Vol.9 No.3 July 1990

BEEBUG

FOR THE BBC MICRO & MASTER SERIES

Phone Call Costing

- ENIGMA: WORLD WAR II CODING MACHINE
- USING BAS128
- HIGH RESOLUTION GRAPH PLOTTER
- TABLES INVADERS

BEEBUG Vol.9 No.3 July 1990

FEATURES

Enigma:
A World War II Coding Machine 6
BEEBUG Survey:
Word Processing (Part 1) 10
Phone Call Costing 15
Thanks for the Memory Bas128
(Part 1) 20
First Course -
A Data Input Routine 23
A High Resolution Graph Plotter 27
Practical Assembler (Part 3) 34
BEEBUG Tables Invaders 37
512 Forum 41
Using the ROM Filing System
(Part 3) 47
BEEBUG Education 50
Permanently Coloured Text 52
BEEBUG Prize Crossword 54
Automating Assembler Input 55

REVIEWS

Star XB24 Printer 32
Crossword 45

REGULAR ITEMS

Editor's Jottings 4
News 5
RISC User 44
Points Arising 49
EdiKit 56
Hints and Tips 57
Postbag 59
Personal Ads 60
Subscriptions & Back Issues 62
Magazine Disc/Cassette 63

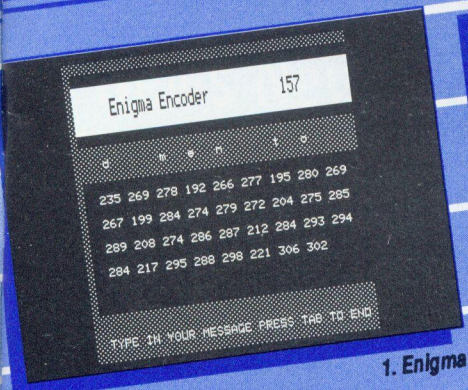
HINTS & TIPS

EXEC Command Tails
Transferring View Files to PC Systems
Using the 512

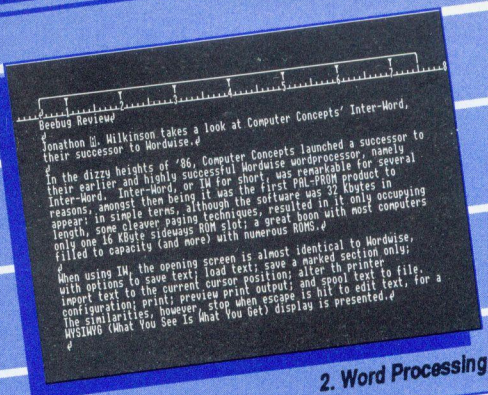
PROGRAM INFORMATION

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 |.

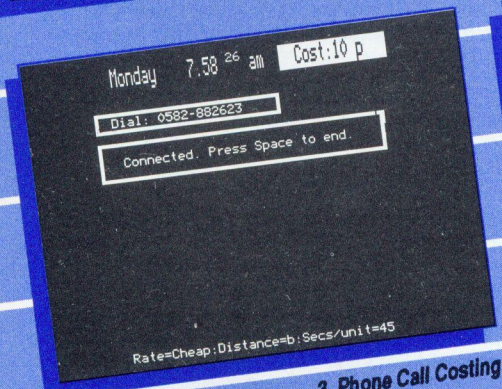
All programs in BEEBUG magazine will run on any BBC micro with Basic II or later, unless otherwise indicated. Members with Basic I are referred to the article on page 44 of BEEBUG Vol.7 No.2 (reprints



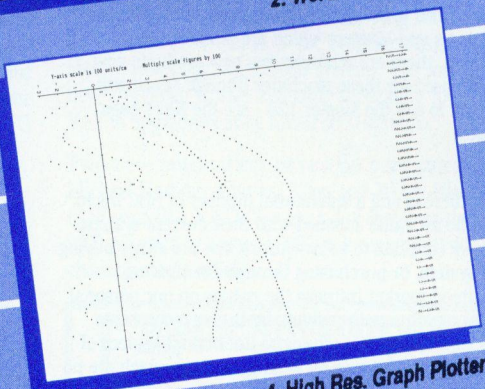
1. Enigma



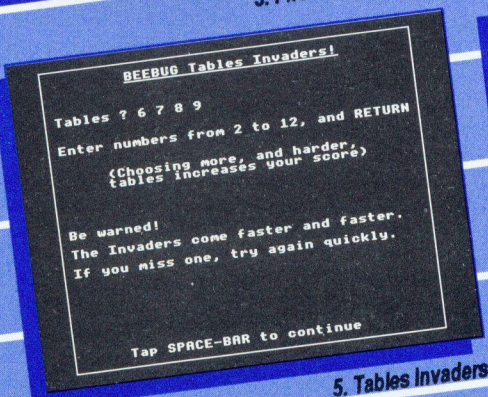
2. Word Processing



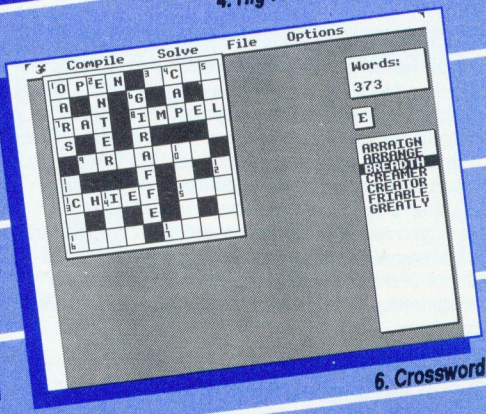
3. Phone Call Costing



4. High Res. Graph Plotter



5. Tables Invaders



6. Crossword

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

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

Program will not function on a cassette-based system.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings

BEEBUG PRODUCT SURVEYS

Most BBC micro users are probably aware that these days it is Acorn's Archimedes range which steals the limelight when it comes to new products. Despite worldwide sales of BBC micros and Masters which are probably in excess of 1.5 million, there are relatively few new products now being released for these machines (though Acorn still continues to sell the Master 128 - see the News page opposite).

Clearly there is still a substantial number of BBC micro users who are quite satisfied with their current system, and there continue to be new users who are either buying a new system, or purchasing through the thriving secondhand market (witness the entries on our personal ads pages each month - which, by the way, can be as useful for locating that otherwise unobtainable piece of software or hardware as for selling off items which are no longer needed).

For the benefit of both new and existing users, we have therefore decided to embark on a series of major surveys. Each survey will deal with a major software application, or a significant hardware expansion. We have decided to start with that most popular of applications, word processors, where there are five major families of products still available to BBC micro and Master users (the View word processor is of course supplied as an integral part of the Master 128 and Master Compact systems). Indeed several of these products have now been released in versions for the Archimedes range, such is their abiding popularity.

It is not out intention in these surveys to provide detailed comparative reviews of all products, but more to make all users aware of what is available, and to cover the essential features and characteristics of each product dealt with. Because word processing is a major topic we have had to split this first survey into two parts.

Future surveys which we are currently planning are likely to include spreadsheets, databases, printers, utility ROMs, communications, graphics and music. We would welcome further feedback on this subject, and you, the readers, can help us in a number ways. You could suggest other topics which you would like to see covered by a survey; more particularly you may know of or use a product yourself which you consider it is vital to include in a future survey; and if you are a regular user of any such product with some talent for writing, then we shall certainly be looking contributions on individual products.

In all cases concerned with the content of the magazine, you should address your communication to the Editor.

BEEBUG MAGAZINE PRODUCTS

Having indicated that the numbers of new products for the BBC micro are few and far between, we have ourselves produced a number of software items over the last eighteen months. The most recent is *EdiKit*, effectively a programmer's toolbox, which combines as a single ROM our previous Basic Booster together with all the EdiKit functions described in the magazines from Vol.8 No.7 to Vol.8 No.10. As a result, Basic Booster will no longer be available separately. Full details on EdiKit are given elsewhere in this issue.

Let me also remind you of the collections of programs which we have also published, *General Utilities*, *Applications I* and *Applications II*, all available to BEEBUG members at £5.75 each (plus 60p p&p). In addition we also have *ASTAAD3*, our popular CAD package for the Master 128 and Master Compact (price £9.95 plus 60p p&p), and the long standing and equally popular Filer database package (at £5.75 plus 60p p&p). All these products derive from original magazine programs and articles, and represent excellent value for money.

NEXT ISSUE OF BEEBUG

It is our intention that each issue of BEEBUG should normally reach you at or by the beginning of each month referred to on the cover. Thus this issue should arrive at the beginning of July. The next issue of BEEBUG is a two month issue for both August and September. Delivery will not therefore follow the normal pattern, and the Aug/Sept issue should be mailed to you mid August.

News News News News News News

ACORN NEWS

Further to the announcement of reduced prices for Acorn's Archimedes range given on our News page in Vol.9 No.1, there has also been a price reduction on the Master 128 which now retails at £399 ex. VAT (£458.85 inc. VAT). Despite all the publicity for the Archimedes range, Acorn is still making and selling the Master 128 in good numbers.

We can also confirm that Acorn will be present at both the BBC Acorn User Show 7th - 9th September, and at the Computer Shopper Show 6th - 9th December (BEEBUG will also be at both shows).

Acorn Computers Ltd is at Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN, tel. (02239 245200. Information on Acorn products may be obtained from BEEBUG, who are official Acorn dealers.

EXPANDING THE 512

Essential Software, the company which has released a number of software products for the 512 co-processor (and in which our 512 Forum author, Robin Burton, is involved) has announced its first hardware add-on for the 512. This is a memory expansion board costing £99.00 inclusive. This covers fitting the expansion to your original 512 board, which must be returned to Essential Software for the work to be carried out and tested. Orders are scheduled so that 512s are returned to users on the day of receipt.

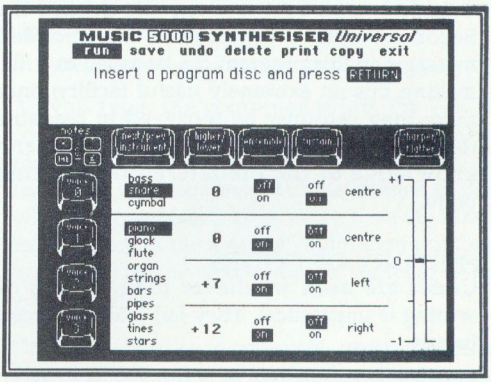
With the expansion fitted, a 512 running DOS Plus 2.1 gives 614K of free RAM for applications (more free RAM than a 640K PC), and the new version of Essential's Ramdisc (£16.95, or £2.00 plus the original disc for an upgrade) provides a RAM disc of up to 128K at the same time, with no loss of user RAM in expanded machines. This gives the user access to nearly 760K bytes.

For more information, or to arrange an upgrade, contact Essential Software, P.O.Box 5, Groby, Leicestershire LE6 0ZB, tel. (0530) 243773.

MUSIC FROM HYBRID TECHNOLOGY

Hybrid Technology's Music 5000 Universal is now available. Costing £113.85 (inc. VAT), this device can be connected to any Master series machine and will greatly enhance the sound generated by any suitable music application by substituting its own digitally synthesized instruments for the micro's 'beep'. The sound can be channelled through the micro's own speaker, or for the best results through an external amplifier and speakers. As well as enhancing the output of any existing music package, the Universal will also handle the growing range of Ample music software (Ample is the programming language of Hybrid's existing Music 5000 system). The software of the new Universal is also available separately for £33.35 inc. VAT to upgrade existing Music 5000 systems.

Two new music discs have also been released, both Michael Harbour albums, *Ashes* and *Windy Island*, costing £4.95 each (inc. VAT). The second of these requires a Music 3000 upgrade to a Music 5000 as it makes full use of all 32 channels then available.



Music 5000 Universal main options screen

A further venture by Hybrid is *Sound World*, a new family of Ample music packages aimed at educational users. The first two titles are *soundscape* and *soundspace*, costing £33.35 each inc. VAT. The first also needs the Music 4000 keyboard and lets children explore, compose and perform sounds. *Soundspace* requires a touch screen or joystick and is aimed at children with special needs. Additional titles in preparation include *soundscene* and *soundstory*.

Further details on all these products may be obtained from Hybrid Technology Ltd, 273 The Science Park, Cambridge CB4 4WE, tel. (0223) 420360.

Enigma: A World War II Coding Machine

Derek Greenacre tells an intriguing tale of wartime secret codes, and their simulation on a BBC micro.

INTRODUCTION

There have been many demonstration programs of secret codes written for the home computer. However, the one presented here is different in many respects. Unlike most programs of this nature, this one has been written because it is a good encoding technique and not just a demonstration of how easily the computer can re-arrange letters (a facility of which we are all aware anyway). In other words, once encoded, the message from this program is extremely difficult to crack.

Secondly, the program is of historic interest as the program, to a large extent, emulates the working of the Enigma Machine, a coding machine used by the Germans during the Second World War. Lastly, the encoded messages in this program can be saved to a file, making this an extremely useful facility; once there they can only be retrieved in readable form by having access to this program and knowledge of the predetermined starting point.

HISTORY OF CODES

Codes are used whenever plain speech or writing is impractical. They fall into two broad camps:

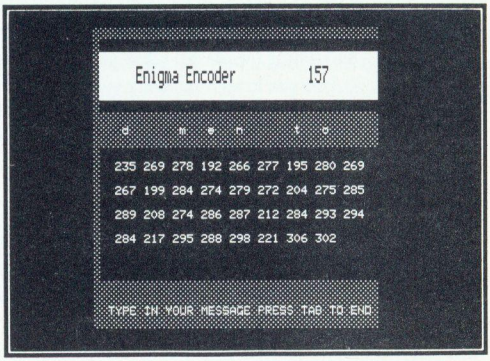
1. Those intended to be easily understood (e.g. Morse, Braille, Basic, Semaphore etc.).

2. Those intended to be difficult to understand. These tend to be collectively known as *cryptograms*.

The earliest known cryptology book dates from 1499, although the use of codes goes back to much earlier civilisations. Secret codes themselves fall into two main methods:

(i) *Substitution* (where another letter or symbol replaces the original letter of the text).

(ii) *Transformation* (where the letters of the text remain constant, but the order in which they are presented is rearranged).



Coding a message

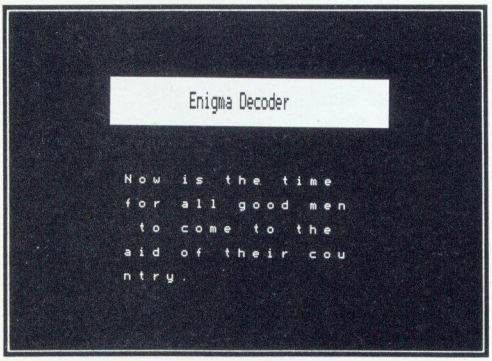
Other methods do exist. For example, Baden Powell incorporated the gun emplacement positions of the Boers into the patterns on the butterflies' wings he was sketching. More recently, Kit Williams brilliantly hid the clues to the whereabouts of a golden hare in paintings in his book, and challenged the world to find it.

THE ENIGMA MACHINE

The Second World War saw feverish attempts to develop codes which could be easily transmitted and received by one's own forces yet remain totally obscure to the enemy. With this in mind, the Germans invented the Enigma Machine. Codes are normally broken by recognising the recurrence of familiar patterns of letters and guessing a particular word.

Using that informed guess, one then uses the resultant letters as a starting point to guess another word. With each new word, more decoded letters are added to the vocabulary. The Enigma Machine as we shall see brilliantly foils that approach.

The machine consisted of a typewriter keyboard, a screen, and (in the case of the Luftwaffe) 3 numbered wheels. These wheels had brass studs at each end of them connected by wires to each other in a complex manner. The touching of adjacent wheels allowed an electric current to be passed (on the depression of a key) from the keyboard to illuminate a letter on the screen. The path of the current, and therefore the illuminated letter, was determined by the complex nature of the wiring of the wheels and their position relative to one another. This in itself produces a coded message which is determined by the original setting of the wheels (something which could and indeed was, changed every day).



Decoding a message

The real ingenious part of the machine, however, lay in the fact that each time a key on the keyboard was depressed the wheel rotated one place until having completed a revolution caused its neighbour to rotate one place in turn (rather like the mileometer on a car). Because the position of the wheels determined the path of the current this had the effect of encoding the repetition of the same letter in a message with a different symbol every time, thus making it extremely difficult to crack the code.

With an understanding of the complexity of the Enigma Machine, one cannot fail to admire the brilliance of the British cryptologists working at Station X (Bletchley Park) during the war who every day had to figure out the workings and starting points of the wheels from the short coded messages transmitted in German.

THE COMPUTER PROGRAM

Obviously the computer cannot hope to emulate every aspect of the Enigma Machine, but the keyboard and the screen suggested that they had sufficient in common to attempt a simulation.

Type in the program listed here and save it to disc. On ADFS systems omit the 'REM' from line 2020. Then run the program. Firstly, try encoding a message by pressing option 1, when you will be asked to type in a number (very important - don't forget it). This is the equivalent of setting the starting position of the 3 wheels. The program creates a file on disc (equivalent to transmitting the coded message) so a file name is also called for. Now the encoding begins. Notice that the starting number appears at the top of the screen. Try a simple message to begin with, say:

```
"An example of a coded message"
```

As you type, the computer prints the ASCII code number of the value of the character plus the starting number and then adds one for each letter you type (equivalent to the rotation of the wheel by one place). Thus every time the same letter occurs in the message, a different number is encoded on the screen (look at the letter 'e' in your sample).

Once the message is complete, press Tab. This gives you the option to save to disc. Once saved, the encoded message can be retrieved from the file by selecting option 2 (the equivalent of receiving a message). Type in the same starting number as before, and the file name and the decoded message should appear. On completion, press any key to return to the main menu.

Typing an incorrect starting number will result in a wrongly decoded message. In other words, the computer program is the equivalent of the Enigma Machine itself, and the number requested is equivalent to the initial starting position of the wheels. Using a separate disc for the files and the program thus enables pieces of

information to be stored in safety, knowing they cannot be decoded without the program.

Incidentally, the file can easily be accessed by writing a short "OPENIN" routine (or just use *TYPE). Looking at the file in this way shows clearly how complex the coding is even when you know what the message reads.

The constant "rotate%=1" at line 140 in the program causes the simulated rotation of the wheels. Changing this to another value offers the operator an even more sophisticated coding option, which was desirable on the original machines, but impracticable as they were essentially mechanical rather than electronic devices.

Lastly, make sure you remember the starting numbers of the files you create, for even though you still have this program they cannot easily be retrieved without them. More importantly, make sure this program is kept safe or the information in the files is surely lost.

The magazine disc contains a coded file Freya (use 137 to decode) containing the first complete message to be successfully decoded by British Intelligence on 27th June 1940.

Recommended reading:

Most Secret War by Professor R.V.Jones published by Coronet in paperback.

```
  10  REM Enigma Coding Machine
  20 REM Version B1.5
  30 REM Author  Derek Greenacre
  40 REM BEEBUG  July 1990
  50 REM Program subject to copyright
  60 :
 100 MODE7:CLOSE#0
 110 MAX%=300:DIM AR$(MAX%)
 120 rotate%=1
 130 P%=4
 140 ON ERROR:CLOSE#0:IF ERR<>17 MODE7:
VDU7:PROCerror:END
 150 VDU23,1,0;0;0;0;
 160 REPEAT
 170 VDU26:CLS
 180 G%=FNstart:T%=1
 190 IF G%=2 THEN PROCdecode
 200 IF G%=1 THEN PROCencode
 210 UNTIL G%=3
 220 MODE7:END
 230 :
```

```
 1000 DEF PROCencode
 1010 REPEAT:VDU26:ST1%=ST%
 1020 CLS:PROCframe
 1030 PROCprntdouble(P%,3,129,"Enigma En
coder")
 1040 PRINTTAB(P%+25,3)CHR$132;ST1%
 1050 PRINTTAB(P%+25,4)CHR$132;ST1%
 1060 PRINTTAB(0,23);CHR$133;CHR$136;"TY
PE IN YOUR MESSAGE PRESS TAB TO END"
 1070 VDU28,0,21,39,8
 1080 X%=2:Y%=3
 1090 REPEAT
 1100 G%=GET:IF G%<>9 PROCenter
 1110 UNTIL G%=9 OR Y%>12
 1120 UNTIL G%=9
 1130 VDU26:CLS:PROCframe
 1140 PRINTTAB(3,4);CHR$132;"Do You Wish
 To Save This Message?":VDU7
 1150 G$=GET$:PRINTTAB(3,4);SPC(33)
 1160 IF G$="Y" OR G$="y" PROCfile
 1170 ENDPROC
 1180 :
 1190 DEF PROCenter
 1200 IF ST%>600 ST%=0
 1210 G$=CHR$(G%):G%=G%+ST%
 1220 CODE$=STR$(G%):IF LEN(CODE$)<3 COD
E$="0"+CODE$
 1230 AR$(T%)=CODE$
 1240 ST%=ST%+rotate%
 1250 PRINTTAB(X%,Y%);CHR$134;CODE$
 1260 T%=T%+1
 1270 IF T%>(MAX%-2) VDU26:CLS:VDU7:PRIN
TTAB(4,12);CHR$130;"File Full - Press An
y Key":T$=GET$:CLS:ENDPROC
 1280 PRINTTAB(X%+1,0);CHR$131;G$
 1290 IF X%>32:TIME=0:REPEAT:UNTIL TIME>
100:PRINTTAB(0,0);SPC(38):Y%=Y%+2:X%=-2
 1300 X%=X%+4
 1310 ENDPROC
 1320 :
 1330 DEF PROCerror
 1340 REPORT
 1350 PRINTTAB(6,9);CHR$133;"THE ERROR I
S AT LINE ";ERL
 1360 ENDPROC
 1370 :
 1380 DEF PROCtop
 1390 PRINTTAB(0,1);CHR$129;CHR$157
 1400 FOR Y%=2  TO 6:PRINTTAB(0,Y%);CHR$
131;CHR$157:NEXT
 1410 PRINTTAB(0,6);CHR$129;CHR$157
 1420 ENDPROC
 1430 :
```

```
 1440 DEF PROCframe
 1450 PRINTTAB(0,1);CHR$132;CHR$157
 1460 FOR Y%=2 TO 6:PRINTTAB(0,Y%);CHR$1
31;CHR$157:NEXT
 1470 PRINTTAB(0,6);CHR$132;CHR$157
 1480 FOR Y%=10TO20:PRINTTAB(0,Y%);CHR$1
32;CHR$157:NEXT
 1490 ENDPROC
 1500 :
 1510 DEF PROCdecode
 1520 Z=OPENIN Name$
 1530 IF Z=0 PRINTTAB(3,24)"NO SUCH FILE
 - PRESS ANY KEY ";:G%=GET:ENDPROC
 1540 REPEAT
 1550 CLS:PROCbac
 1560 I%=2:J%=10
 1570 REPEAT
 1580 INPUT#Z,AR$
 1590 IF DC%>600 DC%=0
 1600 AR%=VAL(AR$)-DC%
 1610 PRINTTAB(I%,J%);CHR$130;CHR$(AR%)
 1620 I%=I%+2
 1630 DC%=DC%+rotate%
 1640 IF AR%=13 I%=2:J%=J%+2
 1650 IF I%>32 I%=2:J%=J%+2
 1660 UNTIL J%>20 OR EOF#Z
 1670 IF J%>20:PRINTTAB(2,22);CHR$131;"P
RESS SPACE BAR FOR REST OF MESSAGE":G%=G
ET
 1680 UNTIL EOF#Z
 1690 PRINTTAB(0,22);SPC(37):PRINTTAB(10
,22);CHR$134;CHR$136;"End of Message"
 1700 CLOSE#Z:G%=GET
 1710 ENDPROC
 1720 :
 1730 DEF PROCfile
 1740 PRINTTAB(8,3);CHR$132;"Please Wait
 a Moment"
 1750 A=OPENOUT Name$
 1760 FOR B=1 TO T%-1
 1770 PRINT#A,AR$(B)
 1780 NEXT
 1790 CLOSE#A
 1800 ENDPROC
 1810 :
 1820 DEF FNstart
 1830 PROCboxes
 1840 PROCprntdouble(7,3,129,"Enigma Cod
ing Machine")
 1850 PROCtop
 1860 PRINTTAB(4,9);CHR$134;"Do you wish
 to:"
 1870 PRINTTAB(7,12);CHR$133;"1. Encode
a message"
 1880 PRINTTAB(7,14);CHR$133;"2. Decode
a message"
 1890 PRINTTAB(7,16);CHR$133;"3. End"
 1900 REPEAT:G%=GET-48:UNTIL G%>0 AND G%
<4
 1910 PRINTTAB(7,10+2*G%)CHR$131
 1920 IF G%=3 =G%
 1930 VDU28,0,24,39,20
 1940 PRINTTAB(2,0)CHR$130"TYPE IN A STA
RTING NUMBER 0 to 200"
 1950 PRINTTAB(10,3)CHR$130;"THEN PRESS
RETURN"
 1960 REPEAT:PRINTTAB(0,2)SPC10:INPUTTAB
(0,2)ST%:UNTIL ST%>=0 AND ST%<=200
 1970 DC%=ST%:CLS
 1980 PRINTTAB(2,0)CHR$130"NOW GIVE THE
FILE A NAME"
 1990 INPUTTAB(0,2);Name$
 2000 PRINTTAB(2,4)CHR$131;"Insert disc
and press any key";:G1%=GET:VDU26
 2010 REM Next line for ADFS only.
 2020 REM*MOUNT0
 2030 =G%
 2040 :
 2050 DEF PROCbac
 2060 PRINTTAB(0,1);CHR$132;CHR$157
 2070 FOR Y%=2  TO 5:PRINTTAB(0,Y%);CHR$
134;CHR$157:NEXT
 2080 PRINTTAB(0,6);CHR$132;CHR$157
 2090 PRINTTAB(10,3);CHR$132;CHR$141;"En
igma Decoder"
 2100 PRINTTAB(10,4);CHR$132;CHR$141;"En
igma Decoder"
 2110 ENDPROC
 2120 :
 2130 DEF PROCboxes
 2140 col%=148:YY%=11
 2150 PRINTTAB(4,YY%);CHR$(col%);"j£££££
£££££££££££££££5"
 2160 REPEAT
 2170 PRINTTAB(4,YY%+1)CHR$(col%);"j";SP
C(22);CHR$(col%)"5"
 2180 YY%=YY%+1
 2190 UNTIL YY%>15
 2200 PRINTTAB(4,YY%+1);CHR$(col%);"jppp
ppppppppppppppppp5"
 2210 ENDPROC
 2220 :
 2230 DEF PROCprntdouble(x,y,c,t$)
 2240 PRINTTAB(x,y)CHR$(c)CHR$141t$
 2250 PRINTTAB(x,y+1)CHR$(c)CHR$141t$
 2260 ENDPROC
```

BEEBUG Survey
Word Processing

INTRODUCTION

This is the first of a number of surveys which we will be publishing in this and forthcoming issues of BEEBUG magazine. The aim is to document and survey the products available for major software applications and as hardware add-ons. According to last year's Reader Survey, word processing is the single most popular application for users of BBC micros (and no doubt for many other micros as well), so what better place to start?

