
 1080 *SAVE M.COMPACT 900 AFF
 1090 END
```

Arcade Games





George and the Dragon - Rescue 'Hideous Hilda' from the flames of the dragon, but beware the flying arrows and the moving holes on the floor.

Ebony Castle - You, the leader of a secret band, have been captured and thrown in the dungeons of the infamous Ebony Castle. Can you escape back to the countryside, fighting off the deadly spiders on the way and collecting the keys necessary to unlock the coloured doors?

Pitfall Pete - Collect all the diamonds on the screen, but try not to trap yourself when you dislodge the many boulders on your way.

Knight Quest - You are a Knight on a quest to find the lost crown, hidden deep in the ruins of a weird castle inhabited by dangerous monsters and protected by a greedy guardian.

Builder Bob - Bob is trapped on the bottom of a building that's being demolished. Can you help him build his way out?

Minefield - Find your way through this grid and try to defuse the mines before they explode, but beware the monsters which increasingly hinder your progress.

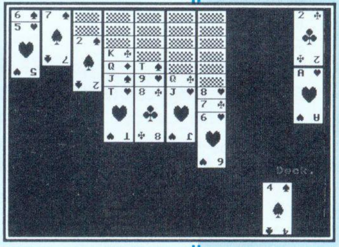
Manic Mechanic - Try to collect all the spanners and reach the broken-down generator, before the factory freezes up.

Quad - You will have hours of entertainment trying to get all these different shapes to fit.

Beebug Arcade Games Disc **£5.95** **+ 60p p&p**
Stock Codes **PAG1** (5.25" DFS 40/80T disc) **PAG2** (3.5" ADFS disc)

Board Games

Solitaire - an elegant implementation of this ancient and fascinating one player game, and a complete solution for those who are unable to find it for themselves.

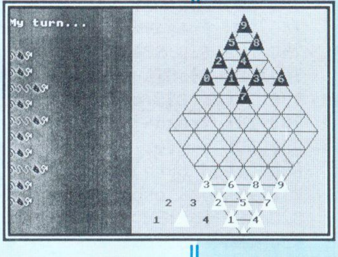


Roll of Honour - Score as many points as possible by throwing the five dice in this on-screen version of 'Yahtze'.

Patience - a very addictive version of one of the oldest and most popular games of Patience.

Elevenses - another popular version of Patience - lay down cards on the table in three by three grid and start turning them over until they add up to eleven.

Cribbage - an authentic implementation of this very traditional card game for two, where the object is to score points for various combinations and sequences of cards.



Twiddle - a close relative of Sam Lloyd's sliding block puzzle and Rubik's cube, where you have to move numbers round a grid to match a pattern.

Chinese Chequers - a traditional board game for two players, where the object is to move your counters, following a pattern, and occupy the opponent's field.

Aces High - another addictive game of Patience, where the object is to remove the cards from the table and finish with the aces at the head of each column.

Beebug Board Games Disc **£5.95** **+ 60p p&p**
Stock Codes **PBG1** (5.25" DFS 40/80T disc) **PBG2** (3.5" ADFS disc)

HINTS *and tips* HINTS *and tips* HINTS *and tips* HINTS *and tips* HINTS *and tips*

Hints and tips on almost any subject relating to the BBC micro and Master series are always welcome, and we pay £5 for each one published.

QUICK AND EASY FILE SAVE

J.M.Shepherd

Here is an easy method of saving a file with the correct name, assuming that you start your program with a REM statement consisting of the file name.

Enter and save the program QSAVE listed below. Turn on the computer and type:

```
  *LOAD QSAVE 900
  *KEY9 T%=TOP|MPAGE=&900|MRUN|MPAGE=&E
00|M
```

(changing the &E00 to &1900 (or whatever is appropriate), for a model B). For regular usage, this can be included in a !BOOT file. The use of function key 9 is not essential - any function key will do.

Saving the current program is carried out by pressing f9 only. This assigns T% to TOP, sets PAGE to &900 and runs QSAVE, before resetting PAGE to return to the current program.

In QSAVE, line 10 assigns a start address (N) and length (L) of the file to be saved. The occurrence of the first REM (token 244) is then found. Line 20 skips any possible spaces and line 30 builds up the file name. Line 40 saves the file as a Basic program and line 50 confirms the action taken.