Word processors have a long history on the BBC micro. *Wordwise* and *View* appeared fairly early on; both have subsequently been improved. *Inter-Word*, from the same stable as Wordwise, appeared later, as did *View Professional* and *Wordpower*. All of these in one form or another are still available (indeed some have been versioned for the Archimedes as well), and no doubt many purchasers of secondhand machines often find a word processor of some kind included in the price they pay.

Our aim in this survey is to invite individuals who are familiar with each word processor to explain why they believe it is so good, and to describe what they see as its main characteristics and features. This is not therefore intended to be a comparative review, but an attempt to present an up-to-date picture of all the major word processors currently available.

We begin with Wordwise, and its successors and derivatives, then move on to Inter-Word (from the same source), and we round off part one of our word processor survey with a look at View. Wordpower and View Professional follow in part 2 of this particular survey.

Note that all prices include VAT, and all products (except Wordwise Plus II) are available from BEEBUG. Different prices may apply from other suppliers.

Wordwise and Family

Ian Waugh reports on Computer Concepts' Wordwise and Wordwise Plus, and its latest enhancement, Wordwise Plus II from IFEL.

Wordwise (Computer Concepts) £27.31
Wordwise-Plus (Computer Concepts) £43.70
Spellmaster (Computer Concepts) £52.25
Wordwise-Plus II (IFEL) £56.35
Upgrade to Wordwise-Plus II options exist for current Wordwise and Wordwise Plus users. Contact IFEL for details.

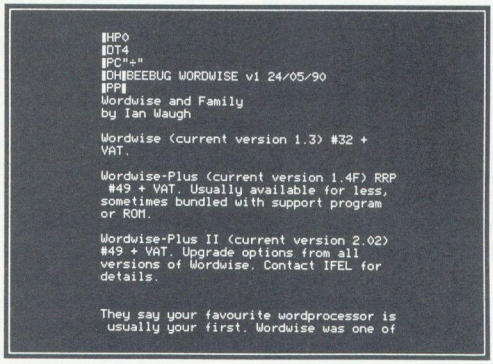
They say your favourite word processor is usually your first. Wordwise was one of the first professional word processors for the BBC micro, and after looking at the others, I plumped for Wordwise.

Why? Mainly because Wordwise seemed the easiest to use, but it has since turned out to be one of the most powerful word processors for the BBC micro. That early version, however, wasn't perfect (beware of anyone who says they've found the perfect word processor) but it has been considerably improved and enhanced, and now Wordwise-Plus II comes pretty close.

It's main disadvantage, say its critics, is that it is not WYSIWYG (that is, what you see laid out on screen is not the same as the version eventually printed out to paper). I agree. Text is entered in a mode 7 screen and formatting instructions for line and page length, tab settings and so on are entered between embedded command markers which are inserted with the function keys. For example, to set the line length to 60 columns, you press f1 (*embedded command start*), enter LL60 then press f2 (*embedded command end*). There are almost 50 embedded commands offering a large range of formatting instructions and printer commands, but even if you don't enter anything you get sensible defaults. *Embedded command start* makes the text following it green so that it's easy to see where the commands are (less so on a monochrome screen though).

You can still see what the text will look like on the page in the 80-column *Preview* mode. It doesn't

show bold or italicised text but then neither do many WYSIWYG word processors. Being able to see the layout is generally sufficient confirmation for most applications. Most 'straight text' documents such as letters, reports, articles, books and plays follow a particular format, and once the layout has been created all you basically have to do is to type in the words. I have written several books and hundreds of articles and letters using Wordwise and its progeny.



Wordwise edit screen

However, creating a column-based document over 40 characters wide is more difficult, especially if embedded commands are used. It usually involves frequent trips to the preview screen. But, again, once you've created a layout it's there to re-use whenever you need it. I have templates for invoices, order forms, letters and articles.

Working in mode 7 has its advantages, too - it's easy to read, particularly on a TV where 80-column text can be a strain, and it's extremely fast. You can scroll through the text very quickly and smoothly without any of the sluggishness you sometimes get with other word processors, especially when memory is almost full.

Wordwise has *Insert* and *Overwrite* modes. Insert pushes existing text to the right as you enter new characters and is probably the most useful. However, Overwrite is excellent for typing column-based documents or ones with fixed layouts such as forms and invoices.

The function keys allow you to delete, move and copy text, and they can store your own functions, too. I've set up two to automatically underline and italicise a word. I had a key which reversed two adjacent letters - like thsi (one of my special typing foibles) - but Wordwise-Plus II (see later) has a dedicated command for this which made it redundant.

Sometimes it's necessary to press the Break key (occasionally the best-behaved computer locks up - I've never traced this to any fault with Wordwise) and it's reassuring to know that unless the lock-up is extremely serious, the text will remain intact. You can even press Ctrl and Break, re-enter Wordwise and find the text is still there - but don't make a habit of it. Robust is the word.

Wordwise also has features which, as a writer, I consider essential - a running word count in the top left of the screen and a Word Count To? function which can be used to count a section of text - not all other word processors can do this. It also shows free memory in the top right of the screen.

SON OF WORDWISE

After Wordwise came Wordwise-Plus. This introduced the idea of *Segments* which are simply separate edit areas - ten in all - in which you can store separate texts. But, and here's the novelty, Wordwise-Plus has its own programming language and programs can be stored in Segments and executed from them. This really adds another dimension to word processing.

Many dozens of Segment programs exist for which Norwich Computer Services is a major supplier. Typical applications include alphabetical sorting and list processing, indexing, the creation of address lists and label printing. There are also sophisticated mail-merge programs and a routine to print multiple copies of a document. I've even seen a random sentence generator!

Other programs include an invoice generator, a continuous processing routine and a program to print several documents continuously. If you have a particular application, a Segment program could probably be written to do it. It can make simple processes even simpler. For example, when I write a letter, being the lazy person that I am, I use a Segment program to delete the body of the text and leave the recipient's address suitably indented to print in the centre of the envelope. Neat, eh?

SON OF WORDWISE-PLUS

Wordwise-Plus II was developed by IFEL in agreement with Computer Concepts. The Plus II ROM adds a great many extra features to Wordwise which, I confess, I would now find very difficult to live without. My main joy is menu-driven file handling. This lists all the files on a disc and these can be selected with the cursor keys. It also lets you step through directories which is particularly useful when using the ADFS.

My other most-used functions are the greatly enhanced *Goto* and *Search and Replace* options, and the ability to alter the key repeat and delay rates which I usually set to 4 and 18 respectively.

Other Plus II features include an on-screen list of the main formatting embedded commands, with explanations. These can be altered and inserted anywhere in the text. Printer set up codes and print options can be entered just as easily. All these work from pop-up menus - brilliant! There are additional embedded commands to print the date and time (just to let the Ed know you're still working at two o'clock in the morning) and it includes several additions to the Wordwise Programming Language.

It interfaces nicely with Computer Concepts' Spellmaster which is highly recommended, too. Finally, several Wordwise support ROMs are available with functions ranging from printer routines and utilities to additional programming functions.

In terms of power and sheer versatility no other BBC word processor comes close to Wordwise-Plus II. And it's still easy to use.

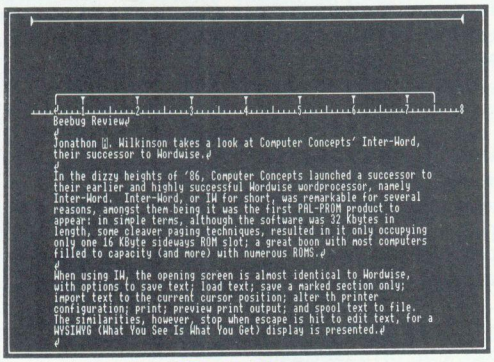
Inter-Word

Jonathon A. Wilkinson takes a look at Computer Concepts' Inter-Word, its successor to Wordwise.

Inter-Word (Computer Concepts) £47.52

In the dizzy heights of '86, Computer Concepts launched a successor to its earlier and highly successful Wordwise word processor, namely Inter-Word (IW for short).

When using IW, the opening screen is almost identical to Wordwise, with options to save text; load text; save a marked section only; import text to the current cursor position; alter the printer configuration; print; preview print output; and spool text to file. The similarities, however, stop when Escape is pressed to edit text, for a WYSIWYG (What You See Is What You Get) display is then presented.



Inter-Word edit screen

A ruler bar displays the settings for the left and right margin, paragraph indentation margin, and the tab stops. All of these are easily changed by simply placing the cursor over the symbol representing the parameter, e.g. T for tab stop, and then literally dragging it across to its new position; absolute simplicity itself.

Being a WYSIWYG-type display, as text is entered, it is automatically shown as it will be printed out. By pressing function keys f4 to f7, the text can be left justified, centred, right justified, or blocked as required. Different print attributes can be applied by pressing Shift in combination with the same function keys to produce underscore, bold, dotted (i.e. italic), and normal.

Access to IW's features, via function keys, results in pull-down menus appearing for the appropriate options to be selected, or data entered.

A number of different options can be selected under the *Preferences* menu. The screen colours

can be chosen, along with the display mode. An unusual feature (if not unique) is the ability to select 106 or 53 characters per row in addition to the usual 80 or 40 columns; this can be rather useful when producing wider than normal documents, although the screen display is slightly slower than usual. Other options allow control codes to be displayed, i.e. Tabs and Returns; line numbers to be displayed; and the display's interlace to be toggled on or off.

The usual search and replace, and block operations are available, which work as expected.

One of the most useful features is the ability to create multiple-file documents. This allows for a single document to be spread over several files. As the document is scanned through, each of its individual file components is automatically loaded in as and when necessary. Although this results in slight delays, it is a small price to pay for a most useful facility.

IW integrates seamlessly with Computer Concepts' spell checker, Spellmaster. Whole documents can be spell checked, or just a marked section, although the most useful action is the continuous check. As each word is typed, it is checked, and a beep emanates if it is not found in the dictionary; as the Spellmaster dictionary is located in ROM, checking is so quick that there is no apparent delay when this is enabled.

Once a document has been created, the most important part is being able to print it out. Here, IW makes life as simple as possible. A printer configuration menu allows the parameters for controlling the printer to be entered easily; certainly easier than, for example, Acorn's View printer drivers.

Unlike Wordwise, the file load and save facilities are menu driven, showing a list of files available for selection. The highlight is moved over the file name required and Return pressed to confirm. This works equally well with both DFS and ADFS filing systems.

In common with other Computer Concepts products, IW is part of its ROM-link series. This enables data to be readily exchanged with other such packages, using a simple 'language'. With this, actions such as mail-merging can be achieved.

In use, IW is extremely friendly to use, with all the facilities that are likely to be needed in domestic use. The WYSIWYG display is very useful, if you happen to like that sort of thing, and doesn't get in the way too much. It could be said that IW is a second generation word processor, having arrived after the likes of View and Wordwise, and having learnt from their earlier success.

Documentation supplied consists of a function keystrip, 108 page Reference Manual (with index), and a double-sided A4 quick reference card. The style is readable, and although there appears to be adequate information, more would have been useful. Further reading can be found in the numerous third-party books available.

View

Mike Williams describes the View family of word processors.

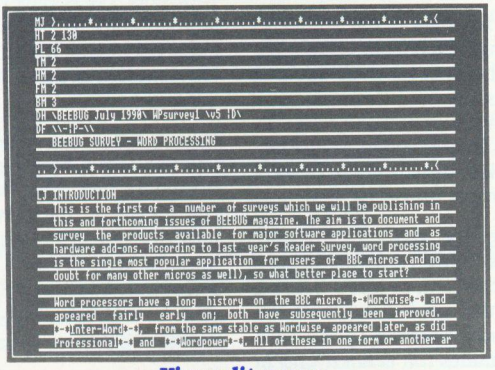
View 3.0 (Acorn Computers) £63.36
View 3.0 includes the Acorn printer drivers.
ViewIndex (Acorn Computers) £13.11
ViewSpell (Acorn Computers) £32.32
View Printer Drivers (Acorn) £9.29

I have been using Acorn's View as my principal word processor ever since it first appeared, and despite at times being forced to learn the rudiments of Wordwise, Inter-Word and the like, my preference has never wavered. The main characteristics which initially won me over were the largely WYSIWYG display and the facility to work in any mode, and given a monitor of sufficient resolution, an 80 column mode such as mode 3 has to be the best. I use a high resolution monochrome screen, which gives a clear, steady image, and cost all of £70 when new.

Unlike most word processors for the BBC micro, View does not present an initial menu - rather it relies on the user remembering a small number of commands (to load or save documents, for example), and all relevant star commands are equally available to the user. Catalogue a disc for example, and the resulting display remains on screen while you load a file. This command approach provides a highly efficient yet simple interface with the user.

Escape switches to editing mode. View is controlled with the function keys, and by combining these also with Shift and Ctrl, some 29 functions are available. User defined functions can also be added using the Shift-Ctrl combination. All expected functions are catered for, including *Delete line* not found in Wordwise. You can also mark blocks for copying or deletion, split and join lines, and set markers.

Formatting and layout are achieved in three ways. View text is controlled by a *ruler*, which specifies left and right margins, and tab settings. New rulers can be introduced at any point within a document to change the layout. *Embedded commands* sit in the extreme left-hand margin (two characters wide), to control features such as centring, page numbering etc. In addition, *highlight markers* can be set with the function keys for underlining, italics, bold and other styles, though these cannot be displayed as such on screen.



View edit screen

The use of rulers makes the creation of tables particularly easy. Simply create the columns using Tab between adjacent entries; subsequently, changing tab positions on the controlling ruler will instantly change the alignment of all entries in the associated column.

Macros are another feature of View which not only provide short cuts for commonly used phrases and paragraphs, but provide the means for creating standard letters, merging a file of user-created names and addresses to perform a mail-merge operation.

Like Inter-Word, View is part of a family which includes ViewIndex and ViewSpell. However, the dictionary for the latter is held on disc which makes it relatively slow in operation, but Computer Concepts' Spellmaster works just as well with View as with CC's own products.

Just as View is independent of screen mode, so View text is created and edited independently of any printer. The link is achieved by the use of *printer drivers*, a concept which in more sophisticated form has now been taken up by the Archimedes. Thus, for example, the same text can be output in draft mode to a cheap dot matrix printer, yet also to a high quality inkjet or laser printer by preselecting an appropriate printer driver.

In its lifetime, there have been (at least) three versions of View, 1.4, 2.1 and 3.0. View 1.4 is not now recommended, having some undesirable characteristics. Printer drivers, other than a single Epson compatible, are available as an extra item with View 2.1, but are an integral part of the View 3.0 package. For straightforward printing to an Epson compatible printer, separate printer drivers are unnecessary.

Despite at one time having a reputation for being difficult, a viewpoint I have never fully understood, View is a straightforward word processor, that lets you see what you are doing, and gives you complete control of your text without getting in your way. To my knowledge, View continues to be used by a good many professional and semi-professional writers, many of whom have migrated to more exotic machines for other work. For me, it always was and likely always will be the best word processor for any BBC micro.

ADDRESSES

Acorn Computers Ltd,
Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN.
Tel. (0223) 245200.

Computer Concepts,
Gaddesden Place, Hemel Hempstead, Herts HP2 6EX.
Tel. (0442) 63933.

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

Norwich Computer Services,
18 Mile End Road, Norwich NR4 7QY.
Tel. (0603) 507057.

Phone Call Costing

Check the cost of your phone calls with David Williams' smooth program.

What I really want is a little device that I can place under the phone, and after a call it tells me how much it has cost. Quite simple! But for some reason no-one seems to make it. It seems to me that there is a gap in the market that someone ought to fill. But while we wait, here is a program that does the job for you, even if it does tie up your computer. With the increasing cost of calls, especially if some are business calls, it could be a useful facility.

The program is written for the BBC model B in the first place. This does not have a TIME function, but an amended version is given for the Master, using its own real-time clock. The tricky bits occur when the day or time period changes, and the Master clock will have to be changed temporarily to verify that different sections of the program are working correctly.

The main problem is in the changing rates of charge at different times of the day, which also vary with the day of the week. So the two main sections of code are to check the day and time, and a look-up table (held in the 3 by 4 array sec(3,4) - see line 3360 onwards) for the number of seconds per unit for a given distance and band charge. I have only allowed for UK calls, but it could easily be extended to cope with overseas calls.

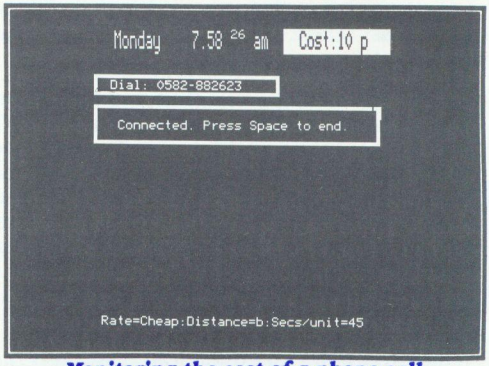
You may need to update this to cater for any price changes. The three rows of data (from line 3420) cater for the three different charging rates (cheap, standard and peak). The four figures in each row relate to the rates for local, 'a' rate (under 35 miles), 'b1' rate and 'b' rate (over 35 miles).

The program does not 'know' which numbers should relate to which rate. Numbers without STD codes (beginning with a zero) will always be treated as local. Other numbers in the local, 'a' or 'b1' rates should be included in data statements to suit your needs (see lines 2640, 2700 and 2810. All other numbers are deemed to be at the 'b' rate.

USING THE PROGRAM

Type the program in and save it. Keep to the line numbering given as this allows the amendments for the Master 128 version to be added. When run, it will first ask you to specify the day of the week, and the current time in hours and minutes. One or two letters are needed for the day (T=Tuesday, TH=Thursday, also B=Bank Holiday). Once checked the program is then ready for you to make a call.

Enter first the dial or STD code (if appropriate), then when prompted the number itself. The program then simulates a dial tone while you actually dial the number. You may also note that I have left in a PROCtone (line 3110) which I decided not to use, but left in for you to experiment with.



Monitoring the cost of a phone call

When the number replies, press the space bar, and the program will keep a running total of cost as the call proceeds. When the program is running, the current cost of the call is shown on a coloured background, green for cheap, yellow for standard rate, and red for peak rate. A muted beep sounds every time the cost increases to keep your mind on the bill! On completion, press the space bar again to see the final cost.

The process can be repeated as often as you wish. Pressing Escape will terminate the program completely.

MASTER 128 VERSION

To create a version for the Master 128 (not Compact for which use the model B version), delete (or omit) the following lines:

```
1060, 1070, 1090, 1100, 1120 to 1160,
1180, 1190, 1200, 1210, 1240, 1250, 1910,
1920, 2020 to 2090, 2170
```

Then type in, or modify, the lines in Listing 2, and save the whole program as Phone-M. The BBC B version has some lines missing in the listing, where the Master version has some extra lines.

In use, this version reads the current day and time from the in-built real-time clock, and goes straight to the call costing part of the program. Operation is otherwise the same.

PROGRAM NOTES

I wanted all users to be able to follow the principles of the program, and so it has been written in a simpler style than might be strictly necessary. For example, I felt that it was easier to follow the three repeated search sections in PROCdist, (line 2480) rather than a combined function. Likewise, the 'sieve' used in PROCrate1 (line 2390) can easily be understood when laid out in this fashion.

I have assumed the worst at a change over period, say from cheap to standard rate. BT may be generous and let a unit started be completed at the lower charge, even though the rate has increased, but I doubt it. It has even been rumoured that a call started at peak rate, continues to be charged at that price even when the call runs into a cheap period. So this program may help as a check against your itemised bill.

When you enter the dial code, and then the number, these are validated in FNcheck1 (line 1000). This is a bit tricky, as you can have up to six digits in the code, and seven in the number, but not more than ten altogether. FNcheck2 (line 1760) sees that you have not used any letters in error, e.g. O instead of 0, or I for 1. This might matter because it gave the wrong charge rate, and so the wrong cost of the call.

If you watch the 'clock' at different times in the program, you will notice that it sometimes goes up two or three seconds at a time. This is because the computer is quite busy doing other things, but it usually checks the time every second. The INKEY$ function in line 170 has a time P% which can be varied from three seconds (which is the 'ringing' cycle) to one second when in the 'charging' cycle.

If you need to change the unit charge when the cost of telephone calls go up, then line 2930 has the variable *unitcharge* (this value includes VAT).

The number of seconds per unit may also be changed in the data statements from line 3420 as previously mentioned. When entering data for code numbers, note the dummy value '000' used as an end-of-file marker (see lines 2680, 2790 and 2820).

To test that your listing has been entered accurately, you will need to test the program at the 'edges' of different zones. So for instance, on Monday at 7.59am, or 8.59am check that the rate changes on the hour. Likewise, say, does Friday at 11.59am become 12 noon? Or does Sunday at 11.59pm become Monday at midnight? And is it 'cheap rate' all day Saturday?

Changing the time or day is easy on the BBC B, but for the Master perhaps the easiest way is to enter:

```
PRINT TIME$     <Return>
```

which gives, say:

```
Mon, 13 Oct 1989.09:13:34
```

Then use the COPY key to change it by entering:

```
TIME$="Fri 13 Oct 1989.11:59:00"
```

followed by Return and then run the program.

Listing 1

```
   10 REM Program Phone-B (model B)
   20 REM Version B1.1
   30 REM Author  David Williams
   40 REM BEEBUG  July 1990
   50 REM Program subject to copyright
   60 :
  100 MODE7:ON ERROR GOTO 320
  110 PROCdef:PROCdata
  120 PROCsetclock
  130 REM **** MAIN LOOP ****
  140 REPEAT
  150 PROCnumber:oldcost=0:PROCrate
  160 REPEAT
  170 PROCclock:G$=INKEY$(P%)
  180 IF G$="" THEN PROCring
  190 UNTIL G$=" "
  200 chargestart=TIME DIV 100
  210 Q%=131:PROCborder1
  220 PRINTTAB(4,7)CHR$136"Connected. Pr
ess Space to end."CHR$137:P%=100
  230 REPEAT
  240 PROCcost:PROCclock:PROCrate
  250 UNTIL INKEY-99
  260 Q%=130:PROCborder1
  270 PRINTTAB(4,7)"Disconnected. Press
Return."CHR$137
  280 REPEAT:PROCclock:UNTIL INKEY-74
  290 UNTIL FALSE
  300 END
```

```
  310 :
  320 MODE 7
  330 IF ERR<>17 REPORT:PRINT" at line "
;ERL
  340 END
  350 :
 1000 DEF PROCsetclock:REM SET CLOCK
 1010 REPEAT
 1020 CLS:Q%=135:PROCborder1
 1030 PRINTTAB(4,7)"Day"CHR$134"(M=Mon T
U=Tues B=Bank Hol)";:INPUTday$
 1040 PROCday
 1050 FOR N=0 TO 1
 1060 PRINTTAB(0,N)CHR$148CHR$157CHR$131
CHR$141;day$
 1070 NEXT
 1080 PROCborder1
 1090 INPUTTAB(4,7)"A(m) or P(m)? "temp$
 1100 IF temp$="A" am$="am":A%=0 ELSE am
$="pm":A%=12
 1110 PROCtime:PROCborder1
 1120 INPUTTAB(4,7)"Hour? "hour$:hour=VA
L(hour$)
 1130 IF hour>12 hour=hour-12:hour$=STR$
(hour)
 1140 IF hour=12 AND am$="pm" THEN A%=0
 1150 hour=hour-1:REM MOD 12<>1-12
 1160 hour=hour+A%
 1170 PROCtime
 1180 INPUT'TAB(4,7)"Minutes? "min$
 1190 min=VAL(min$):IF min<10 min$=RIGHT
$("00"+min$,2)
 1200 PROCtime
 1210 TIME=((hour*60+min)*60+sec)*100
 1220 PRINTTAB(0,3)SPC20
 1230 PROCclock:Q%=131:PROCborder1
 1240 INPUTTAB(4,7)"Is all OK? (Y/N) "ch
eck$
 1250 UNTIL check$="Y"
 1260 ENDPROC
 1270 :
 1280 DEF PROCday:REM VALIDATE DAY
 1300 day$=LEFT$(day$,2)
 1310 day=INSTR("*MOTUWETHFRSASUBA",day$
)
 1320 day=day/2:day$=day$(day)
 1370 ENDPROC
 1380 :
 1390 DEF PROCnumber:REM NUMBER?
 1400 REPEAT:*FX15,1
 1410 SOUND1,-8,153,2:P%=260:REM Pause t
ime
 1420 Q%=134:PROCborder1
 1430 PROCborder2
 1440 code$="":number$=""
 1450 PRINTTAB(4,7)"Enter dial code (if
none Return)."
 1460 REPEAT G=GET:code$=code$+CHR$G
 1470 PRINTTAB(10,4)code$
 1480 IF G=127 code$=LEFT$(code$,LEN(cod
e$)-2)
 1490 UNTIL G=13
 1500 code$=LEFT$(code$,LEN(code$)-1)
 1510 SOUND1,-8,153,2
 1520 PRINTTAB(4,7)"Enter phone number (
and Return)."
 1530 REPEAT:G=GET
 1540 number$=number$+CHR$G
 1550 PRINTTAB(10,4)code$+"-"+number$+CH
R$148
 1560 IF G=127 number$=LEFT$(number$,LEN
(number$)-2)
 1570 UNTIL G=13
 1580 number$=LEFT$(number$,LEN(number$)
-1)
 1590 UNTIL FNcheck1 AND FNcheck2
 1600 PROCdist
 1610 PRINTTAB(4,7)"Press Space Bar when
 connected."
 1620 totalcost%=0:cost%=0
 1630 ENDPROC
 1640 :
 1650 DEF FNcheck1:REM CHECK No. OF DIGI
TS
 1660 flag=TRUE:len=LEN(number$)
 1670 IF len<1 OR len>7 THEN flag=FALSE
 1680 IF LEN(code$)>6 THEN flag=FALSE
 1690 IF len+LEN(code$)>10 flag=FALSE
 1700 IF flag=TRUE THEN =flag
 1710 Q%=131:PROCborder1
 1720 PRINTTAB(4,7)"Wrong number of digi
ts-try again."
 1730 PROCwarn:G=GET
 1740 ENDPROC
 1750 :
 1760 DEF FNcheck2:REM CHECK DIGIT ONLY
 1770 flag=TRUE
 1780 FOR N=1 TO LEN(code$+number$)
 1790 temp$=MID$(code$+number$,N,1)
 1800 temp=INSTR("0123456789",temp$)
 1810 IF temp=0 THEN flag=FALSE
 1820 NEXT
 1830 IF flag=TRUE THEN =flag
 1840 Q%=131:PROCborder1
 1850 PRINTTAB(4,7)"Not all digits-pleas
e try again."
 1860 PROCwarn:G=GET
 1870 ENDPROC
 1880 :
 1890 DEF PROCtime:REM PRINT TIME/COST
 1900 time$=hour$+"."+min$:IF time$="12.
00" THEN PROCnoon
```

```
 1910 IF hour$="1" AND am$="pm" THEN A%=
12
 1920 IF hour$="1" AND am$="am" THEN A%=
0
 1930 cost$="Cost:"+STR$(totalcost%+cost
%)+" p "
 1940 PRINTTAB(13,0)"É"time$;CHR$140;sec
$;CHR$141;am$;SPC2;
 1950 PRINTTAB(25,0)CHR$ppr%CHR$157CHR$i
nk%;cost$
 1960 PRINTTAB(13,1)"É"time$;"    ";CHR$1
41;am$;SPC2;
 1970 PRINTTAB(25,1)CHR$ppr%CHR$157CHR$i
nk%;cost$
 1980 message$="Rate="+rate$+":Distance=
"+dist$+":Secs/unit="+STR$(unitsec):len=
LENmessage$
 1990 spc%=19-len DIV2
 2000 PRINTTAB(0,22)SPC(spc%)CHR$132mess
age$SPC2
 2010 ENDPROC
 2020 :
 2030 DEF PROCnoon:REM NOON/MIDNT am/pm
 2040 IF noon=FALSE THEN ENDPROC
 2050 IF am$="am" THEN am$="pm" ELSE am$
="am"
 2060 IF am$="am" day=day+1:IF day=8 THE
N day=1 ELSE IF day=9 THEN day=8
 2070 noon=FALSE:day$=day$(day)
 2080 FOR N=0 TO 1:PRINTTAB(0,N)CHR$148C
HR$157CHR$131CHR$141;day$;SPC2:NEXT
 2090 ENDPROC
 2100 :
 2110 DEF PROCclock:REM CLOCK
 2120 sec=(TIME DIV 100)MOD 60
 2130 sec$=RIGHT$("00"+STR$(sec),2)
 2150 min=(TIME DIV 6000)MOD 60
 2160 IF oldmin<>min oldmin=min:min$=RIG
HT$("00"+STR$(min),2)
 2170 hour=1+(TIME DIV 360000)MOD 12
 2180 REM IF hour+A%>11 THEN am$="pm" EL
SE am$="am":A%=0
 2190 hour$=STR$(hour)
 2200 IF oldhour=11 AND hour=12 noon=TRU
E
 2210 oldhour=hour
 2220 IF oldsec<>sec PROCtime
 2230 oldsec=sec
 2240 ENDPROC
 2250 :
 2260 DEF PROCcost:REM COST
 2270 PROCrate:IF oldrate<>rate THEN cha
rgestart=TIME DIV 100:totalcost%=totalco
st%+cost%
 2280 chargesec=TIME DIV 100-chargestart
 2290 cost%=(-1*(chargesec MOD unitsec>0
)+(chargesec DIV unitsec))*unitcharge
 2300 IF oldcost<>cost% SOUND0,-10,0,1:o
ldcost=cost%
 2310 oldrate=rate
 2320 ENDPROC
 2330 :
 2340 DEF PROCrate:REM FIND RATE
 2350 IF day>5 rate=1:rate$="Cheap" ELSE
 PROCrate1
 2360 unitsec=sec(rate,dist)
 2370 ENDPROC
 2380 :
 2390 DEF PROCrate1:REM FIND WORKDAY RAT
E
 2400 T%=hour+A%
 2410 IF T%<25 rate=1:rate$="Cheap":ppr%
=146:ink%=129
 2420 IF T%<18 rate=2:rate$="Standard":p
pr%=147:ink%=132
 2430 IF T%<13 rate=3:rate$="Peak":ppr%=
145:ink%=134
 2440 IF T%<9  rate=2:rate$="Standard":p
pr%=147:ink%=132
 2450 IF T%<8  rate=1:rate$="Cheap":ppr%
=146:ink%=129
 2460 ENDPROC
 2470 :
 2480 DEF PROCdist:REM FIND DISTANCE
 2490 IF code$="" OR LEFT$(code$,1)<>"0"
 dist=1:dist$="L":ENDPROC
 2500 RESTORE 2650
 2510 local$="":dist=4:dist$="b"
 2520 REPEAT READ local$
 2530 IF local$=code$ dist=1:dist$="L"
 2540 UNTIL dist=1 OR local$="000"
 2550 IF dist=1 ENDPROC
 2560 REPEAT READ local$
 2570 IF local$=code$ dist=2:dist$="a"
 2580 UNTIL dist=2 OR local$="000"
 2590 REPEAT READ local$
 2600 IF local$=code$ dist=3:dist$="b1"
 2610 UNTIL dist=3 OR local$="000"
 2620 ENDPROC
 2630 :
 2640 REM  DATA LOCAL CALLS
 2650 DATA 0451,0367,036787,036781
 2660 DATA 06667,066641,036786,04515
 2670 DATA 066689,036785,03677,0666
 2680 DATA 036782,066688,04514,000
 2690 :
 2700 REM  DATA 'a' TYPE CALLS
 2710 DATA 0235,027582,078987,0454,09938
7
 2720 DATA 06723,0272,045421,0454,0993
 2730 DATA 060874,0225,0869,0789,059451
 2740 DATA 068481,038682,027588,0869
```

```
 2750 DATA 0789,059451,068481,038682
 2760 DATA 0869,0235,0386,053186,0451
 2770 DATA 0225,048838,0249,02216,060885
 2780 DATA 0380,0684,0272,0386,053181
 2790 DATA 099382,063527,027588,000
 2800 :
 2810 REM  DATA 'b1' ROUTE CALLS
 2820 DATA 0123,000
 2830 :
 2840 DEF PROCdef:REM DEFINE VARS
 2850 DIM day$(8),sec(3,4)
 2860 sec$="00":min$="":hour$=""
 2870 time$="":code$="":number$=""
 2880 noon=FALSE:Q%=134:dist=3:dist$=""
 2890 min=0:sec=0:oldsec=0:oldmin=0
 2900 oldhour=0:chargestart=0
 2910 chargesec=0:unitsec=0:totalcost%=0
 2920 cost%=0:oldrate=0:rate=1:rate$=""
 2930 ppr%=146:ink%=129:unitcharge=5.06
 2940 *FX202,32
 2950 VDU23,1,0;0;0;0;:REM CURSOR OFF
 2960 PROCdata
 2970 ENDPROC
 2980 :
 2990 DEF PROCborder1:REM BORDER 1
 3000 PRINTTAB(0,6)CHR$158CHR$148CHR$183
STRING$(35,CHR$163)CHR$235
 3010 PRINTTAB(1,7)CHR$148CHR$181CHR$Q%S
TRING$(33,CHR$32)CHR$148CHR$234
 3020 PRINTTAB(0,8)CHR$158CHR$148CHR$245
STRING$(35,CHR$240)CHR$250
 3030 ENDPROC
 3040 :
 3050 DEF PROCborder2:REM BORDER 2
 3060 PRINTTAB(0,5)CHR$158CHR$148STRING$
(24,CHR$163)
 3070 PRINTTAB(1,4)CHR$148CHR$181CHR$Q%"
Dial:"STRING$(15,CHR$32)CHR$148CHR$234
 3080 PRINTTAB(0,3)CHR$158CHR$148STRING$
(24,CHR$240)
 3090 ENDPROC
 3100 :
 3110 DEF PROCtone:REM SOUND
 3120 FOR N=1 TO 10:
 3130 SOUND 1,-1,120,10
 3140 SOUND 2,-1,92,10
 3150 SOUND 3,-1,72,10
 3160 NEXT
 3170 ENDPROC
 3180 :
 3190 DEF PROCring:REM RING TONE
 3200 *FX15,1
 3210 SOUND1,-1,121,10:SOUND 2,-1,133,10
 3220 SOUND1,0,120,2:SOUND 2,0,92,2
 3230 SOUND1,-1,121,12:SOUND 2,-1,133,12
 3240 SOUND1,0,0,10:SOUND 2,0,0,10
 3250 ENDPROC
 3260 :
 3270 DEF PROCwarn:REM WARN SOUND
 3280 FOR N=1 TO 6
 3290 SOUND1,-12,173,1
 3300 SOUND1,-12,167,2
 3310 NEXT
 3320 ENDPROC
 3330 :
 3340 DEF PROCdata:REM INPUT DATA
 3350 RESTORE 3420
 3360 FOR Y=1 TO 3:FOR Z=1 TO 4
 3370 READ sec(Y,Z)
 3380 NEXT:NEXT
 3390 FOR Z=0 TO 8:READ day$(Z):NEXT
 3400 ENDPROC
 3410 :
 3420 DATA330,96,60,45
 3430 DATA85,34.3,30,24
 3440 DATA60,25.7,22.5,18
 3450 DATA Wrong Day,Monday,Tuesday,Wedn
esday,Thursday,Friday,Saturday,Sunday,Ba
nk Hol.
```

Listing 2

```
   10 REM Program Phone-M (Master 128)
 1030 PRINTTAB(4,7)CHR$134;:INPUT"Is tod
ay a Bank Holiday? (Y/N) "temp$
 1040 IF temp$="Y" THEN day$="Bank Hol."
:clock=FALSE
 1050 PROCday:PROCclock
 1290 IF clock=TRUE THEN day$=LEFT$(TIME
$,3)
 1300 temp$=LEFT$(day$,2)
 1310 day=INSTR("*MoTuWeThFrSaSuBa",temp
$)
 1330 FOR N=0 TO 1
 1340 PRINTTAB(0,N)CHR$148CHR$157CHR$131
CHR$141;day$;SPC2
 1350 NEXT
 1360 clock=TRUE
 1900 time$=hour$+"."+min$
 2120 sec$=RIGHT$(TIME$,2)
 2130 sec=VAL(sec$)
 2140 min$=MID$(TIME$,20,2)
 2150 min=VAL(min$)
 2160 hour=VAL(MID$(TIME$,17,2))
 2180 IF hour>11 THEN am$="pm" ELSE am$=
"am"
 2190 IF oldhour<>hour THEN hour$=STR$(h
our MOD 12)
 2200 IF hour$="0" THEN hour$="12":PROCd
ay
 2400 T%=hour
 2940 clock=TRUE:*FX202,32
```

Thanks for the Memory - Bas128 (Part 1)

by Andrew Rowland

Ever fancied 64K of free memory in Basic, or wanted to use sideways RAM to store data, and didn't know how? Bas128 provides a simple answer. In this month's article I shall describe how to get started with this useful language, and present a library of procedures and supporting programs.

MORE MEMORY

Bas128 is a version of Basic II (not Basic IV) which was supplied with B+128 and all Master series computers. Unlike normal Basic, which is in ROM and uses main memory for storing programs and data, this version is run from disc and occupies main memory itself. This means that it can use the four banks of sideways RAM as its storage area, giving a total of 64K free to the user, compared with at most 28.5K of free memory in a Master, or 44K free under HiBasic with a 6502 second processor or Master Turbo. The price you pay is in speed. Because it takes longer to access sideways RAM than main memory, programs run more slowly using Bas128.

Normally, of course, accessing sideways RAM from Basic is difficult, and can only be done a byte at a time. Bas128 gives you all the convenience of being able to write longer programs, and the facility to dimension arrays to a much greater size than under ROM Basic, all without needing to learn any special techniques.

So why is Bas128 so little used? Many people were put off by a lack of information - the Master 128's Welcome Guide said nothing about the language - and so found it difficult to adapt programs to take advantage of the extra memory. Not surprisingly, it was soon put aside and forgotten.

GETTING STARTED

You will probably want to make a working disc for Bas128, so start by copying two files from your Welcome disc: *Bas128* and *BasObj* (both are in the library of the Master series' discs). Then simply type *Bas128. Your computer will announce "BBC BASIC 64K bytes free" and you're in business.

The file Bas128 on your disc is a loader for the main program. If you have a second processor it asks you to switch it off and if necessary, changes to the shadow equivalent of the current screen mode. It then runs BasObj, which occupies &3000 to &7FFF. That brings us to "don't" number one: don't type *SHADOW 1 and change to a non-shadow mode, as it would clear Bas128 from memory, with obvious unpleasant consequences! In fact the MODE command will always select a shadow mode, but it can't stop you wreaking havoc with the Ctrl key. Acorn also warn that OSHWM to &3000 may be used as workspace by Bas128, so it's best to avoid it entirely - perhaps a BEEBUG reader knows some safe areas?

If two of your RAM banks have been switched out by altering the internal links, Bas128 will adjust accordingly to give 32K free, and if none of them are available it won't start. What it doesn't do is check to see if the banks are already in use, so "don't" number two is: don't install a ROM image in sideways RAM, as it will probably get corrupted and cause the machine to hang.

EXTENDED ADDRESSING

The first sign that you have extra memory at your disposal is that PAGE announces itself as &10000 and HIMEM as &20000. These seemingly impossible addresses are a feature of Bas128's extended addressing, as I shall call it. The addresses &0000 to &FFFF are the familiar ones relating to the BBC's 64K of main memory. But Bas128 doubles this range to &1FFFF, where all addresses above &FFFF in fact refer to sideways RAM, though the user is not aware of this. &10000 refers to the first byte in bank W and &1FFFF the last one in bank Z (see table 1), but Bas128 switches between the banks of sideways RAM and translates the addresses for

you when you use the indirection operators '?', '$' and '!', DIM and even CALL. A point to remember, however, is that the extended addresses are unique to Bas128: the OS does not recognise them and they should not be confused with the pseudo addressing you can use with *SR.. commands.

Extended address	ROM I.D.	RAM bank (Master)
&10000 - &13FFF	W	4
&14000 - &17FFF	X	5
&18000 - &1BFFF	Y	6
&1C000 - &1FFFF	Z	7

Table 1. Extended addressing

As programs are now stored in sideways RAM, loading and saving them becomes as slow as using SRSAVE or SRLOAD without the Q parameter (OSGBPB is used instead of OSFILE).

PROGRAMMING IN BAS128

The joy of using Bas128 is that for the most part, it is identical to using Basic II on which it is based. Of course, care must be taken with PAGE, HIMEM, TOP and LOMEM. Master users must remember that Bas128 does not support special features of Basic IV and VI like EDIT and LIST IF. Table 2 lists the things to avoid, but next month's article will reveal ways round some of these. Some of them are easy to deal with: VDUs finished with '|' should finish with ;0;0;0;0;. ON-PROC can be re-written with ON-GOTO. Finally, *BASIC takes you back to normal Basic, so only use END.

ON-PROC	LIST IF	EDIT
TIME$	EXT#=	COLOR
VDU .. |	using CHR$141 in REMs	

Table 2. Basic IV keywords not supported by Bas128

There are some other areas where extra help is welcome, and for these I have written a library of procedures to be incorporated into your own programs (listing 1). In each case, renumber as necessary.

A problem can arise with *LOAD and *SAVE: As the disc filing system cannot access sideways RAM directly, they must be changed to the *SR... equivalents. PROCsr does the necessary conversion (line 1000 onwards). Replace a line like:

```
OSCLI("SAVE file "+STR$~(start%)+
" "+ STR$~(end%))
```

by

```
PROCsr("SAVE file",start%,end%)
```

only if the addresses concerned are above &FFFF. If the memory referred to crosses RAM banks, an error will be produced.

There is a bug in version 1.10 (but not version 1.12 provided with the Compact): INSTR seems to return FALSE under all circumstances! I therefore recommend replacing INSTR with FNinstr (line 2000 onwards). This procedure has the same syntax as the extended form of INSTR, so use, for example:

```
IF FNinstr(long$,short$,1) ...
```

For Master 128 users, there is no TIME$, so I have written an emulation (line 3000 on): use FNtime when you would read the date or time, and PROCtime to set the clock. The syntax is exactly the same as for TIME$. You must include a line like line 2910 somewhere early in your program.

CONVERTING EXISTING PROGRAMS

If your database program has just run short of memory, it is a tempting thought that Bas128 may extend its usefulness. Unfortunately, even if the program concerned is not protected, there is likely to be a lot of work involved. Some programs will run straight away; in some cases the only problem is a "PAGE=&1900" or "*BASIC" in the boot file; but in others, careful thought is required.

The first point to consider when converting another program is that there is only any point in doing so if you can use the extra memory. Arrays may have to be dimensioned to a larger size, and any range checks on subscripts altered. Secondly, if any machine code is loaded from disc to run between OSHWM and &8000, it will probably prove impossible to change. Always think through what needs to be done before spending a lot of time: is it really worth it?

FIXING BREAK

One annoyance with Bas128 is that if you press Break, you find yourself back in normal Basic. It is possible to type *KEY 10 CALL &3100|M to ensure it restarts, but this is no good when booting a disc: Bas128 is not restarted until after the boot file has been EXECed, by which time it's too late. A better answer is to trap the Break key using listing 2, which creates a utility called KEEP. This utility installs itself in page 9 and correctly starts Bas128 before the !BOOT is EXECed or run, but take care not to corrupt it or the machine will hang until switched off. You must type *KEEP after starting up Bas128, so the correct procedure is to type *Bas128, *KEEP, then boot the disc. When you want to leave Bas128 type *FX247 before *BASIC.

CONCLUSION

Bas128 offers the Basic programmer over double the memory in a very convenient form. Set against this are slower speeds and the need to alter many existing programs to take advantage of it. Next month, I shall present two useful utilities and describe how Bas128 can be used to assemble direct to sideways RAM.

Listing 1

```
   10 REM Program .>TOOLKIT
   20 REM Listing 1 for Bas128
   30 REM Version B1.00
   40 REM Author  Andrew Rowland
   50 REM BEEBUG  July 1990
   60 REM Program subject to copyright
   70 :
  100 REM *SR.. command emulation
  110 :
 1000 DEF PROCsr(A$,A%,B%)
 1010 IF FNbank(A%)<>FNbank(B%) $&900=CH
R$0+CHR$98+"Wrap"+CHR$0:CALL &900
 1020 A%=A% MOD &4000+&8000
 1030 B%=B% MOD &4000+&8000
 1040 A$="SR"+A$+" "+STR$~A%+" "+STR$~B%
+" "+STR$~FNbank(A%)
 1050 OSCLI A$
 1060 ENDPROC
 1070 :
 1080 DEF FNbank(A%)=(A%-&10000)DIV &400
0+4
 1090 :
 1900 REM INSTR replacement
 1910 :
 2000 DEF FNinstr(A$,B$,A%)
 2010 A$=MID$(A$,A%)
 2020 IFLENB$>LENA$ =FALSE
 2030 LOCAL F%,I%
 2040 FORI%=1 TO LENA$-LENB$+1:IFMID$(A$
,I%,LENB$)=B$ F%=I%:I%=LENA$+1
 2050 NEXT
 2060 IFF% THEN=F%+A%-1 ELSE=FALSE
 2070 :
 2900 REM TIME$ emulation (Master 128 on
ly)
 2910 buffer=&A00
 2920 :
 3000 DEF FNtime
 3010 LOCAL A%,X%,Y%
 3020 X%=buffer MOD &100
 3030 Y%=buffer DIV &100
 3040 A%=14:?buffer=0
 3050 CALL &FFF1
 3060 =$buffer
 3070 :
 3080 DEF PROCtime(A$)
 3090 IFASCA$=32 REPEATA$=MID$(A$,2):UNT
ILASCA$<>32
 3100 IFRIGHT$(A$,1)=" "REPEATA$=LEFT$(A
$,LENA$-1):UNTILRIGHT$(A$,1)<>" "
 3110 LOCAL A%,X%,Y%
 3120 X%=buffer MOD &100
 3130 Y%=buffer DIV &100
 3140 A%=15:?buffer=LEN(A$)
 3150 $(buffer+1)=A$
 3160 CALL &FFF1
 3170 ENDPROC
```

Listing 2

```
   10 REM Program .>KEEPbas
   20 REM Listing 2 for Bas128
   30 REM Version B1.00
   40 REM Author  Andrew Rowland
   50 REM BEEBUG  July 1990
   60 REM Program subject to copyright
   70 :
  100 osbyte=&FFF4:osasci=&FFE3
  110 wordv=&20C:Q%=&900
  120 FOR pass = 0 TO 3 STEP 3
  130 P%=Q%
  140 [OPT pass
  150 \ preserve across break
  160 LDA #247:LDX #&4C
  170 LDY #0:JSR osbyte
  180 LDA #248
  190 LDX #break MOD 256
  200 LDY #0:JSR osbyte
  210 LDA #249
  220 LDX #break DIV 256
  230 LDY #0:JSR osbyte
  240 RTS
```

Continued on page 26

1st course

A Data Input Routine

by Paul Pibworth

Given that computers are used for processing data, the most onerous task for us humans is actually getting the data into the computer in the first place. The routines described here make this process, plus the editing of data, much easier, not to say more professional in appearance. Details appear first on the screen for confirmation, which if given, allows the information to be transferred to main memory.

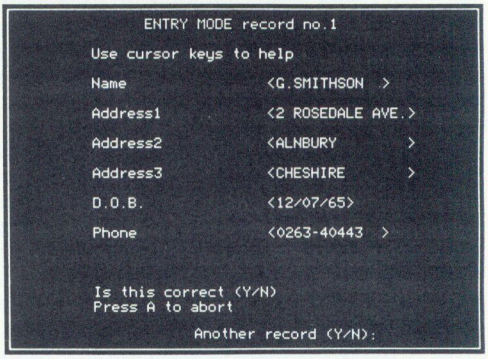
The functions and procedures listed here form a system which can then be adapted for your own use. The program actually works as it stands, to allow it to demonstrate its usefulness. In practice, however, the menu would be replaced by your own menu system.

The system works as follows. A screen is displayed which shows just field names and angled brackets as input markers. I have used mode 7 because of the greater clarity of the screen display. During input, the cursor is then restricted to these bracketed areas, and can be moved up, down, and across these *fields* by means of the cursor keys. Text can be overwritten, and deleted. Text can be restricted to given characters, e.g. numeric for date, or even certain alphabetic characters if a given response is needed.

Once the cursor has reached the bottom of the input area, confirmation is requested. You can abort, or retry, or of course, confirm data entry. The information displayed on the screen is only then stored in a corresponding array, where it can be further processed as required. The same system is used for editing data, except that the brackets are pre-filled with the data to be edited. It also seemed worthwhile to use the same arrangement to display any records for confirmation prior to deletion.

You may find it helpful to type in the program and run it at this stage before proceeding with the more detailed explanation of its workings.

If you look at the listing, you will see that there are basically three main procedures, which are fairly self explanatory. *PROCedit* and *PROCdelete* are similar, and could be rewritten as one, i.e. *PROCeddel*. This would result in a more concise program, but would make it harder to follow. It is the following nine functions and procedures which give the routine its usefulness, some of which may need to be adapted for your own use.