```
   5 REM QSAVE
  10 N=&E00:L=T%-N:F$="":name=FALSE:REPE
AT:N=N+ 1:UNTIL ?N=244
  20 REPEAT:N=N+1:UNTIL ?N<>32
  30 REPEAT:IF?N=130 OR?N=32 name=TRUE E
LSE F$=F$+CHR$?N:N=N+1
  40 UNTIL name=TRUE:OSCLI"SAVE "+F$+"
E00+"+STR$L+" 802B"
  50 PRINT"Saved as ";CHR$34;F$;CHR$34:END
```

Again, note that 'E00' above should be changed to '1900' for the model B (see previous comment), and that '802B' should be changed as appropriate for your version of BBC Basic (use the *EX command to catalogue a disc of programs and use the four or eight digit hex value which appears in the penultimate column).

DATE STAMPING VIEWSTORE REPORTS

Ashley Allerton

There is no immediate facility in ViewStore for dating reports by taking the date from the internal clock on the Master. The date register exists in View ('|D' on any line with an 'RJ', 'LJ', 'CE', 'DH', or 'DF' command), but apparently not in ViewStore.

There is a way round this using the Comment facility, whereby a comment can be included in a report file, which can be different each time a report is printed. First include the following procedure near the beginning of your menu program, to fix 'today's date' in DATE$:

```
  20 th$="th"
  30 IF MID$(TIME$,6,1)="1" th$="st"
  40 IF MID$(TIME$,6,1)="2" th$="nd"
  50 IF MID$(TIME$,6,1)="3" th$="rd"
  60 IF MID$(TIME$,5,1)="0" PROCone ELSE
PROCtwo
  70 DEF PROCone:DATE$=MID$(TIME$,6,1)+
th$+MID$(TIM E$,7,10):ENDPROC
  80 DEF PROCtwo:DATE$=MID$(TIME$,5,2)+
th$+MID$(TIM E$,7,10):ENDPROC
```

Then, in your report file, include something like:

```
    (Date prepared:@@@@@@@@)
```

in the header, and ^C1 in the field list. Then, somewhere in the report (ViewStore allows you to put it anywhere, as it searches for comments before starting work on the report file) put in a comment line, with a brief reminder like 'DATE?'.

When work starts on the report file, a reminder will come up, and you can insert the date, or it can be done automatically by modifying line 995 of the menu program (see Hints & Tips, BEEBUG Vol.8 No.7) to:

```
  995 OSCLI("KEY9 LOAD ADDRESSES|MMODE131
|MUTILITY SELECT|MF|M|MSURNAME|MCHRISTIAN
 NAME|M|M|MUTILITY REPORT|MY|MP|MY|MPHONE
DI R|M|M|M|M "+DATE$+"|M*KEY9|M")
```

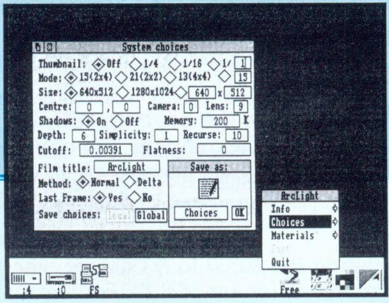
RISC USER

The Archimedes Magazine & Support Group

Now in its fourth year of publication, Risc User continues to enjoy the largest circulation of any magazine devoted solely to the Archimedes range of computers. It provides support for all Archimedes users at work (schools, colleges, universities, industry, government establishments) and home.

Existing Beebug members, interested in the new range of Acorn micros, may either transfer their membership to the new magazine or extend their subscription to include both magazines.

A joint subscription will enable you to keep completely up-to-date with all innovations and the latest information from Acorn and other suppliers on the complete range of BBC micros. RISC User has a massive amount to offer to enthusiasts and professionals at all levels.