A typical data input screen

The basic information which determines the screen display is contained in data statements from line 1120 onwards. The first two values are the number of fields and the maximum number of records to be used. This is then followed, for each field, by a name (which acts as a prompt), a maximum length for that field, and a type. This is used for checking input, and I have used five types:

0 no specification
1 alphabetic
2 numeric
3 money
4 integer

You don't have to use these; you can change or add to them as you wish (for example a date type). All of this data is stored in three arrays called *field$()*, *length%()* and *type%()*.

The outline screen display is created from this data by *PROCscreen*, and user data is entered or edited using *PROCwrite*. Once data entered has been confirmed by the user, it is written to an array *input$()* by *PROCread*. This array is dimensioned at the outset according to the maximum number of records and number of

fields (the first number being the record number, the second the field number).

PROCread uses the advantages of the BBC friendly O.S. The cursor is passed across all the input areas on on the screen, and uses *FNread* to read the character in each position. This uses OSBYTE (an O.S. call at &FFF4) with A% set to 135 to read the screen character in the current cursor position (hence FNread moves the cursor to the required position using VDU31). I have used the string *input$* to build up each field, which is then transferred to the appropriate position in the array *input$()*. Further processing could also be included - to insert leading spaces or remove trailing spaces for example.

The key procedures (almost literally) are *PROCkey* and *PROCkey2*. The former accepts and checks all key presses, and the cursor is moved accordingly. Note that line 2140 sets Caps Lock for upper case input only - this can be omitted if required. If a printable character is input, this is passed to *PROCkey2* to validate according to the *type* specified for that field. This is where the main customisation is likely to take place. You could expand this procedure with as many types and variations as you wish.

The one example of a more complex type included in the listing is type 3 (money sums). This is used to insert a point automatically 2 digits from the end of any field of this type (see line 1870). Cursor movement and deletion then skip over the point (see lines 2160, 2170 and 2360).

Other points to note are as follows. Each data entry starts in position 21 on a line and finishes in position *20+length%(F%)* where F% is the field number (hence the occurrence of this expression at various points - see line 2200 for example). Similarly, the expression *2*F%+2* determines the correct screen line for each field (see line 1860 for example).

The point about this program is that you can use it at two levels. By specifying your own data and copying the routines as they are (except for customising *PROCkey2*) you have a ready-made data input system for your own needs. Studying the various functions and procedures in more detail should provide considerable insight into how you might write your own routines for such a task.

```
   10 REM Program Data Input
   20 REM Version B2.1
   30 REM Author  Paul Pibworth
   40 REM BEEBUG  July 1990
   50 REM Program subject to copyright
   60 :
  100 MODE 7:ON ERROR GOTO 210
  110 PROCsetup
  120 REPEAT
  130 M%=FNmenu
  140 IF M%=1 THEN PROCenter
  150 IF M%=2 THEN PROCedit
  160 IF M%=3 THEN PROCdelete
  170 UNTIL M%=4
  180 MODE7:*FX4,0
  190 END
  200 :
  210 IF ERR=17 THEN GOTO 120
  220 MODE7:REPORT:PRINT" at line ";ERL
  230 *FX4,0
  240 END
  250 :
 1000 DEF PROCsetup:LOCAL I%
 1010 READ fmax%:REM number of fields
 1020 READ rmax%:REM number of records
 1030 DIM field$(fmax%),length%(fmax%),t
ype%(fmax%)
 1040 REM Input and store field descript
ors
 1050 FOR I%=1 TO fmax%
 1060 READ field$(I%),length%(I%),type%(
I%)
 1070 NEXT
 1080 DIM input$(rmax%,fmax%)
 1090 R%=0:REM record counter
 1100 ENDPROC
 1110 :
 1120 REM No of fields, records
 1130 DATA 6,20
 1140 REM Field descriptors
 1150 DATA Input 1,6,4
 1160 DATA Input 2,1,1
 1170 DATA Input 3,3,2
 1180 DATA Input 4,2,0
 1190 DATA Input 5,8,0
 1200 DATA Input 6,6,3
 1210 :
 1220 DEF FNmenu:LOCAL M%
 1230 CLS:PRINTTAB(12,1)"MENU"
 1240 PRINTTAB(10,3)"1....Enter"
 1250 PRINTTAB(10,5)"2....Edit"
 1260 PRINTTAB(10,7)"3....Delete"
 1270 PRINTTAB(10,9)"4....Exit"
 1280 PRINTTAB(5,11)"Press a key (1 to 4
):";
 1290 REPEAT:M%=GET-48:UNTIL M%>0 AND M%
<5
 1300 =M%
 1310 ..............................
 1320 REM         ENTER RECORDS
 1330 :
```

```
 1340 DEF PROCenter:LOCAL G%,G$
 1350 IF R%>rmax%-1 THEN PRINTTAB(2,6);C
HR$134;"Maximum number of records reache
d"''CHR$133;"...Press any key":G%=GET:EN
DPROC
 1360 REPEAT
 1370 R%=R%+1
 1380 PROCscreen(0,R%)
 1390 PROCwrite
 1400 IF G$="Y" THEN PROCread(0,R%) ELSE
 R%=R%-1
 1410 PRINTTAB(10,22);CHR$132;"Another r
ecord (Y/N): "
 1420 REPEAT:G%=GET AND 95:UNTIL G%=78 O
R G%=89
 1430 UNTIL G%=78 OR R%=rmax%
 1440 ENDPROC
 1450 .................................
 1460 REM        EDIT RECORDS
 1470 :
 1480 DEF PROCedit:LOCAL G%,r%,G$
 1490 IF R%<1 PRINTTAB(8,5);CHR$134;CHR$
136;"No data...press any key":G%=GET:END
PROC
 1500 REPEAT
 1510 CLS
 1520 PRINT"Please enter the record No "
:INPUT r%
 1530 IF r%<1 OR r%>R% PRINT"No such rec
ord":G%=INKEY(200):GOTO 1600
 1540 PROCscreen(1,r%)
 1550 PROCedit2(r%)
 1560 PROCwrite
 1570 IF G$="Y" THEN PROCread(1,r%)
 1580 PRINTTAB(10,22);CHR$132;"Another r
ecord (Y/N): "
 1590 REPEAT:G%=GET AND 95:UNTIL G%=78 O
R G%=89
 1600 UNTIL G%=78
 1610 ENDPROC
 1620 .................................
 1630 REM         DELETE RECORD
 1640 :
 1650 DEF PROCdelete:LOCAL r%
 1660 CLS
 1670 IF R%<1 PRINTTAB(8,5);CHR$134;CHR$
136;"No data...press any key":G%=GET:END
PROC
 1680 PRINT"Please enter the record No "
:INPUT r%
 1690 IF r%<1 OR r%>R% PRINT"No such rec
ord":G%=INKEY(200):ENDPROC
 1700 PROCscreen(2,r%)
 1710 PROCedit2(r%)
 1720 PROCdelete2(r%)
 1730 ENDPROC
 1740 .................................
 1750 :
 1760 DEF PROCscreen(action,rec%)
 1770 REM Set out the screen for entry
 1780 LOCAL F%
```

```
 1790 CLS
 1800 PRINTTAB(5)CHR$134;
 1810 IF action=0 PRINT"ENTRY MODE recor
d no.";rec%
 1820 IF action=1 PRINT"EDIT MODE record
 no.";rec%
 1830 IF action=2 PRINT"DELETING record
no.";rec%
 1840 PRINTTAB(0,2)"Use cursor keys to h
elp"
 1850 FOR F%=1 TO fmax%
 1860 PRINTTAB(0,2*F%+2)field$(F%);TAB(2
0,2*F%+2)"<"SPC(length%(F%))">"
 1870 IF type%(F%)=3 PRINTTAB(20+length%
(F%)-2,2*F%+2)"."
 1880 NEXT F%
 1890 ENDPROC
 1900 :
 1910 DEF PROCwrite
 1920 REM Write new record to screen
 1930 VDU23,1,1;0;0;0;
 1940 *FX4,1
 1950 REPEAT
 1960 VDU31,21,4
 1970 G$=""
 1980 REPEAT
 1990 IF VPOS<4 VDU31,21,4
 2000 PROCkey((VPOS-2)/2)
 2010 UNTIL VPOS>2*fmax%+2
 2020 PRINTTAB(0,18)"Is this correct (Y/
N)"
 2030 PRINT"Press";CHR$131;"A";CHR$135"t
o abort"
 2040 REPEAT:G$=GET$:UNTIL INSTR("NnAaYy
",G$)>0
 2050 IF G$="N" OR G$="n" PRINTTAB(0,18)
SPC(21):PRINTTAB(0,19)SPC(16):VDU31,0,4
 2060 UNTIL INSTR(" AaYy",G$)>1
 2070 *FX4,0
 2080 ENDPROC
 2090 :
 2100 DEF PROCkey(F%):LOCAL K%
 2110 REM Validation single key input
 2120 VDU31,21,2*F%+2
 2130 REPEAT
 2140 *FX202,32
 2150 K%=GET
 2160 IF K%=136 AND POS>21 VDU8:IF type%
(F%)=3 AND POS=20+length%(F%)-2 VDU8
 2170 IF K%=137 VDU9:IF type%(F%)=3 AND
POS=20+length%(F%)-2 VDU9
 2180 IF K%>31 AND K%<127 THEN PROCkey2
 2190 IF K%=127 THEN PROCerase
 2200 UNTIL K%=13 OR K%=138 OR K%=139 OR
 POS>20+length%(F%)
 2210 IF K%=13 OR K%=138 OR POS>20+lengt
h%(F%) VDU10,10
 2220 IF K%=139 VDU31,21,VPOS-2
 2230 ENDPROC
 2240 :
 2250 DEF PROCkey2
```

```
 2260 REM Check and validate type
 2270 IF type%(F%)=0 THEN VDU K%
 2280 IF type%(F%)=1 AND K%>64 AND K%<12
3 THEN VDU K%
 2290 IF type%(F%)=2 AND K%>44 AND K%<58
 THEN VDU K%
 2300 IF type%(F%)=3 AND K%>47 AND K%<58
 THEN VDU K%:IF POS=20+length%(F%)-2 THE
N VDU9
 2310 IF type%(F%)=4 AND K%>47 AND K%<58
 THEN VDU K%
 2320 ENDPROC
 2330 :
 2340 DEF PROCerase
 2350 IF POS=21 ENDPROC
 2360 IF type%(F%)=3 AND POS=20+length%(
F%)-1 VDU8,127:ENDPROC
 2370 VDU127
 2380 ENDPROC
 2390 :
 2400 DEF PROCread(action,r%)
 2410 LOCAL oldR%,x,F%
 2420 REM read data from screen to array
s
 2430 IF action=1 oldR%=R%:R%=r%
 2440 FOR F%=1 TO fmax%
 2450 input$=""
 2460 FOR x=21 TO 20+length%(F%)
 2470 input$=input$+FNread(x,2*F%+2)
 2480 NEXT:input$(R%,F%)=input$
 2490 NEXT F%
```

```
 2500 IF action=1 R%=oldR%
 2510 ENDPROC
 2520 :
 2530 DEF FNread(x,y):LOCAL A%
 2540 REM Read screen character
 2550 VDU31,x,y:A%=135
 2560 C=USR(&FFF4)
 2570 C=C AND &FFFF
 2580 C=C DIV &100
 2590 =CHR$C
 2600 :
 2610 DEF PROCedit2(r%):LOCAL F%
 2620 FOR F%=1 TO fmax%
 2630 PRINTTAB(21,2*F%+2)input$(r%,F%)
 2640 NEXT F%
 2650 ENDPROC
 2660 :
 2670 DEF PROCdelete2(n%):LOCAL I%,F%,G%
 2680 REM Delete a record
 2690 PRINTTAB(5,20);CHR$130;"OK to dele
te (Y/N):"
 2700 REPEAT:G%=GET AND 95:UNTIL G%=78 O
R G%=89
 2710 IF G%=78 ENDPROC
 2720 R%=R%-1
 2730 IF R%=r%-1 ENDPROC
 2740 FOR I%=r% TO R%
 2750 FOR F%=1 TO fmax%
 2760 input$(I%,F%)=input$(I%+1,F%)
 2770 NEXTF%,I%
 2780 ENDPROC
```

Thanks for the Memory - Bas128 (continued from page 22)

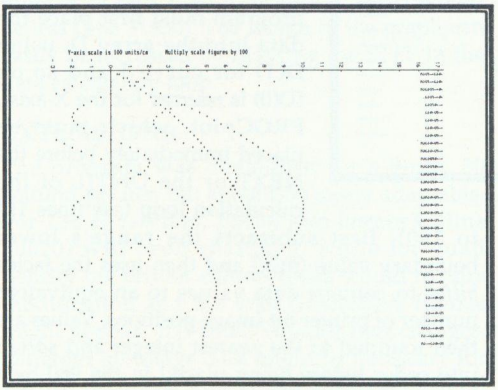
```
  250 :
  260 .break
  270 BCS vectors \ execute once
  280 RTS
  290 :
  300 .vectors \ take over WORDV
  310 PHA:LDA wordv
  320 STA oldwordv
  330 LDA wordv+1
  340 STA oldwordv+1
  350 SEI
  360 LDA #code MOD 256
  370 STA wordv
  380 LDA #code DIV 256
  390 STA wordv+1
  400 CLI:PLA:RTS
  410 :
  420 .code
  430 CMP #0:BNE out
  440 STX &70:STY &71
  450 \ reset WORDV
  460 LDA oldwordv
  470 STA wordv
  480 LDA oldwordv+1
```

```
  490 STA wordv+1
  500 \ find buffer
  510 LDY #0:LDA (&70),Y
  520 STA &72
  530 INY:LDA (&70),Y
  540 STA &73:DEY
  550 \ copy string into buffer
  560 .loop
  570 LDA string,Y:STA (&72),Y
  580 JSR osasci
  590 INY:CMP #13
  600 BNE loop
  610 CLC
  620 .out
  630 RTS
  640 :
  650 .string   EQUS "CALL &3100"
  660           EQUB 13
  670 .oldwordv EQUW 0
  680 ]NEXT
  690 a$="SAVE KEEP "+STR$~Q%+" "+STR$~P
%
  700 PRINT a$:OSCLI a$
  710 END
```

A High Resolution Graph Plotter

A.D.Clark explains how to use your printer to maximum effect to achieve high quality printouts of graphs.

I don't think that I can have been the only person to have lamented the limited vertical resolution of the Beeb (or for that matter an Archimedes etc.) in connection with the production of high quality graphs of experimental or calculated data. Also, the vertical scales of the subsequent screen dumps are seldom very satisfactory. However, with a printer capable of dual-density bit-image printing, much higher resolutions are easily obtained from the 960 discrete positions this provides. Quad-density is, of course, doubly useful. For data logging applications a continuously extending time scale is often essential.



The program *AcuPlot* listed here uses bit-image mode to provide these desirable features by using the printer paper feed to give x (or time) axis movement and the print-head traverse for y-axis movement. Rounding each point to the nearest of the 960 available lateral positions theoretically doubles the plotting accuracy achievable. In reading from the graphs one has to accept the inherent imperfections of both the printer mechanism and the measuring scale being used. However, using my trusty old Epson MX-80F/T III, AcuPlot gives excellent results in which points are plotted with a maximum error very little greater than one tenth of a millimetre.

USING THE PROGRAM

Type the program in and save it before use. The plotting routines are numbered from 30000 but for demonstration purposes the listing includes a calculation program (lines 10-190) to generate suitable data to demonstrate plotting. After entering/debugging the program, you will want to explore its features as it stands. To use the routines with your own program delete the demonstration lines and *SPOOL the lines from 30000 onwards to file as (say) AcuGraf so that they may be added to your own data calculation programs using *EXEC Acugraf. Note that the first line of your program *must* be MODE7:PROCinit (see line 100), and the last interpreted instruction PROCreset (see line 190). In the early stages of debugging, time and paper may be saved by using the printer sink command *FX5,0 (*FX5,1 reinstates). An initial Y range of -200 to 800 is suggested, with identifiers having priority and a periodicity of 2.

As the program stands it can be used in conjunction with a real-time data acquisition program only if acquired data is updated more slowly than the PROCplot cycle. With my EPSON MX80 this is just over three and a half seconds. With faster rates, your acquisition program would need a data buffer.

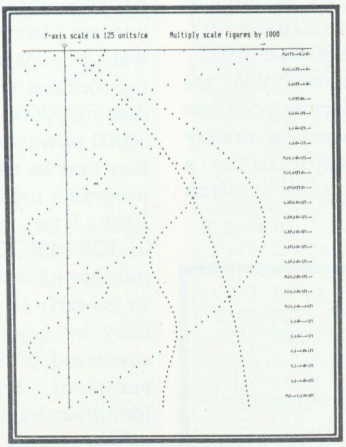
A warning: after the bit-image control code has been transmitted to the printer, the specified amount of bit-image data has to be sent before the printer will again accept control codes. On Escape, PROCreset deals with this, but if the paper runs out one must either re-load or switch off the printer to avoid spurious results when it is next called.

PROGRAM DESCRIPTION

The number of curves which may be plotted simultaneously is limited (to nine) by the need to avoid congestion and to identify the curve to which each point belongs. For this one can

choose to have printed above the points a column of single-digit curve identifiers (1 - 9) arranged in order of point magnitude (see illustrations).

The basic strategy adopted is to import all the Y-values which are to be plotted against a single X-value, to convert them to an appropriate scale (both as to magnitude and datum value) storing the resultant values in 960 bytes of a printer buffer reserved as DIM M%. If curve identifiers have been specified, the appropriate bit-image data is added at the top end of this space before dumping the 960 bytes to the printer. Control code ESC 3 is then used to advance the paper by 17/216 inch. This is almost exactly 2 mm, a very suitable length-wise interval.



Before point plotting can be commenced it is necessary to set up a suitable Y-scale. This is done in PROCdialog. You are first asked to input the identity numbers of the curves which are to be plotted. You need not specify *all* the curves for which data will be provided by your calculation or data acquisition program, but if you include a number for which data will *not* be provided, the program will detect this at the plotting stage and stop (line 31650). The X-axis can be marked at specified multiples of the basic 2 mm step. After you have input the range of (Y) values to be plotted, PROCrange selects (from 1, 2, 2.5, 4, 5 or 8cm) the most suitable interval (*yint*) for the Y-axis scale numbers and determines the necessary factor (*yfac*) be applied. If the scale range corresponding to the selected interval is unacceptable you may re-enter another set of values.

In PROCprintYscale the scale numbers are printed using PROCnumber. This is called once for every place value printed (extracted by PROCplval) as well as for minus signs. PROCchar places data from the font into the correct position in the 960 byte printer buffer. This is first zeroed by calling the machine code (assembled in PROCmcode) at *pridrv* with the vector *!vec* set to 'zeros'. Each column of characters is dumped to the printer by the same code with !vec=print. Finally, the axis itself is drawn by PROCdrawYaxis: in this the 960 byte memory is first filled with data ('4' drives printer pin 3) which provides the axis itself, and then a FOR-NEXT loop adds data ('7' drives printer pins 1, 2 and 3) for the scale marks. After setting 2mm line spacing, PROCinit returns execution to the user's program.

The user's own data calculation or data acquisition program must first place the data into the array D, using D(1) for curve 1 and so on (D(0) is *reserved* for the X-axis). PROCplot, which must be placed immediately before the NEXT or the UNTIL of the calculation loop (see lines 110 to 190), first subtracts the range's lower boundary value (min) and then uses the factor *bitfac* to convert data values to an equivalent number of printer bit-image positions. Values are then rounded to the nearest integer and sorted into order before being placed in the 960 byte buffer by PROCbldata. To plot graph points as dots, the appropriate bytes are set to 4, but if crosses have been selected (cross%=TRUE=-1) the byte is set to 14 and the adjacent bytes to 4. The curve with identifier id%(n%)=0 is the X-axis which, for a 2mm step, requires pins 1 to 6 to be fired i.e. data = 63. After every specified number of steps, a marker is created by setting the next two data-bytes to 4 (line 31950).

Finally, PROCidentify places data for the curve identifier numbers into memory from M%+idpos%, the machine code dump is called, the X value variable incremented and the paper advanced.

FINE ADJUSTMENT OF THE Y-SCALE

Line 30020 sets the printer constant, ptrcon, which is the number of bit-image positions per centimetre of print-head lateral travel. For the 80 characters per line, 10 char/inch MX80 printer, in dual density (bimpos%=960) mode, this gave 960/(8in * 2.54cm/in) = 47.24. However, the value will most probably need to be adjusted slightly so that, with your particular printer and measuring scale, Y-scale measurements are correct (mine had to be corrected to 47.1 - a matter of only one third percent). For quad density or line length greater than 8in, expect values in the range 25 to 100.

VARIATION OF THE X-SCALE

The basic x-axis step could be decreased but as the 2mm spacing is achieved as 17/216ths inch, an exact 1mm spacing must be impossible. The number of pins fired to print the X-axis would be less (line 31950). The length of the graph can easily be increased by reducing the STEP in the data calculation loop.

EXTENSIONS

The program was developed for a 9-pin, 80 column printer, but should be easily adaptable. For 136 characters, *ptrcon* and *bimpos%* (line 30020) will need changing, while for quad density the printer control codes would require modification and the scale/identifiers figures might appear too small without font changes. Different bit-image data might be necessary for 24-pin printers, and if the 216th inch feed increments are replaced by 180 then the 2mm X-axis steps may be difficult to achieve.