```
Thumbnail: ◈Off ◇1/4 ◇1/16 ◇1/ 1
Mode: ◈15(2x4) ◇21(2x2) ◇13(4x4) ◇ 15
Size: ◈640x512 ◇1280x1024 ◇ 640 x 512
Centre: 0 , 0   Camera: 0   Lens: 9
Shadows: ◈On ◇Off   Memory: 200 K
Depth: 6   Simplicity: 1   Recurse: 10
Cutoff: 0.00391   Flatness: 0
Film title: ArcLight
Method: ◈Normal ◇Delta
Last Frame: ◈Yes ◇No
Save choices: Local Global
```

And here are some items covered in the most recent issues of RISC User:

FIRST LOOK AT BEZIER CURVES

New short series starting with using different mathematical methods for drawing curves.

PRINTER AND SERIAL BUFFER FOR THE ARC

A variable sized buffer which can be configured either as a printer buffer, or a serial buffer.

A MULTI-TASKING WIMP DEBUGGER

A multi-tasking application which allows you to debug Wimp based programs while testing and running them.

ARCLIGHT RAY TRACER

A review of the new ray-tracing package from Ace Computing.

IRLAM'S COLOUR SCANNER

A look into the first colour scanner for the Arc.

WP/DTP

A regular column which offers hints on using different DTP and WP packages.

PROGRAMMER'S WORKSHOP

A major new series covering all advanced aspects of the Archimedes and incorporating the Assembler Workshop. This month: understanding E-format discs.

MASTERING THE WIMP

A major and very popular series on the Wimp programming environment. The most recent instalment is devoted to Using Templates.

INTO THE ARC

A regular series for beginners. The latest article explores some aspects of using colour on the Archimedes.

INTRODUCING C

A wide ranging series on the major programming language C.

ARCADE

A round-up of the latest games for the Archimedes: ArcPinball, Interdictor II, The Wimp Game, Professor Mariarti, Bughunter in Space.

As a member of BEEBUG you may extend your subscription to include RISC User for only **£9.10** (overseas see table).

SUBSCRIPTION DETAILS

Destination	**Additional Cost**
UK,BFPO &Ch Is	£ 9.10
Rest of Europe and Eire	£14.00
Middle East	£17.00
Americas and Africa	£19.00
Elsewhere	£20.00

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

RISC User, 117 Hatfield Road, St Albans, Herts AL1 4JS, Telephone (0727) 40303, FAX (0727) 860263

POSTBAG

GOTO FAR

Please allow me to exclaim 'tut tut' on observing the juxtaposition of your article on *Better Programming* (First Course, Vol.9 No.6), and the cover montage which shows, believe it or not a listing with no fewer than ... wait for it ... EIGHT 'GOTO's in about 20 lines! Actually I do not agree with your statement that "A GOTO can never tell you what function within the program is being jumped to." In my (very) early days of programming, I used to put a 'REM <indication>" after GOTO just to do this. I soon discovered the joys of functions and procedures, and hence find the use of GOTO a very rare necessity indeed. I enjoyed your article and found it helpful, not only to me but various younger members of the family, two at least of whom are also subscribers.

I would also welcome (through your pages) to make an appeal for contact with any other English speaking subscribers to BEEBUG near here (Switzerland).

P.L.Stavenhagen

The contrast of style which Mr.Stavenhagen cites in a sense makes the point I was trying to promote, that of encouraging better programming style for BEEBUG readers (and potential contributors?). Putting a REM statement after a GOTO is an excellent method of in-program documentation, but the fact that an additional statement is needed, I think, proves the validity of my own comment, that GOTOs are not self documenting. Readers will find that good Assembler programs are also well commented for the same reason, the cryptic nature of assembler instructions being insufficiently self-documenting.

To some extent, style may be a personal thing anyway, but most experienced programmers agree that there are certain ground rules which should be followed, and that is what I tried to illustrate.

If any readers in (south-west) Switzerland or nearby would like to contact Mr.Stavenhagen his address is 14 Ch. de la Chavanne, 1196 Gland, Vaud, Switzerland.

PROGRAMMING FUNCTION KEYS

Using the *KEYn command to program the function keys works well, but I believe these keys can also be operated in conjunction with Shift, Ctrl, and even Ctrl-Shift, thus greatly extending their use. Some time ago I recall that BEEBUG magazine described how to do this, but I have searched through all my back copies and can find no trace.

Can you therefore please refer me to the appropriate article, or tell me how to program these keys for use with Shift and Ctrl.