VARIABLE NAMES FOR YOUR PROGRAMS

The only variable names of less than four alpha-numeric characters used in AcuGraf (and thus to be avoided) are: D, D%, D(), dig, I%, id%, key%, M%, max, mem, min, N%, opt%, P% and vec.

```
10 REM Program ACUPLOT
20 REM Version B1.0
30 REM Author  A.D.Clark
40 REM BEEBUG  July 1990
```

```
  50 REM Program subject to copyright
  60 :
 100 MODE7:PROCinit
 110 FOR a=0 TO 180 STEP2.5:th=a*PI/180
 130 D(1)=1000*SIN(th)
 140 D(2)=-400*SIN(3*th)
 150 D(3)=200*COS(6*th)
 160 D(4)=500*(1+EXP(-a/60)*COS(5*th))
 170 D(5)=1000*(1-EXP(-a/75))
 180 PROCplot
 190 NEXT:PROCreset:END
 200 :
30000 DEF PROCinit
30010 error=TRUE:ON ERROR GOTO 32080
30020 ptrcon=47.1:bimpos%=960:epsncd%=76
30030 scalencm%=bimpos%/ptrcon:@%=&60A
30040 align%=5:REM aligns marks with nos
30050 maxcvs=9
30060 DIM M% (bimpos%+10),D(maxcvs),id%(
maxcvs):REM maxcvs=max no curvs
30070 CLS
30080 FOR N%=1 TO maxcvs
30090 D(N%)=-1.701E38
30100 NEXT:REM -1.701E38 = no data 'flag
'
30110 FOR N%=3 TO 4
30120 VDU31,14,N%,141,134
30130 PRINT"AcuPlot"
30140 NEXT N%
30150 PRINTTAB(14,6)"by A.D.Clark"
30160 PRINTTAB(2,15)CHR$131"Switch on pr
inter and press Space"
30170 REPEAT:UNTIL GET=32
30180 PROCfont:PROCdialog:PROCmcode
30190 PROCprintYscale:xval%=0
30200 VDU2,1,27,1,64,1,27,1,51,1,17,3
30210 ENDPROC
30220 :
30230 DEF PROCdialog
30240 REPEAT:CLS
30250 INPUTLINE"Input identifiers of cur
ves to be drawn as simple string e.g. 1,
2,3 (Return)"'input$:choise$=""
30260 FOR char=1 TO LENinput$
30270 char$=MID$(input$,char,1)
30280 IF char$>"0" AND char$<":" AND INS
TR(choise$,char$)=0 choise$=choise$+char
$+" "
30290 NEXT:CLS
30300 PRINT"Curves chosen are ";choise$;
'"O.K. (Y/N)"
30310 UNTIL FNkey("YyNn")<3
30320 idpos%=bimpos%-11*LENchoise$/2
30330 REPEAT:CLS
30340 PRINT"Curves ";choise$;"being prin
ted"
30350 INPUTTAB(0,2)"Periodicity of X-axi
s markers = "mstp
30360 UNTIL mstp>0
```

```
30370 PRINT'"Range of values to be plott
ed:"
30380 REPEAT:yfac=1:PROCrange:UNTILr%<3
30390 REPEAT
30400 PRINTTAB(0,13);SPC(40)
30410 INPUTTAB(0,13)"X axis to be drawn
at Y =   "xaxis
30420 UNTIL(xaxis>=min AND xaxis<=max)
30430 PRINT'"Are curve identifiers requi
red?"
30440 PRINTTAB(4)"If yes and with preced
ence"
30450 PRINTTAB(4)"over (high) graph poin
ts   -";CHR$131;"press P"
30460 PRINTTAB(4)"Points given precedenc
e  -";CHR$131;"press G";'TAB(4)"Identifi
ers NOT required   -";CHR$131;"press N"
30470 prec%=FNkey("PpGgNn")
30480 VDU28,0,20,39,15:CLS:VDU26
30490 IF prec%<5 REPEAT:VDU28,36,15,39,1
4:CLS:VDU26:INPUTTAB(0,15)"X-step period
icity of identifiers = "istp:UNTIL istp>
1 ELSE istp=1
30500 PRINTTAB(0,17)"Points plotted as c
rosses or dots"'SPC(13);CHR$131;"press
C   or   D":cross%=FNkey("CcDd")DIV3-1
30510 VDU11:PRINTSPC(29+11*cross%);STRIN
G$(4-3*cross%,"_");SPC(6)
30520 ENDPROC
30530 :
30540 DEF FNkey(test$):*FX21,0
30550 REPEAT
30560 key%=INSTR(test$,GET$)
30570 UNTIL key%>0:=key%
30580 :
30590 DEF PROCrange
30600 REPEAT:VDU28,0,7,39,5:CLS:VDU26
30610 INPUTTAB(15,5)"Minimum(input) = "m
in,TAB(15,6)"Maximum(input) = "max
30620 UNTIL max>min:range=max-min
30630 VDU28,27,11,39,8:CLS:VDU26
30640 IF range>10 REPEAT:range=range/10:
yfac=yfac*10:UNTIL range<10
30650 yint=scalencm%/range:subdv=1
30660 IF yint>10:REPEAT:yint=yint/10:yfa
c=yfac/10:UNTIL yint<10
30670 IF yint<2 yint=1 ELSE subdv=0.5:IF
 yint<2.5 yint=2 ELSE IF yint<4 yint =2.
5 ELSE IF yint<5 yint=4 ELSE IF yint<8 s
ubdv=0.2:yint=5 ELSE subdv=0.1:yint=8
30680 range=scalencm%*yfac/yint
30690 bimfac=scalencm%*ptrcon/range
30700 max=min+range:REM bimfac=no of BIM
 pos per unit of 'range'
30710 PRINTTAB(0,8)"Y-axis will be scale
d from  ";min,'SPC(24)"to  ";max,'"with
curve identifiers"'"(if req'd) printed f
rom Y = ";(min+((idpos%-4)/bimfac)),''"I
f OK";CHR$131;"press Y else N"
30720 r%=FNkey("YyNn")
30730 ENDPROC
30740 :
30750 DEF PROCmcode
30760 oswrch=&FFEE:DIM code 100,vec 4
30770 FOR opt%=0 TO 2 STEP 2:P%=code
30780 [OPT opt%
30790 .prndriv LDA#0:STA &70
30800 LDY #(M%+1) MOD&100
30810 LDX #(M%+1) DIV&100:STX &71
30820 .loop JMP (vec)
30830 .print LDA #1:JSR oswrch
30840 LDA (&70),Y:JSR oswrch:JMP yinc
30850 .yaxis LDA #4:STA (&70),Y:JMP yinc
30860 .zeros LDA #0:STA (&70),Y
30870 .yinc INY:BEQ xinc
30880 .cont CPY #(M%+1+bimpos%)MOD&100
30890 BNE loop
30900 CPX #(M%+1+bimpos%)DIV&100:BNEloop
30910 .exit RTS
30920 .xinc INX:STX &71:JMP cont
30930 ]:NEXT:ENDPROC
30940 :
30950 DEF PROCfont
30960 DIM font%(12,8)
30970 digit=TRUE:minus=10:blank=11
30980 FOR dig=0 TO 11:FOR I%=0 TO 8
30990 READ font%(dig,I%):NEXT:NEXT
31000 DATA6,0,9,0,9,0,9,0,6
31010 DATA2,0,2,0,2,0,2,4,2
31020 DATA15,0,8,4,2,1,0,9,6
31030 DATA6,9,1,0,2,0,1,9,6
31040 DATA2,0,15,0,10,0,6,0,2
31050 DATA6,8,1,1,14,0,8,0,15
31060 DATA6,9,9,9,6,8,8,4,2
31070 DATA4,4,0,2,2,0,1,0,7
31080 DATA6,9,0,9,6,9,0,9,6
31090 DATA4,2,1,1,6,9,8,9,6
31100 DATA0,0,0,0,15,0,0,0,0
31110 DATA0,0,0,0,0,0,0,0,0
31120 ENDPROC
31130 :
31140 DEF PROCprintYscale
31150 VDU2,1,27,1,64,1,15,1,27,1,51,1,14
,1,27,1,85,1,1
31160 PRINTSPC(6)"Y-axis scale is ";yfac
/yint" units/cm";
31170 VDU3,13,10,2
31180 PRINT SPC(6)"Multiply scale figure
s by ";yfac
31190 VDU1,10,1,18,3
31200 fmax=max/yfac:maxnum=INT(fmax)
31210 fmin=min/yfac
31220 IF INT(fmin)=fmin minnum=INT(fmin)
 ELSE minnum=INT(fmin)+1
31230 mino=minnum:maxo=maxnum:order=1
31240 IF(ABS(mino)>=10 OR ABSmaxo>=10) R
EPEAT:mino=mino/10:maxo=maxo/10:order=or
der*10:UNTIL(ABSmino<10 AND ABSmaxo<10)
```

```
31250 IF(minnum<0 AND maxnum<0)PROCnumbe
rY(minnum,maxnum,minus) ELSE IF minnum<0
 PROCnumberY(minnum,-1,minus)
31260 fstdig=TRUE
31270 REPEAT
31280 PROCnumberY(minnum,maxnum,digit)
31290 fstdig=FALSE:order=order/10
31300 UNTIL order<1
31310 VDU2,1,27,1,51,1,5,1,13,1,10,3
31320 PROCdrawYax
31330 ENDPROC
31340 :
31350 DEF PROCnumberY(start,fnsh,type)
31360 !vec=zeros:CALL prndriv
31370 FOR ypos=start TO fnsh
31380 IF type=digit PROCplval:PROCchar(d
ig) ELSE PROCchar(minus)
31390 NEXT
31400 VDU2,1,27,1,epsncd%,1,bimpos% MOD2
56,1,bimpos% DIV256:!vec=print
31410 CALLprndriv:VDU1,13,1,10,3:ENDPROC
31420 :
31430 DEF PROCplval
31440 dig=ABS((ypos MOD(order*10))DIV or
der)
31450 IF(fstdig AND dig=0 AND order>1)di
g=blank
31460 ENDPROC
31470 :
31480 DEF PROCchar(dig)
31490 D%=INT((ypos*yfac-min)*bimfac)
31500 IF D%<bimpos% mem%=M%+D%+1:FOR I%=
0 TO 8:mem%?I%=font%(dig,I%):NEXT
31510 ENDPROC
31520 :
31530 DEF PROCdrawYax
31540 !vec=yaxis:CALL prndriv
31550 FOR ypos=minnum-1 TO maxnum+1 STEP
 subdv
31560 D=(ypos*yfac-min)*bimfac:D%=D
31570 IF D-D%>0.5 D%=D%+1
31580 IF D%+algn%>0 AND D%<bimpos% M%?(D
%+algn%)=7
31590 NEXT
31600 VDU2,1,27,1,epsncd%,1,bimpos% MOD2
56,1,bimpos% DIV256
31610 !vec=print:CALL prndriv:VDU3
31620 ENDPROC
31630 :
31640 DEF PROCplot
31650 FOR N%=1 TO maxcvs
31660 id%(N%)=N%
31670 IF INSTR(choise$,STR$(N%))=FALSE D
(N%)=2*max:id%(N%)=-2 ELSE IF(xval%=0 AN
D D(N%)<-1.7E38)PRINT'"No data for curve
 number ";N%;'"(or D(";N%;") almost 'too
 big')":STOP
31680 NEXT:D(0)=xaxis:id%(0)=0
31690 FOR N%=0 TO maxcvs
31700 D(N%)=(D(N%)-min)*bimfac
31710 IF D(N%)<0 OR D(N%)>bimpos%+1 ELSE
 IF D(N%)-INT(D(N%))>0.5 D(N%)=INT(D(N%)
)+1 ELSE D(N%)=INT(D(N%))
31720 D(N%)=D(N%)+algn%
31730 NEXT
31740 !vec=zeros:CALL prndriv
31750 PROCsort:PROCbldata
31760 VDU2,1,27,1,epsncd%,1,bimpos% MOD2
56,1,bimpos% DIV256
31770 !vec=print:CALL prndriv
31780 xval%=xval%+1:VDU1,13,1,10,3
31790 ENDPROC
31800 :
31810 DEF PROCsort
31820 REPEAT:sorted=TRUE
31830 FOR N%=0 TO maxcvs-1
31840 IF D(N%)>D(N%+1) temp=D(N%+1):D(N%
+1)=D(N%):D(N%)=temp:temp=id%(N%+1):id%(
N%+1)=id%(N%):id%(N%)=temp:sorted=FALSE
ELSE sorted=sorted AND TRUE
31850 NEXT
31860 UNTIL sorted=TRUE
31870 ENDPROC
31880 :
31890 DEF PROCbldata
31900 Dhiest=idpos%
31910 IF prec%<3 ANDxval%MODistp=1 Dmax=
idpos%-1 ELSE Dmax=bimpos%
31920 FOR N%=0 TO maxcvs
31930 IF id%(N%)<1 ELSE IF D(N%)<1 OR D(
N%)>Dmax id%(N%)=-1 ELSE Dhiest=D(N%)
31940 IF id%(N%)>0 FOR offset%=cross% TO
 -cross%:M%?(D(N%)+offset%)=M%?(D(N%)+of
fset%)OR 4-10*(offset%=0 ANDcross%):NEXT
31950 IF id%(N%)=0 FOR offset%=0 TO -2*(
xval% MODmstp=0):M%?(D(N%)+offset%)=M%?(
D(N%)+offset%)OR 63+59*(offset%>0):NEXT
31960 NEXT
31970 IF xval% MODistp=1 AND Dhiest<idpo
s% PROCidentify
31980 ENDPROC
31990 :
32000 DEF PROCidentify:posn%=idpos%
32010 FOR N%=0 TO maxcvs
32020 IF id%(N%)>0 dig=id%(N%)MOD10 ELSE
 IF id%(N%)=-1 dig=blank ELSE dig=0
32030 IF dig>0 FOR I%=0 TO 8:M%?(posn%+I
%)=font%(dig,I%):NEXT:posn%=posn%+11
32040 NEXT
32050 ENDPROC
32060 :
32070 DEF PROCreset:error=FALSE
32080 VDU7,2
32090 FOR N%=1 TO bimpos%:VDU1,0:NEXT
32100 VDU1,27,1,64,1,10,3:@%=&A
32110 IF error REPORT:PRINTTAB(25,22)"at
 line ";ERL:END
32120 ENDPROC
```

Star XB24 Printer

Reviewed by Alan Wrigley

Product	**XB24-10 and XB24-15 printers**
Supplier	**Star Micronics UK Ltd, Star House, Peregrine Business Park, Goss Road, High Wycombe, Bucks. Tel: (0494) 471111**
Prices	**XB24-10 £469.95 inc.VAT XB24-15 £659.95 inc.VAT Colour kit £31.95 inc.VAT**

These are the current BEEBUG members prices

The Star XB24 is one of the new breed of printers which offer a range of facilities and a standard of printout that was unheard of a few years ago, and it should make an excellent companion for any BBC micro. It is a 24-pin machine which comes in two versions: the XB24-10 has a 10-inch carriage (80-column) while the XB24-15 is a wide-carriage version (15-inch). Functionally, however, the two versions are identical and so this review will not differentiate between them.

PRINTER DESCRIPTION

The XB24 comes with a paper guide, a ribbon cartridge and a manual. An automatic sheet feeder and a colour upgrade kit are available as optional extras. The printer is attractively designed, though a little bulkier than some others on the market. It is constructed from lightweight but durable plastic, and looks to be fairly robust, though no printer will stand up to repeated ill-treatment.

The printer has both tractor-feed and cut sheet paper handling built in. These have been ingeniously arranged so that it is not necessary to remove the fanfold paper in order to insert a single sheet. Instead, a method known as *paper parking* is employed. The fanfold paper is "parked" safely out of the way while the cut sheet is printed, then can be moved back into the paper path afterwards. Loading of single sheets is delightfully simple - the paper is slotted into the guide on top of the printer, and with one press of a button it is fed to the correct place to start printing. What is more, there is a default setting which allows you to choose where this position will be - either at the top of the paper or up to 6 lines down.

The front panel is neatly designed, and as well as having the usual on-line and feed buttons, allows you to set the pitch and print quality, and to select one of a number of built-in fonts.

As usual, there are a few DIP switches in the machine which may be altered, but many of the defaults are set from the front panel by the following method, known as the memory switch mode. Switching on the machine with certain buttons held down will cause the XB24 to print out a menu. You can use buttons to move the print head below the option required, whereupon a sub-menu will be printed, and so on. Once you get the hang of it, it is very simple to select the options required, and once selected they remain the defaults in the future.

This is proportionally spaced Times Roman
This is proportionally spaced TW-Light
This is proportionally spaced Courier
This is proportionally spaced Prestige
This is proportionally spaced Script
This is OCR-A, and this is OCR-B
This is proportionally spaced Letter-Gothic
This is proportionally spaced Blippo
THIS IS PROPORTIONALLY SPACED ORATOR
This is proportionally spaced Helvetica
This is proportionally spaced Optimo
This is proportionally spaced Cinema
This is proportionally spaced Greek

Blippo font with outline printing
ORATOR FONT WITH SHADOW PRINTING
Times Roman font in proportionally spaced superscript

Enlarged double-height bold Cinema

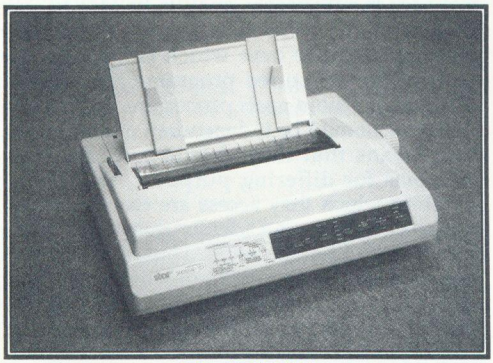
The XB24's built-in fonts

THE XB24 IN USE

The printer has two emulation modes, Epson and IBM. The manual contains comprehensive descriptions of all the control codes, although there is no quick-reference appendix - you have to plough through the book to find the one you want. There should be no difficulty using the XB24 with any BBC software which is Epson compatible, although some of the more advanced features of the printer, such as outline and shadow printing and the various fonts,

may not be easily accessible unless the software is customisable.

There are two features which in my opinion set the XB24 aside from its competitors. Firstly, it has an excellent range of fonts built in, all of which can be used in 10, 12, 15, 17 and 20 cpi pitches, as well as proportional spacing; and all the fonts can be printed in italic, bold, outline, shadow, superscript and so on. This all assumes your software can handle it of course! Samples of all these fonts are shown in Figure 1. There is even a font which prints bar codes.



The XB24-10

Secondly, in addition to the normal draft and letter quality print, the XB24 has super letter quality (SLQ). This is achieved by making two passes of the print head, which effectively simulates a 48-pin printer. The resulting quality is truly excellent, as you might expect.

Another extremely worthwhile feature is the large (27K) buffer built into the machine. This makes a tremendous difference when printing listings or long documents (an average A4 page will contain about 3K of data), and as a result the computer will be free to get on with the next task much more quickly. The buffer can be expanded to 187K, which should be enough for almost all requirements!

PERFORMANCE

The XB24 is not exceptionally fast by today's standards, but nevertheless turns in an impressive performance for a printer in its price bracket. The maximum speed for draft output is claimed to be 240 cps, and for letter quality, 80 cps. To get significantly more speed than this, you will have to pay a lot more. Using SLQ will obviously halve the latter figure, but this is compensated for by the extra quality, which is not normally available from a 24-pin printer.

The printer is not unduly noisy in use. Some 24-pin machines when printing letter quality emit a high-pitched screech which seems to come perilously close to the limits of comfort. However, my ears did not find the XB24 too uncomfortable, though I doubt if I would want to hear it all day. Some fonts are noisier than others, though. The fan operates the whole time, and this might contribute to the noise level in an otherwise quiet office.

COLOUR

Upgrading the printer for multi-colour printing is simplicity itself. The colour kit consists of a small module which just drops into a slot already waiting for it in the ribbon carriage, and a four-colour ribbon which replaces the standard black one. The XB24 then performs like any other Epson-compatible colour printer. All the facilities, including SLQ, are available in colour, provided that you have the correct software.

The printer has a short tear-off function, which allows you to advance the paper sufficiently far to tear off the form without opening up the printer case, and then reverses the paper to its previous position. If you will pardon the pun, you may come unstuck if you try to use this facility with labels, as it is never a good idea to reverse feed a sheet of labels back through the platen guard. If you are going to use labels, you can switch off the short tear-off function by using the memory switch mode described above.

CONCLUSIONS

There are many things to consider before buying a printer. Will it work correctly with *all* your software? Is the noise acceptable, bearing in mind that some people may find certain frequencies more objectionable than others? Do you actually need a 24-pin printer, or will you only be printing program listings? Provided you are satisfied that the XB24 fits the bill, then I doubt if you will find better value for money anywhere. For its price it is simply outstanding. If you are thinking of upgrading to an Archimedes, then the XB24 will be ideal for that machine too, and will handle DTP work very well.

The XB24-10, XB24-15 and colour upgrade kits are all available from BEEBUG. See our retail catalogue for details. B

Practical Assembler (Part 3)

by Bernard Hill

In the third article in this series I want to look more closely at the techniques available when actually writing an assembler program, and the methods and types of tools which are available.

So how does one start to write a large program in assembler? This was the question I asked myself when five years ago Beebugsoft asked me to go ahead with some ideas I had sent in and write the Sideways RAM Filing System which was eventually sold as ROMIT. This was the first significant program of any size which I had written in assembler as my background was in high-level languages, but I naturally carried over a great many of the methodologies from this.

Basically, there are three good methods of program design for larger systems:

1. **Top-down**
2. **Bottom-up**
3. **Increasing Functionality**

These all have their own advantages and disadvantages, and can usually be combined. *Top-down* is the most well-known, and consists of breaking the overall problem into smaller chunks, again breaking these into even smaller ones until each is a manageable size to code and test. In Basic, the best tool here is the procedure, and there is no reason why this method cannot work in assembler too. Let's use a concrete example to illustrate the technique.

Suppose we want to write a program which will take an existing disc file and write an encrypted (coded) version of it. This sort of program (together with its decrypting partner) could be the heart of a security system for transmitting confidential data down a modem or on floppy disc. We will be developing these encryption ideas in later articles as we look at the problems of interfacing to a disc filing system.

The top-down approach in Basic would be to sketch out the main program first:

```
100 PROCopenfiles
110 REPEAT
120 x%=FNreadbyte
130 y%=FNencrypt(x%)
140 PROCwritebyte(y%)
150 UNTIL endfile
160 PROCclosefiles
170 END
```

So now we can see the project as a whole and how it breaks down into more manageable sized sub-problems. In a larger program we would almost certainly break each procedure down into smaller chunks again until we feel that each is sufficiently small to be able to start writing its code. But there is a very important point to be made here which is really at the heart of the structured programming method: as far as possible each procedure or function definition should be independent of any others. This means that they should not share any variables for differing purposes, that global variables which they access are clearly defined and understood, and ideally that they can be tested as stand-alone units. Thus, when tested, they can be slipped into position in the final version with complete confidence in the knowledge that they are working correctly.

BBC Basic has a LOCAL statement, and parameters in the procedure call, to enable us to use variable names in the procedure which will automatically be independent of those in any other part of the program. We do not have this convenience in assembler, and in this respect assembler programming is like using an 'old' Basic: all the variables are nominally global, and there are no parameters as such available when a subroutine is called. In fact there is so much similarity here that I have at times used the technique to perfect a difficult algorithm in Basic (using all the bad old bits such as GOTO and GOSUB and never using REPEAT or PROC) with three main variables A%, X% and Y%. Translating this to assembler then becomes very straightforward indeed.

The converse of top-down programming is *bottom-up* programming, and I find that this too is a very useful concept in assembler; you can arrange for your top-down routines to meet bottom-up ideas somewhere near half-way! Bottom-up programming is really just another way of saying you are building a set of routines which you know you will want, carefully testing them as you proceed.

There is one good spin-off to bottom-up programming. Since you start at the bottom you can write routines which are very general-purpose, and therefore re-usable in any section of code. If you document these very carefully, you can build a library of tools for use in the future. As an example look at Listing 1. In the last article, I produced a macro EQUS FNprint(S$) which can be used to print a string. In a larger program, with lots of printing, then macros are very expensive on memory, and it is better to use a subroutine. Listing 1 is one such which prints the zero-terminated string following the call:

```
...
JSR print
EQUS "This is what I want to say":EQUB 0
LDA #0 \ etc. or whatever comes next
...
```

Listing 1

```
   10 REM Assembler Listing 1
   20 REM Version B1.0
   30 REM Author Bernard Hill
   40 REM Beebug July 1990
   50 REM Program subject to copyright
   60 :
32000 .print
32010 \ routine to print the string
32020 \ following the call; must be zero
32030 \ terminated.
32040 \ Usage:
32050 \           A,X,Y are preserved
32060 \           flags are not
32070 \ uses temporary space &70-71 as
32080 \ a pointer, and 72-3 as temporary
32090 \ saving space for A and Y
32100 \
32110 STA &72:STY &73
32120 PLA : STA &70 \get addr of string
32130 PLA : STA &71
32140 JMP printinc \increment this first
32150 .printlp LDY #0:LDA (&70),Y
32160 \ Master users can just LDA (&70)
32170 \ and omit use of Y at all
32180 BEQ printfin : JSR &FFE3
32190 \ now inc the 2-byte address of
32200 \ next character
32210 .printinc INC &70:BNE printlp
32220 INC &71 : JMP printlp
32230 .printfin LDA &71:PHA \ set up
32240 LDA &70:PHA          \ return addr
32250 LDY &73:LDA &72
32260 RTS
```

Listing 1 is also a very good example of stack handling. After the JSR instruction, the stack will contain an address one byte before the start of the string and so two PLAs and an INC are used to find this address. Moving through the string and printing it out then means that we can push the address of the next byte back onto the stack (two PHAs) and RTS back to the calling program.

The important thing when writing such a library of routines for yourself is to test them quite thoroughly and document exactly what they do. You should keep a full description with the source code (that's what comments are for), stating a few very important points:

1. **The registers or flags which are used for input (if any).**
2. **Any registers and flags which are used as output (if any).**
3. **A list of registers or flags which are preserved or overwritten.**
4. **A list of other locations needed or affected by the routine.**

See listing 1 for an example.

Keeping to a long clear list of subroutine actions such as this may seem a waste of time if you are only wanting to use the routine in one particular program, but you will find such a discipline quite invaluable when you want to re-use the code later. You will almost certainly have forgotten the details in a month's time.

Before we leave the topic of methodologies, I must clarify the third term - *Increasing Functionality*. As the term suggests, this is where we add more functions bit by bit. In terms of our encryption program, we could get the whole program working with the assembler equivalent of DEF FNencrypt(x%)=x% (i.e. no change) and completely debug the outer modules before testing the function FNencrypt and slipping it smoothly into place. Or increasing functionality could be a question of adding a further star command to a ROM you are writing. The important thing once again is to make your routines independent so that the addition of any new feature or facility does not disturb anything already present. Obviously this requires careful planning, and I believe that one of the most successful ways of writing programs which are bug-free is to combine carefully these three methods.

ASSEMBLER DEBUGGING

The most fearsome thing when first starting to write assembler programs is the complete invisibility of the thing. Many times have I written a module, and tested it in immediate mode from Basic with:

```
CALL code
```

and had to press Break to get my machine back into operation as it went into an infinite loop, scattered drivel all over the screen and printer, and probably erased half the files on the disc! In Basic you can step through the program one line at a time by typing it to the command prompt, but in assembler you can't. That's where a machine code emulator comes in very handy.

If you've never used such a beast before then they can be a bit of a revelation and are excellent for learning purposes. You can step through a section of machine code watching the registers and memory locations change, insert 'breakpoints' ("run now and stop when you get to label 'fred'") or conditional stops ("run but stop when X is 17"), and generally see a picture of machine and processor status as you single-step through the program. ROMIT contained 160,000 bytes of source code assembled into 16K bytes when completed, but I used no tools other than Beebugsoft's Exmon 2 and careful module testing.

While I can heartily recommend Exmon 2 (especially in its new Master compatible version) you don't actually need to have one to get started and do some error trapping. Judicious use of the error-handling I introduced in the last article can help you a long way down the road, and Listing 2 is an example. When this section of code is executed it simply places the contents of A, X and Y into &70, &71 and &72 and stops with an error code of 100, making error trapping from Basic very simple. In Listing 2 the macro EQUS FNstop can be placed wherever desired and the program will stop at that point announcing the values of A,X,Y. When your problem has been resolved, the line containing the EQUS FNstop can be removed, or placed elsewhere to stop the program at another place.

Next month we'll be looking at some more useful assembler routines of particular use in Basic programs and the difference between USR, CALL and CALL with parameters.

Meanwhile I would love to hear from any of you who write with questions or comments. As I'm a freelance writer, programmer and teacher and not part of the permanent BEEBUG staff then please write to me c/o BEEBUG and they will forward your letter. I would very much like to know what your problems are, maybe the solutions would be of interest to other readers. In my opinion there has never been a better machine for teaching and self-education purposes than the BBC micro, and its life in this sphere is very far indeed from exhausted.