Edmund Jupp

To program the function keys as you describe requires you to issue an appropriate FX call first:

```
*FX226,1     Shift + fn
*FX227,1     Ctrl + fn
*FX228,1     Ctrl + Shift + fn
```

*One article covering this topic with regard to its use with View appeared in Vol.4 No.10, and this also contains some information relevant to Wordwise. See also **First Course** in Vol.5 No.10.*

WORDWISE AND THE NEW MASTER OS

With regard to the article by Derek Gibbons in BEEBUG Vol.9 No.6, I would like to re-assure readers that whatever may be the case in View, the new OS ROM for the Master 128 makes no difference at all to the use of Ctrl-Shift-fkeys in Wordwise and Wordwise Plus 2.

It is still possible to include in a function key definition, codes which have a particular meaning in Wordwise Plus, and to implement these with Ctrl-Shift-fkey. Thus, for example, if key 0 is defined with ||!LM7|!, the use of Ctrl-Shift-f0 in edit mode will cause the green command LM7 to be embedded in the text. The sequence ||! generates code 161, simulating the 'normal' f1 ('green') key press as configured in Wordwise Plus, and the sequence || generates code 162, equivalent to a normal 'f2' ('white'). A list of relevant codes is given in Paul Beverley's book *The Complete Wordwise Plus.*

C.W.Robertson

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 40p per word (inclusive of VAT) and these will be featured separately. Please send us all ads (personal and business) to MEMBERS' ADS, BEEBUG, 117 Hatfield Road, St. Albans, Herts AL1 4JS. The normal copy date for receipt of all ads will be the 15th of each month.

Master 65C102 Turbo board with manual and support disc, £55. Acornsoft BCPL ROM with manual and support disc, boxed £20. Originals of Elite, Exile, Holed out, UIM £6 each. Tel. (0752) 896077.

Master Compact entry system, twin disc drives with PSU's, ROM boards, lots of cheap ROMs, stacks of software, Citizen printer, Tandy printer, Joysticks etc. suitable for BBC B. Send a SAE for list to: 9 The Acres, Downley, High Wycombe, Bucks, HP13 5NR. Tel. (0494) 452106.

BEEBUG Magazines Vol. 1 Nos. 8-10, Vol. 2 Nos. 1-8, Vol. 3 Nos. 8-9, Vols. 4 to 7 inclusive, Vol. 8 Nos. 9-10 Vol. 9 Nos. 1-6; £1.20 per issue o.n.o. plus postage (or buyer collects - Gloucestershire). Tel. (0242) 677321 after 6pm.

View Professional £35. Overview £40. ADI ROM £12. Repton II, Repton III, Citadel, Barbarian, Play It Again Sam 13, Graphic Adventure Creator, all discs at £6. HCR EPROM programmer including software, modified for 21/12.5v £20. Scarab RTTY decoder and software £30. Tel. (0263 78) 488.

WANTED: StarBASE Database utilities disc for BBC/Master, mine got lost in moving house, can anyone help?
WANTED: Digital Research's Dos Plus User Guide for Master 512 plus 512 compatible software. **FOR SALE:** Commstar II ROM and manual £10. Tel. 081-684 9340 after 6pm.

Master 128 with manual £270, dual drives 40/80T switchable £70, EPROM Programmer £40. Tel. 021-472 4544 eves.

Disc drive Watford DP35-800S double 40/80T 5.25" and 3.5" drives in plinth unit. As new £100. Tel. (0707) 54311.

BBC Master compact (including 3.5" disc drive and colour monitor), welcome disc, AMX mouse (£70 when new), lots of games and original manuals plus BASIC manual. Make me an offer. Tel. 081-995 2400 after 5.50pm, or w/ends.

BBC Issue 7, Watford DFS 40/80T Watford Single Drive. Microvitec CUB colour monitor. Interword and Dumpmaster ROMs. Watford 32k Shadow RAM board plus lots of disc based software £350. Tel. 081-207 1679 after 6pm.

Printmaster ROM, Pascal ROM, 2xCP ROMs, 2x NTQ font extensions, 2x NTQ drivers & instruction plus 2 discs. Offers accepted. Tel. (0935) 812 682 eves.

BBC B, Shadow RAM, BASIC 2, EPROM board and Sideways RAM, EPROM ZIF Socket, one owner since new £150. Challenger 3 80T floppy disc drive with 512k RAM disc, plus advanced disc investigator £120, monochrome green screen monitor, long persistence, composite and video + sync. £30, or computer with disc drive and monitor for £250. Morley teletext adaptor with ATS ROM £55, Epson FX80+ printer with serial port and NLQ, as new, little use £95, Toshiba P351SX dot matrix printer, new & unused £400, BEEBUG cassette tapes Vols. 1-10 to 8-7 £50, Watford NLQ ROM for Epson £10, Interbase £35, Interword £30, Clares BROM Plus £15, Pen Friend 2 £15, Clares Fontwise Plus and Editor £15, Clares Graphdisk £5, Gemini Database + Beebcalc Spreadsheet + Beebplot £7, Psion Vufile + Vucalc £5, Design 7 Screen Designer £5, Genius Serial Mouse for IBM PC £15. Tel. (0276) 35228.

BBC B issue 7, 40/80T disc drive, Toolkit ROM, joysticks, over 60 disc and tape games inc. Repton, Mini Office II, Elite, Revs, White Knight etc. Micro User from Feb '84 inc. disc, BEEBUG from July '82 Offers to: Mr P Fuller, 110 Lydgate, Burnley, Lancs, BB10 2DU.

BBC B 1.2 with Wordwise Plus, 40/80T disc drive, various games and ed. programmes on tape and disc, computer tape recorder and all leads, software books, monitor (colour) & joystick, all for £375. Tel. 081-883 3271.

2nd Processor 6502 with DNFS and HiBasic ROMs with instruction manual £50, also solderless Sideways ROM socket board £25. Tel. 081-876 4367.

Model B issue 7, software & books £125, Watford 13 ROM board inc 16k SWR £15, PMS B2P 6502 2nd Processor inc HIBasic & HIWW+ great for WP £45, Replay for Watford Mk1 DDFS £15, Cumana 40/80 drive with PSU, hardly used £50, Prism modem £15, Viglen 'PC' case for model B £10, Cumana Touchpad £7.50. Tel. (0707) 276761 (home) or 081-441 6951 (day).

Archimedes A310, colour monitor and various software, excellent condition £500. Tel. (0256) 782619 eves.

BBC B issue 7, Acorn DFS, Acorn single drive, Wordwise Plus and Word Ex, Scythe manager plus lots of other software £210 or any sensible offer. Will deliver reasonable distance (Bury St Edmunds). Tel. (0359) 70942.

FOR SALE: Individually or as job lot:- BBC B issue 7, PC type Watford cabinet complete with Integra B RAM/ROM expansion board with Gem mouse & software (Equates to Master), ZIF socket, cooling fan, twin double sided 40/80 Opus disc drives, c/w ADFS, DNFS & Advanced DFS ROMs £335. Solidisk DFDC inc 8271 £40, Cumana 20Mb model CAB20 hard disc £350, cased Acorn 512k co-processor £100, with full Gem software and mouse. ROMs:- Hyperdriver £25, CP-ROM £25, Word-Ex £20, Printmaster £10, Wordwise Plus II £20, Spellmaster £25, Pen Friend £15, Genie-in-Box £45, ADE £10, ADI £10, AMX Stop Press £30, Acorn Teletext Adaptor c/w ATS+ ROM £35, Miracle WS2000 modem & Commstar II ROM £60, autodial card. Marconi trackerball £30, MEP Bar code reader project £35, Acorn CP/M 2nd processor with Acorn software and manuals £55, Acorn Data Recorder £15, joysticks and games inc. Elite £15, Diagnostics disc £12. Package price £800 including manuals and books. Upgrading to a PC and need shelf space! Tel. (0273) 729506 (eves) or (0273) 200817 (days).

Viglen dual disc drive PSU £20, Vine Micros Replay (for Watford DDFS) £20, Watford File-Plus database, boxed with manual and utilities disc £10, BEEBUG Basic Toolkit ROM & manual £10, Watford DFS ROM £10. All originals. Tel. (0727) 30264.