Listing 2

```
   10 REM Assembler Listing 2
   20 REM Version B1.0
   30 REM Author Bernard Hill
   40 REM BEEBUG July 1990
   50 REM Program subject to copyright
   60 :
  100 ON ERROR GOTO 280
  110 DIM code 50
  120 FOR opt=0 TO 2 STEP 2
  130 P%=code
  140 [ OPT opt
  150 LDX #10 : LDA #0
  160 .lp TXA : SEC
  170 ADC sum:STA sum:BCC ov1
  180 INC sum+1
  190 .ov1 DEX:BPL lp
  200 EQUS FNstop \ insert this anywhere
  210 RTS
  220 .sum EQUW 0
  230 ]
  240 NEXT
  250 CALL code
  260 END
  270 :
  280 REPORT:IF ERR<>100 PRINT" at line
"ERL:END
  290 PRINT'"A="~?&70'"X="~?&71'"Y="~?&7
2
  300 END
  310 :
 1000 DEF FNstop
 1010 IF opt MOD 4>1 THEN PRINT"Stop ins
erted at address ";~P%
 1020 [OPT opt
 1030 STA &70 : STX &71 : STY &72
 1040 BRK:EQUB 100
 1050 EQUS "Program halted":BRK
 1060 ]:""
```

Programs for Education

BEEBUG Tables Invaders

Robin Murphy presents a novel approach to the task of learning multiplication tables.

How well do you, or your children, or your pupils know their multiplication tables? Improvement requires hard work and that sounds "boring". Why not try this program as a stimulus to learning, an addictive source of practice, and even a reward for other work well done.

Type in the program as listed and save it as TABLES. It will run on all versions of BBC machines, but will require an additional line:

```
0 IF PAGE>&1900:PAGE=&1200:CHAIN"TABLES"
```

for use on a machine such as a NET+DISC BBC Micro which otherwise has insufficient free memory.

When you run the program you will be asked to specify which tables you want. Simply enter a selection of the numbers from 2 to 12. Should you enter a 10 you will find that the digit 0 will be replaced by a capital letter O in order to avoid confusion with the number 8. This replacement occurs throughout the program.

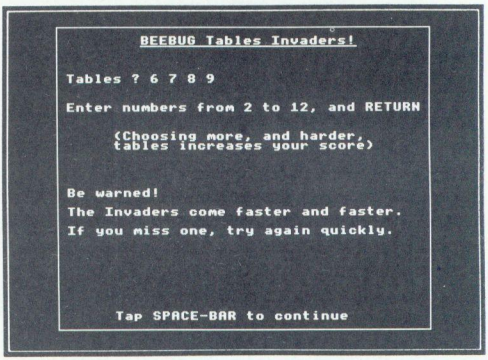
The scoring is so arranged that the more tables you choose, and the more difficult they are, the more you score. Your score is also increased when the question is more difficult and when you answer it quickly.

You will be prompted to press the space bar to start the game. This causes invaders, questions like 3x4=, to fall down the screen. You must kill them by typing the correct answer before they can land at the bottom of the screen. There is no need to press Return, merely entering the correct number of characters is sufficient to have you answer marked. Return and Delete may be used if required.

A correct answer is left flashing on the screen for a while (subliminal reinforcement?), your score increases, and another invader is released to fall more quickly than the last.

An incorrect answer is marked "x" and left on the screen until your next attempt. The invader continues to fall, but more slowly, allowing you as many tries as the time allows.

The game ends when you drive off all the invaders or one manages to land unscathed. In the latter case the correct result is given in huge writing.



Setting up the Tables Invaders challenge

To add interest to group work, a Hall of Fame is included to record the top 10 performances. When successful players enter their names they will appear as lower case letters starting with a capital letter, without any use of the Shift or Caps Lock keys.

The speed and number of invaders may be adjusted to suit the level of ability of the players. Merely alter the line:

```
1010 inv=20:max=40:min=20
```

where:

`inv`	is the number of invaders
`max`	is the maximum delay between moves (slowest speed)
`min`	is the minimum delay between moves (fastest speed)

You may also like to experiment with the weighting given to the different tables, which

will determine the frequency with which particular types of tables question appear. Those given here (which will allow scores in excess of 400) are given in the last line of the program:

```
DATA .1,.1,.2,.2,.3,.4,.3,.4,.1,.1,.3
```

for x2, x3, x4 etc. Any numbers from .1 to .9 would be sensible.

For children, the dreadful noises are important ingredients, even if they do drive adults mad! Entering *FX210,1 before running the program will quieten it down completely.

Good shooting!

PROCEDURE DETAILS

set	initialise screen and variables
reset	initialise for new game
off/on	cursor control
title	centre and underline title to new screen
play	play a single game
question	define the "invader" and its TAB position, different from last
go	input and animation control
check	process input of answer
move	move "invader" one line lower
kill	acknowledge correct answer
end	erase previously killed "invader"
wrong	acknowledge wrong answer
hall	process and show latest top 10
new	insert new entry into the top 10
name	input new name
cap/nocap	conversion to capitals or lower case
space	wait for tap of Space Bar
which	input of tables required and calculation of the Scoring Factor(SF)
down	result of "invader" landing
huge	display product in large size
blobs	print character as a 8x6 matrix of squares
x	draw large "x" (as blobs result looks very poor)

FUNCTION USED

z	convert X to a string with any zero replaced by a letter O

```
  10 REM Program Tables Invaders
  20 REM Version B1.0
  30 REM Author  Robin Murphy
  40 REM BEEBUG  July 1990
  50 REM Program subject to copyright
  60 :
 100 MODE1:@%=0
 110 ON ERROR GOTO 180
 120 PROCset
 130 REPEAT PROCreset:PROCwhich
 140 PROCplay:PROChall
 150 UNTIL FALSE
 160 END
 170 :
 180 MODE7
 190 REPORT:PRINT" at line "ERL
 200 *FX4
 210 *FX12,0
 220 END
 240 :
1000 DEF PROCset
1010 inv=20:max=40:min=20
1020 b%=2:H%=0:XN=1:a$=STRING$(20,"?")
1030 PROCoff
1040 VDU19,0,4;0;0;0
1050 VDU19,1,9;0;0;0
1060 VDU19,2,10;0;0;0
1070 VDU23,255,-1;-1;-1;-1;
1080 DIM N$(10), H%(10), X%(3), V(12)
1090 DIM B 10, T 10
1100 FOR I%=2 TO 12:READ V(I%):NEXT
1110 FOR I%=1 TO 10:N$(I%)=a$:NEXT
1120 GCOL0,3:MOVE0,0
1130 DRAW1276,0:DRAW1276,1020
1140 DRAW0,1020:DRAW0,0
1150 VDU28,1,30,38,1
1160 d$="123456789"+CHR$79
1170 X%(1)=1:X%(2)=14:X%(3)=27
1180 *FX9,10
1190 *FX10,10
1200 *FX12
1210 *FX4,1
1220 ENVELOPE 1,2,126,96,1,30,30,30,30,
-2,0,-1,120,60
1230 ENDPROC
1240 :
1250 DEF PROCreset
1260 Q$="":A$="":a$="":wait=max
1270 c%=0:down=0:N%=inv:S%=0:w%=0
1280 ENDPROC
1290 :
1300 DEF PROCoff:VDU23;10,32;0;0;0:ENDP
ROC
```

```
 1310 DEF PROCon:VDU23;10,100;0;0;0:ENDP
ROC
 1320 :
 1330 DEF PROCtitle(T$)
 1340 LOCALT%:T%=19-LEN(T$)DIV2
 1350 CLS:PRINTTAB(T%,0)T$
 1360 MOVE(T%+1)*32,1020-2*32
 1370 PLOT1,32*LENT$,0:ENDPROC
 1380 :
 1390 DEF PROCplay
 1400 PRINT"Score=0"TAB(26);
 1410 IFH%:PRINT"Hi-Score="FNz(H%(1))
 1420 REPEAT PROCquestion:PROCgo
 1430 IF a$=A$:PROCkill
 1440 UNTIL N%=0 OR down:ENDPROC
 1450 :
 1460 DEF FNz(X):$T=STR$(X)
 1470 X=INSTR($T,CHR$48):IF X:T?(X-1)=79
:GOTO1470
 1480 =$T
 1490 :
 1500 DEF PROCquestion
 1510 oldQ$=Q$:REPEAT A%=1+RND(11)
 1520 J%=RND(LENt$):IFJ%=0:J%=1
 1530 B%=ASCMID$(t$,J%)
 1540 q$=FNz(A%)+"x"+FNz(B%)
 1550 Q$=q$+"=":L%=LEN(Q$+A$)
 1560 UNTIL Q$<>oldQ$
 1570 A$=FNz(A%*B%)
 1580 B$=STRING$(LEN(Q$+A$)," ")
 1590 REPEAT X%=RND(3):UNTILX%<>XN
 1600 XN=X%:X%=X%(XN):Y%=2
 1610 PRINTTAB(X%,Y%)Q$;:Z%=POS:W%=0
 1620 T1=TIME+wait:N%=N%-1:ENDPROC
 1630 :
 1640 DEF PROCgo
 1650 a$="":b$=B$
 1660 REPEAT PROCon:D$=INKEY$(0)
 1670 IF D$>"":PROCcheck
 1680 IF TIME>T1 PROCmove:b$=B$
 1690 IF c%:IF TIME>T2:PROCend
 1700 Z%=POS:UNTIL a$=A$ OR down=TRUE
 1710 PROCoff:ENDPROC
 1720 :
 1730 DEF PROCcheck
 1740 D=ASCD$:IF D=48 OR D=111:D$=CHR$79
 1750 IF INSTR(d$,D$) AND LENa$<LENA$:a$
=a$+D$:PRINTD$;
 1760 IF D$=CHR$127 AND a$>"":PRINTD$;:a
$=LEFT$(a$,LEN(a$)-1)
 1770 IF a$>"":IF a$<>A$:IF LENa$=LENA$
OR D$=CHR$13:PROCwrong
 1780 ENDPROC
 1790 :
 1800 DEF PROCmove
 1810 Y%=Y%+1:SOUND18,-10,200-5*Y%,10
 1820 PROCoff:PRINTTAB(X%,Y%)Q$a$;TAB(X%
,Y%-1)b$;TAB(Z%,Y%);
 1830 IF Y%=29 D$=CHR$13:down=(A$<>a$)
 1840 T1=TIME+wait+W%:ENDPROC
 1850 :
 1860 DEF PROCkill
 1870 *FX15
 1880 IF c%:PROCend
 1890 IF w%:PRINTTAB(w%,z%)B$"  ";
 1900 x%=X%:y%=Y%:QA$=Q$+A$
 1910 SOUND19,1,50,70:b%=3-b%:COLOURb%
 1920 PRINTTAB(x%,y%)QA$;:COLOUR3
 1930 IF wait>min:wait=wait-2
 1940 V=(36-Y%)/5:IF W%=0:V=V*SF*(1+V(A%
))
 1950 S%=S%+V:PRINTTAB(6,0)FNz(S%)
 1960 c%=1:T2=TIME+400:ENDPROC
 1970 :
 1980 DEF PROCend
 1990 COLOUR0:PROCoff
 2000 PRINTTAB(x%,y%)QA$TAB(Z%,Y%);
 2010 COLOUR3:c%=0:ENDPROC
 2020 :
 2030 DEF PROCwrong
 2040 W%=W%+5:SOUND 0,-10,4,25
 2050 PRINT" x";
 2060 IF w%:PRINTTAB(w%,z%)B$"  ";
 2070 Z%=X%+LENQ$:VDU31,Z%,Y%
 2080 w%=X%:z%=Y%:b$="":a$=""
 2090 wait=wait+2:ENDPROC
 2100 :
 2110 DEF PROChall
 2120 IF down:PROCdown:ELSE PRINTTAB(x%,
y%)B$TAB(0,4)"You drove off all the Inva
ders."
 2130 IF S%>0AND(H%<10 OR S%>=H%(H%)):PR
OCnew:ELSE PROCspace
 2140 PROCtitle("Hall of Fame")
 2150 IF H%:FOR I%=1 TO H%:PRINTTAB(2,2*
I%+3)FNz(I%)". "FNz(H%(I%))TAB(12,VPOS)N
$(I%):NEXT
 2160 PROCspace:ENDPROC
 2170 :
 2180 DEF PROCnew
 2190 PRINTTAB(0,7)"Your score goes into
 the top ten!"'''"What is your name ? ";
:PROCname
 2200 IF H%=0:H%=1:N$(1)=a$:H%(1)=S%:END
PROC
 2210 IF S%<H%(H%):H%=H%+1:N$(H%)=a$:H%(
```

```
H%)=S%:ENDPROC
 2220 H%=H%-(H%<10):h%=H%
 2230 REPEAT H%(h%)=H%(h%-1)
 2240 N$(h%)=N$(h%-1):h%=h%-1
 2250 UNTIL S%<H%(h%-1) OR h%=1
 2260 N$(h%)=a$:H%(h%)=S%:ENDPROC
 2270 :
 2280 DEF PROCname
 2290 a$=" ":PROCon:REPEAT D$=GET$
 2300 IF RIGHT$(a$,1)=" ":PROCcap:ELSEPR
OCnocap
 2310 IF LENa$<18 AND D$>=" " AND D$<CHR
$127:a$=a$+D$:PRINTD$;
 2320 IF D$=CHR$127ANDa$>" ":PRINTD$;:a$
=LEFT$(a$,LEN(a$)-1)
 2330 IF RIGHT$(a$,2)="  ":a$=LEFT$(a$,L
EN(a$)-1):VDU8,7
 2340 UNTIL D$=CHR$13ANDa$>" A"
 2350 PROCoff:ENDPROC
 2360 :
 2370 DEF PROCspace
 2380 PRINTTAB(5,29)"Tap SPACE-BAR to co
ntinue";
 2390 REPEAT:UNTILGET=32:ENDPROC
 2400 :
 2410 DEF PROCwhich
 2420 PROCtitle("BEEBUG Tables Invaders!
")
 2430 PRINT'''"Tables ? ";:X=POS:Y=VPOS
 2440 PRINT'''"Enter numbers from 2 to 1
2, and RETURN"
 2450 PRINT'"      (Choosing more, and ha
rder,"'"      tables increases your score
)"
 2460 PRINTTAB(X,Y);:T$=" ":PROCon
 2470 REPEAT C$=GET$:R$=RIGHT$(T$,1)
 2480 IF LENT$>=25:GOTO2530
 2490 IF R$=" "AND INSTR("23456789",C$):
T$=T$+C$+" ":PRINTC$" ";
 2500 IF R$="1"AND INSTR("12",C$):T$=T$+
C$+" ":PRINTC$" ";
 2510 IF R$="1"AND INSTR("oO0",C$):T$=T$
+CHR$48+" ":VDU79,32
 2520 IF R$=" "AND C$="1":T$=T$+C$:PRINT
C$;
 2530 IF C$=CHR$127 AND T$>" ":PRINTC$;:
T$=LEFT$(T$,LEN(T$)-1):IF R$=" "ANDT$>"
":PRINTC$;:T$=LEFT$(T$,LEN(T$)-1)
 2540 UNTIL C$=CHR$13
 2550 PROCoff:T$=T$+" ":SF=1:J%=0:t$=""
 2560 FOR I%=2 TO 12:C$=" "+STR$I%+" "
 2570 IF INSTR(T$,C$):SF=SF+V(I%):t$=t$+
CHR$I%
 2580 NEXT:IF SF=1:VDU7:GOTO2420
 2590 V=RND(-TIME)
 2600 PRINTTAB(0,16)"Be warned!"'"The I
nvaders come faster and faster."'"If yo
u miss one, try again quickly."
 2610 PROCspace:CLS:ENDPROC
 2620 :
 2630 DEF PROCcap
 2640 IF D$>="a"ANDD$<="z":D$=CHR$(NOT32
ANDASCD$)
 2650 ENDPROC
 2660 :
 2670 DEF PROCnocap
 2680 IF D$>="A"ANDD$<="Z":D$=CHR$(32ORA
SCD$)
 2690 ENDPROC
 2700 :
 2710 DEF PROCdown
 2720 SOUND 0,-10,4,50
 2730 PRINTTAB(0,4)"Sorry, the Invaders
win this time."
 2740 VDU31,0,12:PROChuge(q$)
 2750 VDU31,0,20:PROChuge("="+A$)
 2760 ENDPROC
 2770 :
 2780 DEF PROChuge(T$)
 2790 $T=T$:VDU31,(38-8*LENT$)DIV2,VPOS
 2800 FOR I%=0 TO LEN(T$)-1
 2810 ?B=T?I%:IF?B=48:?B=79
 2820 IF ?B=ASC"x":PROCx:ELSEPROCblobs
 2830 VDU31,X%+8,Y%
 2840 NEXT I%:ENDPROC
 2850 :
 2860 DEF PROCblobs
 2870 A%=10:X%=B MOD256:Y%=B DIV256
 2880 CALL&FFF1:X%=POS:Y%=VPOS
 2890 FOR J%=1 TO 8:A%=B?J%
 2900 FOR K%=1 TO 6
 2910 IF A%AND64:VDU255:ELSEVDU32
 2920 A%=A%+A%:NEXT K%
 2930 VDU31,X%,Y%+J%
 2940 NEXT J%:ENDPROC
 2950 :
 2960 DEF PROCx
 2970 X%=POS:Y%=VPOS:X=112:Y=24
 2980 MOVE X%*32+64,(26-Y%)*32
 2990 PLOT0,Y,-Y:PLOT81,X,X
 3000 PLOT0,-Y,Y:PLOT81,-X,-X
 3010 PLOT0,0,X-Y:PLOT0,Y,Y
 3020 PLOT81,X,-X:PLOT0,-Y,-Y
 3030 PLOT81,-X,X:ENDPROC
 3040 :
 3050 DATA .1,.1,.2,.2,.3,.4,.3,.4,.1,.1
,.3
```

512 Forum

by Robin Burton

There's a slight change of plan this month (what's new?) brought about by something I came across since writing the last Forum - we'll get back to DOS commands next month (hopefully). It concerns a tip about running *Autoroute*, a package I've had numerous queries about over the last couple of years.

Before that, a word or two more seems called for on running MS-DOS in the 512 as a follow up to the Forum of a couple of months ago.

MS-DOS REVISITED

I've had quite a bit of feedback about the article in the May issue (BEEBUG Vol.9 No.1) from members who've tried the trick of attempting to 'warm-boot' MS-DOS on the 512. I suppose I should really have expected it, but this item seems to have elicited more of an immediate response from some of you than almost anything else I've written about in the Forum. In consequence I think a few of the 'facts of life' about the 512 wouldn't be out of place.

I was mildly surprised at the strength of response to this article and I deduce the reason is the fact that some of you obviously think MS-DOS is the 'Holy Grail' so far as DOS usage concerned. I, on the contrary, as a programmer more than a user, think that DOS Plus with its CP/M BDOS is a far more flexible and capable system and one that puts MS and PC-DOS in the shade. I must say I don't miss either of them one bit.

To those who are not aware of the facts let me restate something I haven't said in the Forum since its early days. The applications software incompatibility of the 512 and PC programs is very much more a direct result of the machine's configuration than the fact that it employs DOS Plus. Even if the 512 was supplied with MS-DOS version 3 or later (suitably customised), the vast majority of applications which fail under DOS Plus in the 512 now would continue to fail under MS-DOS in the 512 because of the hardware.

Obviously it's not possible to put accurately firm values on this sort of statement, but ever one to stick my neck out, I'm going to try to give you some idea. My feeling is that if the 512 with DOS Plus runs, for the sake of argument say, 75% of all software that runs in PCs under MS or PC-DOS, then these operating systems running in the 512 instead of DOS Plus might increase the number to 80% if you were very lucky, but probably not. The actual numbers don't matter and these examples might be wildly inaccurate, but I do believe the relative proportions are realistic, if not perhaps very scientific.

Anyway, back to the MS-DOS article. Your results inevitably have been rather mixed, with almost certainly more failure than success overall. Now while this may be disappointing, no-one should find it surprising; it is after all something that the designers of neither DOS Plus nor MS-DOS could have expected anyone to try. In other words if you tried it and had no luck you are in the majority and it's probably not because you made a mistake.

On the other hand, contrary to a suggestion from someone who had hacked about in DOS Plus on his own account quite a bit prior to my article and who said he didn't think it could ever work, I do know that some of you at least had some success. In fact even amongst those who failed, some have had reasonable error messages, although it's also true some just had silent 'hang-ups'.

Let me reinforce what I said in the May Forum. Sometimes this trick might work, but often it won't. The point is that the chance of success depends not only on which version (i.e. number) of DOS you try to use, but also

precisely which machine it was originally supplied for. This is not difficult to understand, those with access to a variety of PCs and their software will already know that not all PC compatibles are the same as each other. In the April issue of BEEBUG (Vol.8 No.10) I wrote about the development of DOS, so if you read the part about the purpose of the BIOS again everything should make more sense. PCs can be quite different to each other internally and are certainly very different to the 512.

In simple terms, you would no more expect to transplant the version of PC-DOS from say, an IBM PC to a clone and expect it to work than from any PC to the 512. If such an experiment does work you should consider yourself fortunate - if it doesn't you should be neither surprised nor suspicious of the technique.

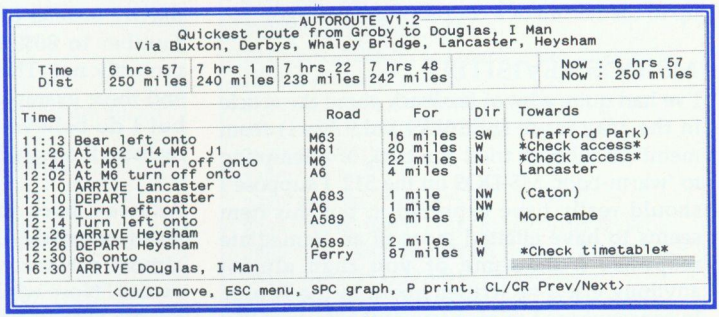
AUTOROUTE

This is a well known package from NextBase which is unique so far as I'm aware, and it's rather popular. For anyone who doesn't know the package (is there anyone?) here's a brief explanation of what it does.

Autoroute is a route finder for road journeys. I presume that there are versions for our Dutch, French and German readers and other countries too, but I've only seen the U.K. version. However, if there are other versions they'll all work in the same way with the same facilities, just using different map data which is supplied in separate files.

In defining your intended journey you are permitted to set various preferences for things like the type of roads you want to travel on (e.g. motorways, main roads, minor roads and so on) and the average speed you expect to maintain on each type of road. Having done this you then tell the program where you wish to travel from and to, with the option of travelling via several specified places on the way, with or without breaks in your journey of variable duration at each of these places.

When you have informed the package about the details of your intended journey, it then works out a number of possible routes you might use based on the information you have supplied. In some cases, particularly for very short trips, it may only come up with one suggested route, but for longer journeys it might offer half a dozen or so, showing the cheapest route, the quickest, the shortest and so on. Each of these routes can then be viewed in tabular form showing time, distance and road directions for each stage of the journey.



AUTOROUTE V1.2
Quickest route from Groby to Douglas, I Man
Via Buxton, Derbys, Whaley Bridge, Lancaster, Heysham

Time	6 hrs 57	7 hrs 1 m	7 hrs 22	7 hrs 48	Now : 6 hrs 57
Dist	250 miles	240 miles	238 miles	242 miles	Now : 250 miles

Time		Road	For	Dir	Towards
11:13	Bear left onto	M63	16 miles	SW	(Trafford Park)
11:26	At M62 J14 M61 J1	M61	20 miles	W	*Check access*
11:44	At M61 turn off onto	M6	22 miles	N	*Check access*
12:02	At M6 turn off onto	A6	4 miles	N	Lancaster
12:10	ARRIVE Lancaster				
12:10	DEPART Lancaster	A683	1 mile	NW	(Caton)
12:12	Turn left onto	A6	1 mile	NW	
12:14	Turn left onto	A589	6 miles	W	Morecambe
12:26	ARRIVE Heysham				
12:26	DEPART Heysham	A589	2 miles	W	
12:30	Go onto	Ferry	87 miles	W	*Check timetable*
16:30	ARRIVE Douglas, I Man				

<CU/CD move, ESC menu, SPC graph, P print, CL/CR Prev/Next>

Sample output from Autoroute

Routes can also be displayed in the form of a map, with the ability to show more or less detail for road and place names, to zoom in to a larger scale or out again to see more of the entire country. This facility seems to be, perhaps not surprisingly, one of the main attractions of the package. Unfortunately it also seems to be the area where some 512 users run into trouble. Even if you have used this package and didn't think there was a problem read on, You might find that you've avoided trouble simply by chance.

THE PROBLEMS

The first point is that the package is memory hungry, as it stores all the routes it finds for each journey in memory, until there's room for no more (if it finds enough). If you have a long journey when several routes might be suggested, lack of memory is the first restriction

to using the program. In this case you'll also find that, before it gets as far as crashing, the program will tell you that there isn't enough RAM to display the map.

Assuming this isn't your problem, the set-up configuration might be, and there are two aspects and two problems with this. The package should be configured for a CGA display, and it can be run either in two colour or four colour mode, but for finer map detail I find the two colour option better. On the screen type configuration menu however, it's not very clear which is the currently selected option, so let me tell you it's the fourth one down in the window. Having done this you move on to the second part of the configuration, the control method (mouse type, keyboard etc.).

This menu isn't clear either, for the same reason as the screen type selection, and again the selected option is the fourth one down in the window, but that's not really the the problem. It turns out that, regardless of what you select, including 'No Mouse' the package crashes on the map display as I'd been told. I'd had a copy on floppy disc some time ago from Chris Hughes, a Forum reader from Wakefield who had tried unsuccessfully to use the map and had asked me to see if I could do any better. The answer was no, so I can confirm that originally it failed to work in the 512, no matter how it was set up.

THE CURE

Recently I heard a tip from a 512 user that he'd had similar problems, but found that Autoroute would work correctly so long as he ran it from his hard disc and so long as he had loaded GEM 3 before using it. He said he didn't need to use an application in GEM, the fix worked even if he left GEM as soon as the desktop appeared on the screen.