BBC B with Acorn DFS and 1.2 OS & Basic II +2 Acorn joysticks, Floppy-wise-Plus ROM £120 o.n.o. New Aries B12 ROM board unused with ROMs Sleuth, Toolkit and Dump Out 3 complete with instructions and manuals £30, Seikosha printer GP100A complete with manual & ribbon connector £40 o.n.o. Tel. (074) 488 2643.

ATPL ROM board £10, Watford 32k RAM board £15, Z80 second processor £40, Hybrid Music 500, Island Music System & other music software £50. Tel. (0245) 468707.

Master 128 with 512 co-processor, mouse and Gem software, 40/80T double disc drive, colour monitor, Taxan printer, Master ROM, Wordwise Plus, Master File, Spellcheck III, Wordease, Fontaid, all manuals and heaps of other software. For quick sale only £575. Tel. (0442) 862484.

Cumana 40T DS drive with PSU £30. Interbase £35. Tel. (0525) 715013 after 6pm or weekends.

Single 40/80 DS 5.25" disc drive less PSU-Watford £50, View Printer Driver Generator disc and manual-Acorn £3. Tel. (0227) 711138.

BEEBUG DiscMaster £7, Print Wise £8, Hershey Characters £8, (80T) CC Speech ROM £9, Watford Dumpout III ROM £9, 14 assorted MU 80T discs (inc. Mini Office and classic games) £6. HCR External ROM/RAM system, self powered, holds up to 24 ROMs or 32k RAM (cost £87) £20 o.n.o. plus postage. Canon PW1080-A printer as new with cable (cost £270) £90 plus carriage. Tel. (0275) 462979.

WANTED: Brother 9-pin printer. Tel. (0730) 816786.

All boxed with original manuals, Watford 32k RAM board £20, ATPL ROM board with battery backup £20, STL RTC £15, Cheetah Speech Synth £5, STL DDFS 8271/1770 £20, Basic Extensions ROM £10, ROMIT £10, Iconmaster £10, ROMmanager (WE) £5, or £100 the lot. Tel. (0332) 662955.

BBC B issue 4 OS 1.2, Basic 2, Watford DFS, Teac 40T disc drive, ROM board, ROM programmmer, Voltmace 14B joystick, approx 160 good games around 20 bought recently, also handbooks, around 30 MU magazines and almost a complete set of BEEBUG magazines £230 the lot. Tel. 081-894 2926.

Kaga Taxan Super Vision 625 hi. res. colour monitor £195, excellent condition. Also Holed Out golf game on 5.25" disc £6, Advanced User Guide £5, Kaga KP810 printer still on 1st ribbon £95. Tel. (0268) 693770 eves.

Master series reference manuals parts 1&2, as new £18. Tel. (0442) 64003.

WANTED: Hobbit on disc for M128. Tel. (0628) 482623.

BBC B issue 7 OS 1.2, Watford DDFS dual 40/80T double sided Cumana PSU drive, 14" med. res. colour TV/monitor, 2Mb 128k SWR Shadow RAM, teletext adaptor, Watford ZIF, joystick, EPROM programmer and eraser and blank EPROMs, ISO-Pascal, BCPL, ROMIT, and various disc games (including Exile, Lancelot, Enthar Seven and others) £600 the lot (may split) Tel. (0473) 748300 eves.

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 SUBSCRIPTION RATES		BEEBUG & RISC USER
£18.40	1 year (10 issues) UK, BFPO, Ch.I	£27.50
£27.50	Rest of Europe & Eire	£41.50
£33.50	Middle East	£50.50
£36.50	Americas & Africa	£55.50
£39.50	Elsewhere	£59.50

BACK ISSUE PRICES (per issue)

Volume	Magazine	5"Disc	3.5"Disc
5	£1.20	£4.50	£4.50
6	£1.30	£4.75	£4.75
7	£1.30	£4.75	£4.75
8	£1.60	£4.75	£4.75
9	£1.60	£4.75	£4.75

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

POST AND PACKING

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

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

BEEBUG
117 Hatfield Road, St.Albans, Herts AL1 4JS
Tel. St.Albans (0727) 40303, FAX: (0727) 860263
Manned Mon-Fri 9am-5pm
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by BEEBUG Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Technical Editor: Alan Wrigley
Technical Assistant: Glynn Clements
Production Assistant: Sheila Stoneman
Advertising: Sarah Shrive
Managing Editor: Sheridan Williams

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

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

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

In all communication, please quote your membership number.

BEEBUG Ltd (c) 1991
ISSN - 0263 - 7561

Printed by Arlon Printers (0923) 268328

Magazine Disc

January/February 1991 DISC CONTENTS

- you can highlight any keyword within a text with this program and display further notes related to this word. A demonstration text file is provided.

- a useful program for BEEB owners with limited RAM, which provides a series of star commands on disc for controlling printer styles and settings.

- display or print a calendar month by month for any year after 1753 up to the year 5000.

- a useful utility which allows you to structure Basic programs and offers features like: indentation of statements, splitting multi-statement lines and displaying hex codes of non-printable characters.

- a short function which increases the available program memory by sacrificing a few screen lines in high resolution screen modes.

- use this program to encode and decode simple text messages or complete binary files (programs, data etc.).

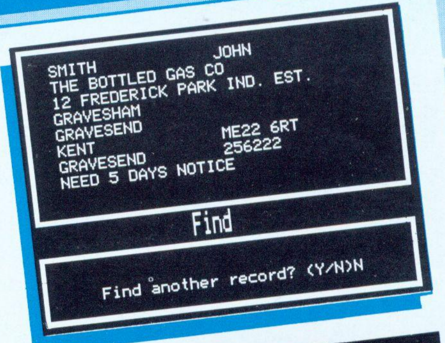
- a program for formatting PC compatible discs on a BBC micro.

- the complete database for storing telephone numbers,names and addresses, and a compaction routine for compressing the data file.

- a short program demonstrating file copying within Assembler.

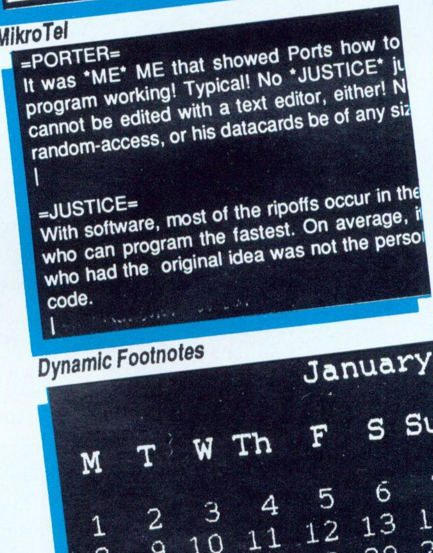
- the last three utilities from this series, which introduce a function key editor and a Basic line editor, and a complete ROM image including some additional features.

- bibliography for this issue (Vol.9 No.8).



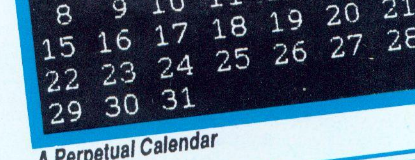
```
                  JOHN
SMITH
THE BOTTLED GAS CO
12 FREDERICK PARK IND. EST.
GRAVESHAM
GRAVESEND           ME22 6RT
KENT                256222
GRAVESEND
NEED 5 DAYS NOTICE

             Find

   Find another record? (Y/N)N
```

MikroTel



```
=PORTER=
It was *ME* ME that showed Ports how to
program working! Typical! No *JUSTICE* ju
cannot be edited with a text editor, either! N
random-access, or his datacards be of any siz
|
=JUSTICE=
With software, most of the ripoffs occur in the
who can program the fastest. On average, i
who had the  original idea was not the persor
code.
|
```

Dynamic Footnotes



```
                         January
 M   T   W  Th   F   S  Su
                 5   6   7
 1   2   3   4  12  13  14
 8   9  10  11  19  20  21
15  16  17  18  26  27  28
22  23  24  25
29  30  31
```

A Perpetual Calendar

ALL THIS FOR £4.75 (5.25" & 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) available at the same prices.