After hearing this I decided to investigate again. Sure enough, the same copy of the software that previously failed ran perfectly well if it was copied to the hard disc and if GEM 3 was loaded first. So far so good, but it didn't make any sense to me. I wanted to know what really made the difference. I didn't believe it was GEM or the hard disc, so using trial and error and a process of elimination I soon found that in fact neither of these two items is relevant, it's only the mouse driver (which is needed to use GEM 3) that dictates whether the Autoroute map works or not, nothing else.

It turns out that, at least with this version of the software, an MS compatible mouse driver must be installed when you run Autoroute. Oddly, this applies even if you have set the package's configuration to 'No Mouse'. Without a mouse driver the map display always crashes halfway through drawing the screen, even if you do have enough RAM. With a mouse driver installed the entire package works quite happily, even from floppy discs (it needs two). As part of my trial and error, I proved that not only was GEM 3 irrelevant, but unfortunately so is GEM 2 and its supplied version of the mouse driver (contained in ACORNBW.SYS or ACORNCOL.SYS on the 512 GEM issue discs).

If you're one of those users who has successfully run Autoroute v1.2, including the map, you probably have the T.C.S. mouse driver and load it by default in your AUTOEXEC.BAT file. If you didn't realise it, this is why the package has always worked so far as you're concerned.

For those of you who've had problems with Autoroute version 1.2, you now know the solution. Assuming you have sufficient RAM you must install an MS-type mouse driver before the map can work. So far as I know there's only one suitable program and that's the one from T.C.S. (see Forum in BEEBUG Vol.8 No.7 for their address and phone number).

One other point, this package can be seen with the same version number but with different file sizes. I think this relates to different updates to the maps, it doesn't seem to affect anything else. B

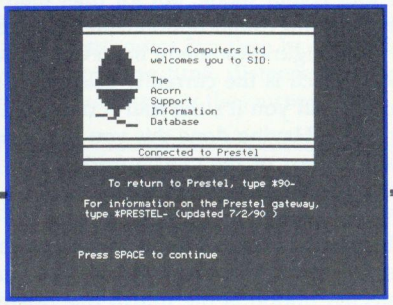
RISC USER

The Archimedes Magazine & Support Group

Risc User is enjoying the largest circulation of any magazine devoted solely to the Archimedes range of computers. Now in its third year of publication, it provides support to schools, colleges, universities, industry, government establishments and private individuals.

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

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



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

WIMP APPLICATIONS MADE EASY
This program demonstrates how a ready-made Wimp application shell can be used for the quick and simple creation of single-window applications.

INTRODUCING C
A wide ranging new series on C, a major programming language for the Archimedes.

STEPPED RENUMBER
A utility for renumbering Basic programs, which can be used with the RISC User Basic editor.

DTP RESOURCES
A look into some recent software releases which provide fonts or artwork for use with DTP packages.

A HANDY MOUSE SPEED CONTROLLER
A handy utility which allows you to select different mouse pointer speeds by clicking on the icon bar icon.

MAKING THE MOST OF SID
Exploring Acorn's support Information Database.

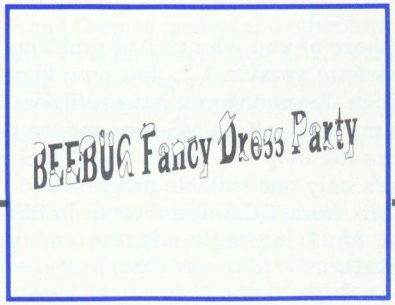
ASSEMBLER WORKSHOP
A major series for the more advanced ARM processor programmer. The latest one explains how to write applications in ARM code.

CD ROM FOR THE ARCHIMEDES
A preview of the latest innovation for the Archimedes.

MASTERING THE WIMP
A major series for beginners to the Wimp programming environment. The most recent installment is dedicated to Wimp menu systems.

INTO THE ARC
A regular series for beginners. The latest article introduces the reader to different printer types.

FILE ENCRYPTION UTILITY
A utility which allows you to code files for protection and decode them for further use.



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

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

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

SUBSCRIPTION DETAILS

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

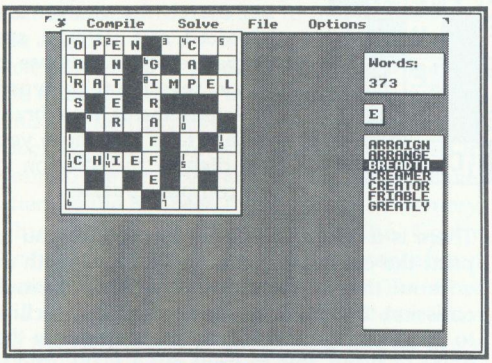
RISC User, 117 Hatfield Road, St Albans, Herts AL1 4JS, Telephone (0727) 40303, FAX (0727) 60263

Crossword

Alan Wrigley looks at a new crossword compiler and solver from Panda Discs.

Product	**Crossword**
Supplier	**Panda Discs** **Four Seasons, Tinkers Lane,** **Brewood, Stafford ST19 9DE.**
Price	**£12.95 inclusive**

Panda is best known for its Ample music discs, some of which were reviewed in BEEBUG recently (see Vol.8 No.10). However, the latest item of "bear-ware" (the term is Panda's, not mine!) offers a crossword designer package which should appeal to all crossword lovers, and also to teachers seeking classroom projects.



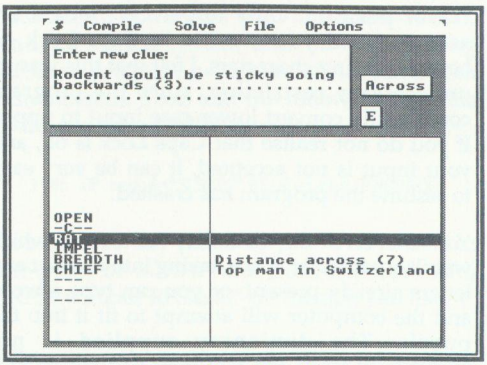
Using the built-in dictionary to insert words into a skeleton crossword

The package consists of two discs and a slim manual. One disc is the system disc, containing the programs and some example crosswords. The second is the dictionary disc which as supplied, holds a general dictionary of almost 6000 words. A range of thematic dictionaries is being produced, which will be available separately in due course. These will cover specific subjects likely to be useful in a classroom, such as travel and money. The program will work on all BBC micros, but is not really suitable for the model B unless sideways RAM is fitted. This is because the program uses sideways RAM to handle the screen displays, and if it is not present, windows are left on the screen after use instead of being closed, which can eventually be confusing.

USING THE PROGRAM

My first impression of the package was very favourable. It has been designed to emulate a Wimp environment, and this has been achieved in a very professional manner. On booting up the disc, a title screen appears, followed by a full-screen "window" with a menu bar at the top. Selecting any of the options, using cursor and Return keys, displays a pull-down sub-menu. Although this makes the response to keypresses a little slow, the whole arrangement has a nice feel to it.

The main menu offers four options: *Compile*, *Solve*, *File* and *Options*. The last of these allows you to set various defaults, such as screen colours, input device (you can use a joystick if you have one), and also the drives where the program resources are to be found. If you have a dual-sided 80-track drive, for example, you may place your dictionaries on drive 2 and avoid the need to keep separate discs. The other options will be explained in their context.

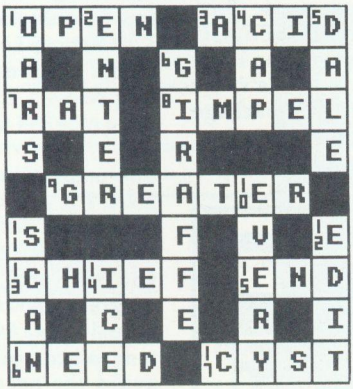


Compiling the clues

The program works on one puzzle at a time, and saves the work to disc each time you exit from an option. Once a puzzle file has been opened, all menu options will then operate on that file. You can, however, change the puzzle you are working on by using the *File* option from the main menu.

DESIGNING A CROSSWORD

To start designing a crossword, you select *Compile*, and then choose to create a framework, insert words, or compose clues. The frame can be up to 15 x 15 squares, and once it is displayed on screen you can move around inserting or deleting blank squares wherever you like. Exiting from the option will save the skeleton crossword, but if you are editing the frame of a puzzle which has already had words or clues compiled, these will be lost as the old puzzle is overwritten on the disc. I did not feel this was a good idea, particularly since no warning is given. In some circumstances you may wish to change a space in a completed crossword to or from a blank without altering the rest, and it would be very easy to lose several hours work without realising it.



Once the skeleton is prepared you may then enter the words. The program is very flexible, allowing you to enter whole words or just individual letters at the cursor position. Only upper case letters are accepted and you must have Caps Lock on before entering characters. I felt that this was an unnecessary restriction, since the program could easily convert lower case input to upper. If you do not realise that Caps Lock is off, and your input is not accepted, it can be very easy to assume the program has crashed.

You can search the dictionary for a word which will fit a particular space, taking into account any letters already present, or you can type a word and the computer will attempt to fit it into the puzzle. The dictionary supplied is not particularly large, but you may add words to it by selecting *Options* from the main menu. Alternatively, of course, you could make use of a spelling checker such as Spellmaster, which has a crossword search facility, but unfortunately there is no provision to use star commands from within Crossword. Nevertheless, compiling a crossword with this package is fun, and no doubt a good deal faster and easier than by hand.

The final element of the design is the clues. You can enter a line of text for any word which will then be displayed as the clue when solving the puzzle. It would have been nice to be able to edit clues, but sadly if you discover a mistake later you must type the entire clue again.

THE SOLUTION

Solving the puzzle involves inserting words in a similar way to the compilation process. Owing to limitations of screen size, only one clue can be displayed at any one time. You can insert complete words or individual letters, and when you need help you can request that all the letters in a word which do not match the answer are blanked out. If you are desperate you can choose to have the correct word inserted, and if your brain cannot take any more you can display the solution to the complete puzzle.

There is a dump facility which enables you to print the crossword onto paper, either with or without the words. However, a most serious omission to my mind is the lack of any facility to print out the clues. I simply do not see the point of being able to print out a blank grid if the clues cannot accompany it. After all, you are not going to want to solve your own puzzle, and if you wish to distribute it to friends or for a class to solve, each will then have to have a copy of the disc. After discussions with Panda, however, I understand that this facility should be included, and hopefully will be by the time you read this review.

CONCLUSIONS

My various criticisms should be taken as constructive ones, since overall I was quite impressed with this package. If you enjoy compiling crosswords, as I do, you will have hours of fun using it, and apart from the specific shortcomings detailed in this review, it is a delight to use. I was so impressed that I used it to compile the BEEBUG crossword on Page 54 of this issue. B

Using the ROM Filing System (Part 3)

Jon Keates adds a useful delete function to round off the RFS utilities published previously.

After the completion of the two articles *Using the ROM Filing System*, published in BEEBUG Vol.8 Nos.8 & 9 (which should be referred to for more information on the ROM Filing System), it seemed that the only filing system command now lacking in the RFS was a *Delete* routine, to remove an unwanted file from an RFS image. This, with the help of the RFSload program, would then allow full editing of an RFS image, and make the ROM Filing System as versatile as any other Filing System.

At first sight this seems an easy thing to do; all that is needed is a routine to find the selected file and then move every file, after the one to be deleted, forwards in memory, overwriting the file to be deleted. But by doing this the existing End of File (EOF) address in the RFS header block becomes incorrect and must be altered to point to the new EOF address the file will have, after the file has been moved down in memory. This in turn will mean that the Cyclic Redundancy Check data (CRC) in every altered header block will also be incorrect. Therefore the new EOF and recalculated CRC data must be written in place of the existing data for every file that is moved.

With these requirements in mind the most obvious approach is to use a machine code routine which will "Page In" the RAM bank containing the RFS image, and then work directly on it. After locating the file to be deleted we then need a simple and efficient routine that can be repeated until all the remaining files have been moved down in memory.

USING THE PROGRAM

This month's program RFSdel can be used in two different ways, as a separate utility just to delete named files, or, the relevant procedures can be appended to the RFSload program (see BEEBUG Vol.8 No.9) to produce a complete RFS editing program. Both versions can be useful at different times, although when incorporated with RFSload the resulting program will only run with PAGE set to &E00. If PAGE is at &1900 (as it will be by default on a model B with Acorn's DFS) the values set for HIMEM and the variable *fload%* (line 110) must be reset to a higher value, and this will then reduce the file buffer area.

To use RFSdel as a stand-alone delete routine, just type in the listing as printed, omitting lines with "REM with RFSload" appended to them. When this version of the program is used, the program will initially prompt for the RAM bank number, and then the filename to be deleted. No checking is done to ensure valid filenames or correct RAM banks, but the program will always fail safe.

For use in conjunction with RFSload, append the two procedures PROCerase and PROCdelcode, all lines with line numbers 3000 and upwards, to the end of the existing RFSload program (which must not have been renumbered). Then add the following new line to the procedure PROCoscom:

```
1945 IF requ$="SCRAP" PROCerase:ENDPROC
```

You can use any name of your own choice in the place of SCRAP and omit the lines with "REM without RFSload" on them, if you like.

When the new version of the RFSLoad program is run the delete routine can be called by using *SCRAP from the "Any more Files to load .." prompt. This in turn will be trapped by the new line in PROCoscom and control passes to PROCerase. The procedure then prompts for the filename to be deleted. If the named file is found in the current RAM bank, it will be

overwritten by the following files. No confirmation is requested so use with care.

If the message " .. RFS NOT found!" is returned by calling the delete routine and the RAM bank does contain a valid RFS image, then terminate the program, press Ctrl-Break and run the program again. This error is caused by the Operating System not recognizing the RFS, and not responding to OSBYTE 143.

TECHNICAL INFORMATION

The main part of the program is the code produced by PROCdelcode. This uses OSBYTE 143, the paged ROM service request, with the X register set to 13, for the initialise RFS service call, and the Y register set to 15 minus the RAM bank number. This call will return the start address of data in the RFS bank. By using the EOF address in the file header block to skip the file data, the image is searched until the file is found or the End of RFS marker is reached. If the file is not found then &FF is returned in A%.

When the file is found, the start address of the named file is then subtracted from the start address of the following file, to calculate the gap, all the subsequent files are moved down in memory by this amount.

The main loop of the code starts with the first header block of any file, marked by the sync byte &2A. As the Header Block data is moved down the length of the gap is subtracted from the existing EOF address and the new EOF is then inserted. This is followed by a call to crc% to recalculate the CRC data for the Header Block, which is then written in. Following this, another loop is used to move each full block of data that starts with the byte &23, until the sync byte for the last Header Block of a file is found. The program then jumps to the beginning of the main loop, to handle the last Header Block and the remaining partially full block of data. The whole process is then repeated until the last EOF address points to the End of RFS marker &2B.

Note: The augmented version of RFSload (called RFSEdit) is included on the magazine disc in its entirety, together with the program RFSDel listed here.

```
   10 REM  Program RFSdelete
   20 REM  Version B3.03
   30 REM  Author  Jon Keates
   40 REM  BEEBUG  July 1990
   50 REM  Program subject to copyright
   55 :
   60 REM To use with RFSload, append
   70 REM PROC's erase and delcode add
   80 REM 1945 IF requ$="SCRAP" PROCeras
e:ENDPROC
   90 :
  100 MODE 7
  110 DIM fload% &180,fname% 20
  120 FOR a%=1 TO 2
  130 PRINT TAB(6,a%)CHR$141;CHR$134;"RF
S DELETE & COMPACT"
  140 NEXT a%
  150 INPUT TAB(2,9)"DELETE File from RA
M bank  ? "ramblk%
  160 VDU 28,0,22,39,12
  170 crc%=&900:PROCchek
  180 PROCerase
  190 END
  200 :
 3000 DEF PROCerase
 3010 PROCdelcode
 3020 PRINT''SPC5"DELETE file   ?  ..."
 3030 PRINT'SPC5;CHR$133".. ";
 3040 INPUT"  "file$:REM without RFSload
 3050 REM file$=FNverchr(valid$,10):REM
with RFSload
 3060 $fname%=file$:PRINT''CHR$129;SPC5;
 3070 act%=USR(fload%) AND &FF
 3080 IF act%=&FF PRINT"... File NOT fou
nd !"
 3090 IF act%=&EE PRINT"... RFS NOT foun
d !"
 3100 IF act%=&DD PRINT CHR$136"... BAD
RFS !"
 3110 IF act%<>0 VDU 7,26:ENDPROC
 3120 PRINT file$;" is DELETED"
 3130 srload%=(Lpt?1*256+?Lpt)-1
 3140 PRINT'SPC6"End of RFS now at &";~s
rload%
 3150 VDU26:REM PROCwait(3):REM with RFS
load
 3160 ENDPROC
 3170 :
 3180 DEF PROCdelcode
 3190 Hpt=&70:Lpt=&72:gap=&74
 3200 stpreg=&76:resi=&77:Lstor=&8B
 3210 FOR pas=0 TO 1
 3220 P%=fload%
```

```
3230  [ OPT pas*2
3240 LDX #&D:LDY #(ramblk% EOR &F)
3250 LDA #&8F:JSR &FFF4
3260 LDA &F6:STA Hpt:LDA &F7:STA Hpt+1
3270 CPX #0:BEQ pagein:LDA #&EE:RTS
3280 .pagein LDA &FE30:STA resi:SEI
3290 LDA #ramblk%:STA &F4:STA &FE30
3300 CLI:JMP enter
3310 .comp LDY #0
3320 .loop JSR get:CMP #0:BEQ found
3330 CMP fname%,Y:BNE notf:INY:JMP loop
3340 .fspc INY
3350 .found LDA fname%,Y
3360 CMP #32:BEQ fspc:CMP #13:BNE endn
3370 JSR nextf:JMP delit
3380 .notf JSR get:CMP #0:BNE notf
3390 .endn JSR nextf
3400 .enter
3410 LDA Hpt:STA Lpt
3420 LDA Hpt+1:STA Lpt+1
3430 JSR get:CMP #&2A:BEQ comp
3440 CMP #&2B:BEQ fail:JMP bad
3450 .fail
3460 JSR exit:LDA #&FF:RTS
3470 :
3480 .nextf
3490 CLC:LDA Hpt:ADC #13:STA Hpt
3500 LDA Hpt+1:ADC #&0:STA Hpt+1
3510 JSR get:PHA:JSR get
3520 STA Hpt+1:PLA:STA Hpt:RTS
3530 .get LDA (Hpt)
3540 INC Hpt:BNE bret:INC Hpt+1
3550 .bret RTS
3560 .put STA (Lpt)
3570 INC Lpt:BNE sret:INC Lpt+1
3580 .sret RTS
3590 :
3600 .delit JSR get:JSR put
3610 CMP #&2A:BEQ mark
3620 CMP #&2B:BNE err:JMP exit
3630 .err JMP bad
3640 .mark
3650 SEC:LDA Hpt:SBC Lpt:STA gap
3660 LDA Hpt+1:SBC Lpt+1:STA gap+1
```

```
3670 .storbase
3680 LDA Lpt:STA Lstor:LDA Lpt+1
3690 STA Lstor+1:LDX #&0
3700 .cloop INX:JSR get:JSR put
3710 CMP #0:BEQ endl:JMP cloop
3720 .endl LDY #&FF
3730 .mloop INY:JSR get:JSR put
3740 CPY #10:BNE subm:STA stpreg
3750 .subm CPY #12:BNE mloop
3760 JSR get:SEC:SBC gap:PHP:JSR put
3770 JSR get:PLP:SBC gap+1:JSR put
3780 JSR get:JSR put:JSR get:JSR put
3790 JSR get:JSR get
3800 CLC:TXA:ADC #17:STA &8F:JSR crc%
3810 LDA &8E:JSR put:LDA &8D:JSR put
3820 .premv LDY #0
3830 .move JSR get:JSR put
3840 INY:CPY stpreg:BNE move
3850 JSR get:JSR put:JSR get:JSR put
3860 JSR get:JSR put:CMP #&23:BEQ premv
3870 CMP #&2A:BNE lastb:JMP storbase
3880 .lastb CMP #&2B:BEQ exit
3890 :
3900 .bad JSR exit:LDA #&DD:RTS
3910 BRK:EQUS ".. BAD RFS !":BRK:
3920 .exit SEI:LDA resi
3930 STA &F4:STA &FE30:CLI:LDA #0:RTS
3940 ]:NEXT:ENDPROC
9000 :
9010 DEF PROCchek
9020 FOR pas=0 TO 1
9030 P%=crc%
9040 [OPT pas*2
9050 LDY #0:STY &8E:STY &8D:CLC
9060 .loop1
9070 LDA &8E:EOR (&8B),Y:STA &8E:LDX #8
9080 .loop2
9090 LDA &8E:ROL A:BCC skip
9100 LDA &8E:EOR #8:STA &8E
9110 LDA &8D:EOR #16:STA &8D
9120 .skip
9130 ROL &8D:ROL &8E:DEX
9140 BNE loop2:INY:CPY &8F:BNE loop1
9150 RTS:]:NEXT:ENDPROC
```

B

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

PRINTING CHARACTERS 128-255 (BEEBUG VOL.8 NO.9)

Unfortunately the program as listed does not call *FX20 from PROCdefineB, as stated in the text. To rectify this, insert the following line:

```
1365 OSCLI"FX20,"+STR$((last%-159)DIV32)
```

This modification is for the model B only and Master owners need take no action.

WEAVING PATTERNS (BEEBUG VOL.8 NO.10)

The very last line of this program was omitted. It should read:

```
1810 =(Q$="Q")
```

This allows the program to terminate in response to 'Q' for quit when prompted. Thanks to Max Lievers for this.

B

BEEBUG Education

by Mark Sealey

When the new term begins - in some eight to nine weeks time - for many educationalists the National Curriculum will at last be here for real. At least in the core subjects of Mathematics, Science and English as well as Technology. What is more, we now have both Statutory and Non-Statutory guidance to suggest how best to approach it all.

This month BEEBUG Education examines how this will affect teachers at both Primary and Secondary level as far as the requirements for Information Technology are concerned.

Earlier BEEBUG Education articles argued that there are shortcomings in the way the National Curriculum has been introduced. These criticisms - and the chronic lack of funds to implement it from the computer/IT point of view - will not be repeated again.

Instead, what follows is a checklist of how teachers with experience of using Acorn equipment may want to make sure that no area of pupil's experience is ignored. Little mention is made of specific software titles: it will be found more productive to assess the packages you already know and use against the requirements outlined here. Indeed, IT co-ordinators or Heads of Department might like to draw up lists of their own and compare them with the implied list that follows.

LANGUAGE

Central to all activities in any classroom are the four modes of communicating verbally: talking (and listening), writing (and reading). The English document is quite specific about the implications for IT. With reference [non-statutory guidance, para 4.10] to Attainment Target (AT) 1, Speaking and Listening, the importance of discussion around the computer - with pupils perhaps engaged on adventure games - is seen as crucial.

The same activity is also cited [8.12] as a valuable reading development tool, though clearly it will be necessary for teachers to use as rich a selection of reading material on the computer as possible, provided reading is tackled with the emphasis on inquiry, not drill and practice.

Since language is about communication, the sending and receiving of messages - perhaps via Electronic Mail - has a large part to play. Here is an immediate hardware implication: equip yourself with a modem and access to an on-line system of some kind, Prestel or Campus 2000, for instance.

Where writing is concerned, a word processor and desktop publisher of some kind - preferably with mailmerge and a built-in spelling checker (and perhaps a style-checker and thesaurus) must now be seen as essential. Once more, it is the idea of having an audience that counts. Since presentation to this end is important, making sure that there is adequate provision for hardcopy is as crucial as ever. Maybe this will mean more printers.

On a different tack, those programs such as the recently released *Knowledge Organiser* (Clares, for the A3000 and Archimedes range) which assist pupils in the logging and direct manipulation of thought and ideas have a special place. The English non-statutory guidance again [paragraph 12.7] suggests that some writing should be the basis for thought. This may alter, too, the way word processors are used in class. It is to be hoped that they will be more frequently seen as tools to draft, develop and compose writing in preference to just the means of knocking out fair-copies.

MATHEMATICS

A key area in maths that will be new for some is that of data-handling (ATs 12, 13 and 14). A whole range of software is relevant to this area of work. These include graph packages, capable of displaying block-graphs, Carroll Diagrams, line and bar-line graphs, scattergrams, frequency tables, and pie-charts. Packages capable of best-fit (as well as mean, median, mode and range of frequency) calculations will have the edge too. Maybe one comprehensive suite that handles all aspects of data manipulation and display will fit the bill.

Data, once entered into a database, must be capable of interrogation and manipulation; so a full-feature and easy to use database (not just Teletext, say) is a must. Since it is likely that staff from curriculum areas other than IT will integrate their work with such an activity, it will be wise to

choose a user-friendly database. Decide now where *Quest* and *dBase* come on the continuum that has *Grass* and *Multistore* at its most accessible end!

Sadly - as there is little or no evidence that Logo enhances understanding of either algebra or geometry - it has been included in ATs 7 and 11; now all teachers working with children and mathematics will have to be able to teach Logo competently, covering the basic turtle graphics primitives and their incorporation into procedures (including recursive ones).

SCIENCE

The transmission and receipt of data via electronic media again figures in the Science curriculum. There is a specific Attainment Target (12) covering the use of IT, where it does not already impinge on other scientific work. Here it can easily be overlooked that the Programmes of Study are concerned not only with text but also the handling by computers of numbers, graphics and sound.