DISC (5.25" or 3.5") SUBSCRIPTION RATES	**UK ONLY**	**OVERSEAS**
6 months (5 issues)	£25.50	£30.00
12 months (10 issues)	£50.00	£56.00

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

BEEBUG, 117 Hatfield Road, St.Albans, Herts AL1 4JS

Special Offers to BEEBUG Members Jan/Feb 1991

BEEBUG'S OWN SOFTWARE

Code	Product	Members Price inc.VAT	Code	Product	Members Price inc.VAT
1407a	ASTAAD3 - 5" Disc (DFS)	**5.95**	PAG1	Arcade Games (5.25" 40/80T)	5.95
1408a	ASTAAD3 - 3.5" Disc (ADFS)	**5.95**	PAG2	Arcade Games (3.5")	5.95
1404a	Beebug Applics I - 5" Disc	**4.00**	PBG1	Board Games (5.25" 40/80T)	5.95
1409a	Beebug Applics I - 3.5"Disc	**4.00**	PBG2	Board Games (3.5")	5.95
1411a	Beebug Applics II - 5" Disc	**4.00**	0077b	C - Stand Alone Generator	14.25
1412a	Beebug Applics II - 3.5" Disc	**4.00**	0081c	Masterfile ADFS M128 80 T	16.50
1421b	Beebug Binder	4.20	0024c	Masterfile DFS 40 T	16.50
1600a	Beebug magazine disc	4.75	0025c	Masterfile DFS 80 T	16.50
1405a	Beebug Utilities - 5" Disc	**4.00**	0085c	Printwise 40 Track	22.50
1413a	Beebug Utilities - 3.5" Disc	**4.00**	0086c	Printwise 80 Track	22.50
1450a	EdiKit 40/80 Track	5.75	0009c	Studio 8	16.50
1451a	EdiKit EPROM	7.75	0074c	Beebug C 40 Track	44.25
1452a	EdiKit 3.5"	5.75	0075c	Beebug C 80 Track	44.25
0005b	Magscan Vol.1 - 8 40 Track	12.50	0084c	Command	29.25
0006b	Magscan Vol.1 - 8 80 Track	12.50	0073c	Command(Hayes compatible)	29.25
0011a	Magscan Update 40 track	4.75	0053c	Dumpmaster II	23.25
0010a	Magscan Update 80 track	4.75	0004c	Exmon II	24.00
			0087c	Master ROM	29.25

*Please add p&p. **UK** : **a** - 60p, **b** - £1.50, **c** - £2.50 , **Europe:** **a** - £1.00, **b** - £1.50, **c** - £9.30*

OTHER MEMBERS OFFERS

Due to a popular demand we are extending the offers from last month. The products listed below will be available for a limited period only while stocks are available.

LISP

RRP £20.00 (inc VAT)
Offer price £2.99 (inc VAT) + £0.60 p&p

The fundamental LISP language on ROM (no manual).

Stock code 1003a

FORTH

RRP £20.00 (inc VAT)
Offer price £2.99 (inc VAT) + £0.60 p&p

A complete implementation of the FORTH language on ROM (no manual).

Stock code 1041a

View 3 ROM

Normal Members price £51.75 (inc VAT)
Offer price £12.50 (inc VAT) + £2.50 p&p

The enhanced version of the excellent View wordprocessor from Acorn. Supplied with manual,keystrip and printer driver generator on disc or tape. Many features over the original version of View.

Stock code 1022c

ViewSheet

Normal Members price £42.49 (inc VAT)
Offer price £12.50 (inc VAT) + £1.50 p&p

Acorn's excellent spreadsheet for the BBC micro and Master, supplied on a 16K ROM. A very powerful and extremely useable product which will produce results ready to print or for direct merging into View text files.

Stock code 1001b

ViewSpell

Normal Members price £28.75 (inc VAT)
Offer price £7.00 (inc VAT) + £1.50 p&p

The automatic spelling checker with a built-in 75 000 word dictionary. Supplied on ROM with full manual, examples disc and reference card. Ideal for View or ASCII text files, and can use updateable user dictionaries.

Stock code 1043b

ViewStore

Normal Members price £42.49 (inc VAT)
Offer price £13.80 (inc VAT) + £1.50 p&p

ViewStore is Acorn's database manager program. It offers 40 and 80 column display, multiple indexes, detailed report generation, variable field lengths and much, much more. Excellent value for money.

Stock code 1019b