For Level 4 of this AT, every child will have to have access to sensing devices and the software to detect and measure environmental change. Actual control doesn't appear until Level 7, though Level 6 is concerned with the difference between analogue and digital representation. There are sufficient input-output devices for the 8-bit range of BBC machines to make this eminently possible, though, once more, it will be necessary to choose software with which non-specialists can become familiar, or to equip later machines with appropriate A-D and I/O expansion cards etc.

TECHNOLOGY

The biggest impact of the new regime will be in obliging teachers of children of all ages to use IT in its technological role, equipment in its own right (rather than as a tool: an art package, for example).

AT 5 specifically addresses the uses to which some quite sophisticated hard and software is put. You should make sure that your institution has the machinery, for example, to "use computer-generated pictures, symbols, words and phrases to communicate meaning" [Level 2]: mouse as well as keyboard-controlled selection of icons is meant here; and the touchscreen or much under-used Concept Keyboard too.

Databases crop up from Level 2 onwards - again they must be fully interactive. Data-logging and control are to be taught from Level 6

Software permitting the entry and running of sequences of commands, Logo-like, are mandatory from Level 4.

Pupils must select and use software for its ability to handle and present information in different ways according to the form it takes (words, graphics, figures etc); a good DTP package is the obvious workhorse here, though not the only one: some of the newer integrated data-handling software only realistically implementable on the 32-bit range - like Genesis (Software Solutions) - must be considered.

Pupils are expected to be familiar with the idea that a piece of software can simulate a real-life situation and can handle the data and variables which have been input effectively to problem-solve and predict something that would otherwise be impractical or impossible in real life.

This may well be a humble spreadsheet - or (more difficult to plan for because not at all content-free) a simulation whose subject matter relates only to the topic being worked on - say breeding patterns, a budget or humidity control. It is expected that pupils will be taught methods not only of interpreting the data processed by this and similar packages but also learn methods of checking plausibility and reliability of results.

GENERAL

Although it is unlikely that IT will figure prominently in the remaining curricula to be published, it ought to have a place in Music and Art in the obvious ways. Anticipating just how great a place would not be wise. What cannot be ignored, though, is that all pupils are expected to treat electronic storage and interrogation thereof as a natural study skill.

This really does imply at the least discs (not cassettes) and full-time access to a printer as well as a variety of input devices other than the keyboard. Clearly, the greater the integration of these media into all areas of pupils' experience the more successful the learning.

This has implications for staff-awareness and training as well as resourcing from this September onwards. The case has been made in earlier columns of BEEBUG Education that Acorn users are particularly (even singularly) well equipped to cope with these implications. All that remains to be said for next term is: "Good Luck"!

Permanently Coloured Text

by John Lasruk

I prefer to do my word processing and Basic editing in mode 3 with cyan or green text on a black background, and with interlace turned off. The trouble is, every mode change on my Beeb resets the colours to the default of white on black and a Ctrl-Break will turn interlace back on again. Furthermore, some annoying ROMs insist on changing modes (thus turning off text colour) to give up their *Help messages.

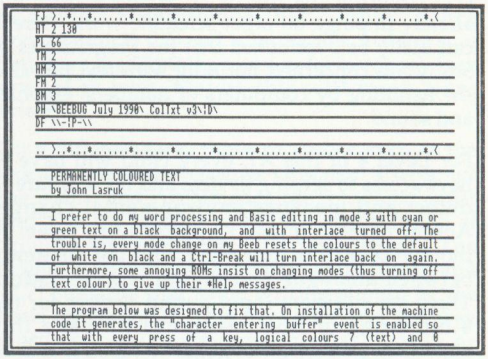
The program below was designed to fix that. On installation of the machine code it generates, the "character entering buffer" event is enabled so that with every press of a key, logical colours 7 (text) and 0 (background) are reset to whatever actual colours you prefer. If you change modes or modify the colours with a VDU19 command, the next key press will bring them back (except in mode 7 of course). If you press Break, the event is re-enabled and the machine code equivalent of *TV0,1 is called.

Just type the program in and then run it. The program asks you which colours you would prefer to use for the text and background, and where you want to have the code installed (use hexadecimal here, without the ampersand, "&", which is provided). The default location is &380, for reasons I'll outline later.

The object code is then *SAVEd, using the convention "XYtext", where "X" represents the first letter of the colour chosen for text and "Y" represents the background colour. Thus, blue text on a white background would be saved as "BWtext". Since blue and black start with the same letters, I was forced to use another letter, "K", for black. Cyan letters on a black background would be saved as "CKtext".

Once the code is saved, you can thereafter install it by entering (for instance) *GKtext or *RUN GKtext. To be called as a *star command, the file must, of course, be in the library directory if it has been set, otherwise in the $ directory. While the logical colours are amended immediately on installation of the code, interlace will not be turned off until the next mode change, unless it is off already.

Since the utility uses the keyboard event, you can disable it with *FX13,2, but Break will re-enable it. You can disable the Break intercept by entering *FX247,0. *FX14,2 will re-enable the event and *FX247,76 will re-enable the Break intercept.



Word processing with coloured text and background

There is one important thing to beware of: whatever you do, DON'T OVERWRITE THE CODE WHILE BREAK INTERCEPT IS ON! The only way to recover from the resulting system crash is to turn off your computer! It is easy to forget that comms packages and cassette use can overwrite pages &9 and &A. Defining the soft keys will overwrite page &B and defining characters 224-255 will overwrite page &C. Games programs may cram themselves into every spare nook and cranny, leaving NO safe place to store something valuable.

If you are a disc user, there is a reasonably safe place to store the code: &380, the cassette filing system workspace. The program accepts &380 as the default location for assembly. While the code actually overlaps the keyboard input buffer, the bit that overlaps is used only on

entry. In case you were wondering, that is why the execution address of the utility is placed toward the end. The rest of it should remain secure, though some games use even this area. But then, why would you use this utility with a game?

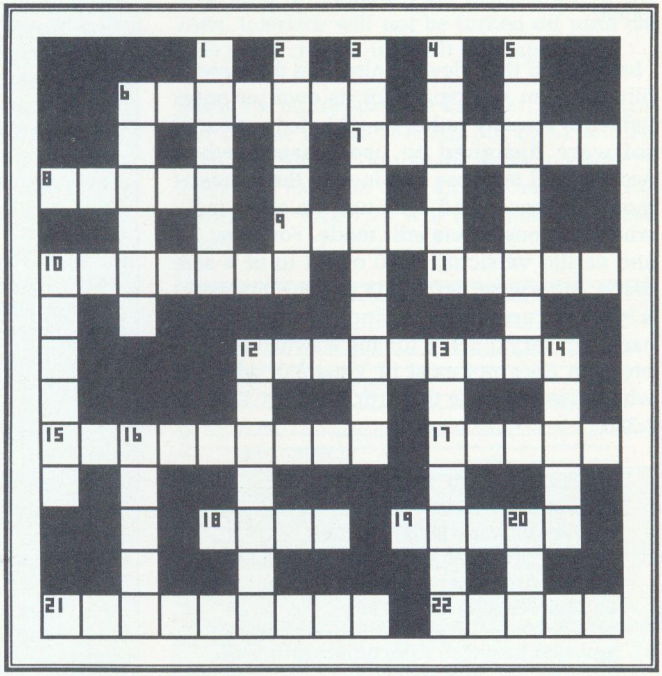
I have found that View 2.1 also uses the cassette filing system workspace for its own purposes (which is actually rather naughty for a piece of software designed to use cassette when necessary.) I suppose this is why the "cassette motor" solenoid and LED turn on annoyingly when first one enters edit mode. For View 2.1 and earlier versions, &900 ought to be a safe place, otherwise, with this routine placed at &380, you are liable to find your computer hangs up very quickly, forcing a switch off. The problem does not exist in View 3.0, which is what I used to write this article with *CKtext at &380.

```
   10 REM COLOURED TEXT KEEPER
   20 REM Version B2.2
   30 REM Author  John Lasruk
   40 REM Survives mode changes & BREAKs
   50 REM Disc users may install at &380
   60 REM BEEBUG  July 1990
   70 REM Program subject to copyright
   80 :
  100 choice$= "(R G Y B M C W)  RETURN
= black"
  110 MODE7: FORX=1TO2: VDU131,157,132,1
41
  120 PRINT SPC(6);"Coloured Text Keeper
"
  130 NEXT
  140 PRINTTAB(0,5);CHR$131;
  150 PRINT"Change text to which colour?
"
  160 PRINTCHR$131;choice$
  170 K$=GET$: real%=FNwhich
  180 U%=FNcaps: KEY$=CHR$U%
  190 IF real%=0: KEY$="K"
  200 PRINT'CHR$130;"Change background t
o which colour?"
  210 PRINTCHR$130;choice$
  220 K$=GET$: real2%=FNwhich
  230 U%=FNcaps: KEY2$=CHR$U%
  240 IF real2%=0: KEY2$="K"
  250 VDU10,134: INPUT "Assemble at? (RE
TURN = &380) &"assem$
  260 IF assem$="": assem$="380"
  270 P$="&"+assem$: C%=EVAL P$
  280 VDU21: FORX=1TO2: PROCassemble: NE
XT: VDU6
  290 end$= STR$ ~P%: jump$= STR$ ~setup
  300 OSCLI "SAVE "+KEY$+KEY2$+"text "+a
ssem$+" "+end$+" "+jump$
  310 MODE4: PRINT"Works in modes 0 - 6"
''
  320 CALL setup: OSCLI"FX138,0,0": END
  330 :
 1000 DEF PROCassemble
 1010 P%=C%
 1020 [
 1030 OPT0
 1040 .keyevent
 1050 BCCgo
 1060 LDX#0: LDY#1: LDA#144: JSR&FFF4
 1070 CLD: LDX#2: LDA#14: JSR&FFF4
 1080 LDA# begin DIV256: STA &221
 1090 LDA# begin MOD256: STA &220
 1100 .begin
 1110 PHP: PHA: TXA: PHA: TYA: PHA
 1120 LDA #real%: STA forecols+1: LDA#&C
 1130 LDX# forecols MOD256: LDY# forecol
s DIV256
 1140 JSR&FFF1: LDA #real2%: STA backcol
s+1
 1150 LDA#&C: LDX# backcols MOD256
 1160 LDY# backcols DIV256: JSR&FFF1
 1170 .out
 1180 PLA: TAY: PLA: TAX: PLA: PLP
 1190 .go: RTS
 1200 .forecols
 1210 EQUB 7: EQUD real%
 1220 .backcols
 1230 EQUB 0: EQUD real2%
 1240 .setup
 1250 LDA#247: LDY#0: LDX# &4C: JSR&FFF4
 1260 LDA#248: LDY#0: LDX# keyevent MOD2
56: JSR&FFF4
 1270 LDA#249: LDY#0: LDX# keyevent DIV2
56: JSR&FFF4
 1280 SEC: JMP keyevent
 1290 ]
 1300 ENDPROC
 1310 :
 1320 DEF FNwhich
 1330 =(INSTR("xRrGgYyBbMmCcWw",K$))DIV2
 1340 :
 1350 DEF FNcaps
 1360 =((ASC K$) OR 32) EOR32
```

BEEBUG Prize Crossword

Clues Across

6. Programmed a ROM set for the Master (7)
7. Looked after elderly person (7)
8. Move up to powerful computer language on the ARM (5)
9. Corrupted ROM contains silicon chip computer (5)
10. Appears to combine between two points (7)
11. Source of power provided by Poles after I am back (5)
12. Wealth of material about people who supply information (9)
15. Another roast mule provides solutions to incompatibility between micros (9)
17. Suitable place to slip a disc or have a stroke (5)
18. Look! A word processor! (4)
19. It's perfectly legal to have a cover after five (5)
21. Broken cup I raise to universal text format (4,5)
22. Voucher is acceptable in the heartless north (5)



Clues Down

1. Pair of insects provide the company we all need (6)
2. Cooks upset mates from the south (6)
3. Funny man reserves description that does NOT apply to issues of BEEBUG! (5,5)
4. ...and in non-volatile memory it's a matter of chance (6)
5. Actress without direction goes around endlessly looking at arms cache (8)
6. Sent to you every month like an old warrior (6)
10. Ice cream within an alien network (6)
12. Sells engineers, not heads! (7)
13. Shining in the aura Di anticipated (7)
14. Ultimate state for souls and files alike (5)
16. It's a case of keeping on top (5)
20. Printers use it in thousands (3)

***Panda Discs** has kindly agreed to donate a copy of "Crossword" as a prize for the first correct solution to be drawn from the hat.*

Please address your entries to

Crossword Competition, BEEBUG, 117 Hatfield Road, St Albans AL1 4JS.

All entries must be received by Friday July 20th.

Automating Assembler Input

Julian Potter outlines a handy utility which makes the typing of assembler code much easier.

When programming in assembler, all the commands are in upper case, but most programmers use lower case variables for labels etc. This means excessive use of the Caps Lock key (or Shift).

Once installed, the program listed here will automatically switch the machine between upper and lower case as necessary. For example, the following line of code may be entered without using Caps Lock (or Shift):

```
LDX#7:.loop DEX:LDA(old),X:STA(new),X
```

USING THE UTILITY

The program should be typed in and saved. When run it assembles a machine code routine, with a prompt for you to save this with a filename of your choice. The utility can be subsequently installed by typing *<filename> or: *RUN <filename>.

In use, the routine will automatically switch to lower case when '(', '#', or '.' are pressed. It will switch back to upper case when ':' '&' ',' or Return are pressed, while Space will toggle between them.

It will auto-enable when '[' is pressed and disable when ']' is pressed, thus the routine will only operate when typing assembler and not when typing any Basic parts to the program. Pressing f9 will exit from the routine completely.

There is one point to remember: when using ROL, ROR, ASL or LSR with the accumulator do not leave a space, for example use ROLA not ROL A.

PROGRAM NOTES

The routine is interrupt driven, and when entered looks at the value held in &EC. This holds the internal key number+128 of the last key pressed. This number is compared with the numbers of the keys we are interested in, and if it is one of our keys, the relevant section of code is called.

The sections shiftlow and shiftup are called if the character is '#', '&' or '(' to check if Shift is pressed at the same time as the key. If it is, then control is passed to the relevant section.

```
  10 REM Program Keys
  20 REM Version B1.1
  30 REM Author  J.Potter
  40 REM BEEBUG  May 1990
  50 REM Program subject to copyright
  60 :
 100 ONERROR GOTO150
 110 PROCassemble
 120 PROCsave
 130 END
 140 :
 150 MODE7:IF ERR=17 END
 160 REPORT:PRINT" at line ";ERL
 170 END
 180 :
1000 DEF PROCassemble
1010 FOR pass=0 TO 3 STEP 3
1020 P%=&900:start=P%
1030 [OPT pass
1040 LDA&EC
1050 CMP#247:BNEover:JMPclosedown:.over
1060 CMP#216:BEQnomore
1070 CMP#184:BEQbackon
1080 LDX&71:CPX#0:BEQon:RTS:.on
1090 CMP#226:BEQswap
1100 STA&72
1110 CMP#231:BEQlower
1120 CMP#200:BEQupper
1130 CMP#201:BEQupper
1140 CMP#149:BEQshiftlow
1150 CMP#145:BEQshiftlow
1160 CMP#230:BEQupper
1170 CMP#180:BEQshiftup
1180 .out RTS
1190 .lower:LDA#202:LDX#247:LDY#0
1200 JSR&FFF4:LDA#0:STA&70:RTS
1210 .upper:LDA#202:LDX#111:LDY#0
1220 JSR&FFF4:LDA#1:STA&70:RTS
1230 .shiftlow:LDA&70:CMP#0:BEQout
1240 LDA#&81:LDX#&FF:LDY#&FF:JSR&FFF4
1250 TYA:BNElower:JMPout
1260 .shiftup:LDA&70:CMP#1:BEQout
1270 LDA#&81:LDX#&FF:LDY#&FF:JSR&FFF4
1280 TYA:BNEupper:JMPout
1290 .nomore:LDX#1:STX&71:RTS
1300 .backon:LDX#0:STX&71:RTS
1310 .swap:LDX&72:CPX#226:BEQout
1320 LDX#226:STX&72:LDA&70
1330 CMP#1:BEQlower:JMPupper
1340 .initial:LDA#1:STA&71:STA&72
1350 LDA#0:STA&220:LDA#9:STA&221
1360 LDA#14:LDX#3:JSR&FFF4:RTS
1370 .closedown:LDA#13:LDX#3:JSR&FFF4
1380 LDA#&A6:STA&220
1390 LDA#&FF:STA&221:RTS
1400 ]:NEXT
1410 ENDPROC
1420 :
1430 DEF FNcheck
1440 tot=0:FORcheck=start TO P%
1450 tot=tot+?check:NEXT
1460 =(tot=&6DE5)
1470 :
1478 DEF PROCsave
1490 CLS:PRINT'"Save code with :"'"   *
SAVE <Filename> 900 ";~P%;" ";~initial
1500 PRINT'"Use *RUN <Filename> to exec
ute."
1510 ENDPROC
```

EDIKIT ROM

(incorporating Basic Booster)

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

- ***FTEXT** (find text) , ***FBASIC** (find Basic), ***FPROCFN** (find procedure/function) - three FIND commands, designed to help locate lines of programs according to their contents.
- ***LFROM** (list eight lines of a program), ***LPROC** (list procedure), ***LFN** (list function) - three commands, which list out significant segments of a program.
- ***RTEXT** (replace text) and ***RBASIC** (replace Basic) are find and replace commands directly analogous to FTEXT and FBASIC.
- ***SYSINF** (system information) - gives background information on the system variables and their sizes, together with the size of your program
- ***VARLIST** (list program variables) - lists variable names
- ***FKDEFS** (function key definitions) - prints out current function keys definitions.

```
>*Help Edikit

EdiKit 1.10

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

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

Including the updated Basic Booster utilities:

SUPER SQUEEZE - A program compressor.

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

PROGRAM LISTER - List any program direct from a file.

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

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

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

EDIKIT (including Basic Booster) is available on:

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

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

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

Phone your order now on (0727) 40303

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

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

EXEC COMMAND TAILS

Andrew Rowland

Sometimes you need to supply extra information to an EXEC file, for example a filename. However, you cannot use INPUT within an EXEC file, as it will always take the next line from the file rather than from the keyboard. The answer is to give the file a parameter when called, for example

```
*EXEC DELETE filename
```

where DELETE is an EXEC file. The filename is called the *command tail*, and the first part of listing 1 (an EXEC file called *DELETE*) shows you how to access it.

It assumes you called the file from Basic's command mode, and reads the right-most word from the original *command into the variable *tail$*. Note that you must specify the parameter, and that you must type *EXEC or *E. - do not shorten this to just a star.

The lines following the single star are an example for ADFS, which instead of merely deleting the file specified, renames it into a directory called $.LIMBO. Files are therefore not permanently deleted but put 'in limbo', in case you should change your mind. When you need the disc space, enter *DIR $.LIMBO and delete the files in the normal way. On my disc, I also *STAMP the files with the date using Bernard Hill's utility (BEEBUG Vol.6 No.10 p.45), so that I can see which are oldest.

Another application is suggested in listing 2 - just replace the lines following '*' in listing 1 and save the result as an EXEC file called *SAVE*. Use it to save Basic programs with:

```
*EXEC SAVE filename
```

It automatically saves any existing program of the same name into $.BACKUP before saving the new program. Remember to create the directories $.LIMBO and $.BACKUP before using these EXEC files.

Listing 1

```
I%=0:IF RIGHT$($!&F2,1)=" " REPEAT $!&F2
=LEFT$($!&F2,LEN$!&F2-1):UNTILRIGHT$($!&
F2,1)<>" "
REPEAT tail$=RIGHT$($!&F2,I%):I%=I%+1:UN
TIL ASC(RIGHT$($!&F2,I%))=32 OR
I%=LEN($!&F2)
IF I%=LEN($!&F2) tail$=""
*
OSCLI"ACCESS "+tail$+" WR"
OSCLI"RENAME "+tail$+" $.LIMBO."+tail$
```

Listing 2

```
OSCLI"ACCESS "+tail$+" WR"
OSCLI"DELETE $.BACKUP"+tail$
OSCLI"RENAME "+tail$+" $.BACKUP."+tail$
SAVE tail$
```

TRANSFERRING VIEW FILES TO PC SYSTEMS USING THE 512

Ian McWhinnie

To take a View file created on a Master under the ADFS and transfer it to an MS/DOS compatible disc using the 512 co-processor, proceed as follows:

1. Remove any embedded commands, such as *PB* or *CE*, and load the file into Edit using *EDIT <filename>.

2. All control codes appear as inverted characters on a white background. View has a Return at the end of each line (|M), and two to terminate a paragraph. GEMWRITE, for example, uses a Return together with a Line Feed (|J), and a paragraph stop (|T). Use f5 in Edit (global search and replace) to convert control codes as required. For GEMWRITE:

```
LOAD View file into Edit
Select f5, global search
Type: |M/|M|J <Return>
Select f5, global search again
Type: |M|J|M/|T|M|J|M <Return>
```

To remove unwanted control codes, for example Ctrl-Z used for justifying:

```
Type: |Z/ <Return>
```

To replace Tabs:

```
Type: |I/<8 spaces> <Return>
SAVE file with another name.
```

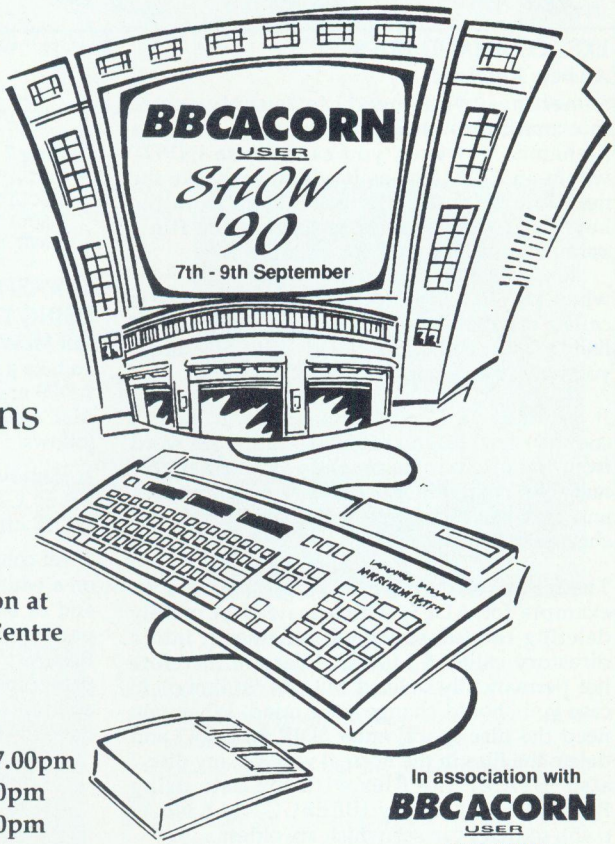
Access DOS on the 512, and format a disc as appropriate for the receiving PC. Copy GETFILE.CMD and PUTFILE.CMD on to it. With this disc in drive 0, and your edited View file on disc in drive 1 type:

```
GETFILE :1.$.<filename> a:<filename>.DOC
```

Then use COPY to transfer the file to other directories if necessary.

This has been tested with two MS/DOS systems, and a View file has found its way successfully onto Unix. B

Make a Date in Your Diary For the



*New Products
*Old Favourites
*Workshops
*Seminars
*Demonstrations
*Daily Competitions
*Free Draws
*Fabulous Prizes

Find us close to Victoria Station at
The Westminster Exhibition Centre
(Horticultural Hall)
Elverton Street, London SW1

Friday 7th Sept .... 12 noon to 7.00pm
Saturday 8th ..... 10am to 6.00pm
Sunday 9th ..... 10am to 6.00pm

In association with
BBCACORN
USER
MAGAZINE

Complete Range of BBC ACORN Hardware and Software

BOOK NOW, SAVE MONEY

For advance tickets fill in the coupon below and send with remittance to:-
SAFESELL EXHIBITIONS
Market House, Cross Road, Tadworth, Surrey KT20 5SR

Advance tickets:- Adults £3.50 Under 16's £2.50 At the door:- Adults £4.00 Under 16's £3.00
Please send ..... Adult tickets at £3.50 each. Please send ..... Under 16's tickets at £2.50 each
I enclose cheque / P.O. for £..... Payable to **Safesell Exhibitions Ltd.**

DON'T SEND CASH !!

PLEASE ENCLOSE A STAMPED ADDRESSED ENVELOPE

POSTBAG

ARCHIMEDES THEFT

I would like to inform you that my A310 system was stolen today. The details are as follows:

A310 (serial number 27-AKB15-1000816) with Watford I/O controller, BEEBUG 40Mbyte drive (Ser. No. 20052), Computer Concepts ROM/RAM podule, Watford 5.25" disc buffer, Cannon 5.25" drive, Acorn colour monitor (28-AKF11-1001998) and Star LC-10 printer (313080772520 I think).

Yours depressed, Martin Platts, Birmingham.

If you should happen to come across any of the above items or have information which might be relevant, then please contact BEEBUG and we will pass the details on to Mr.Platts. It may also be opportune to remind readers of the desirability of checking that their computer system is adequately covered by household insurance, and that reasonable attention is given to security aspects.

GOOD VIBRATIONS

The May 1990 issue of BEEBUG (Vol.9 No.1) has fortuitously drawn together several topics which are of particular interest to me: Mechanical Vibrations, Curve Fitting, and the letter from Mr.Horne on beam design. From the reference to the Central Heating program (from D.J.Holden - see News, Vol.8 No.8), I wonder if he, like me, is a builder.

Through necessity, I have had a program up and running for several years which produces shear force and bending moment data (and diagrams) for simply supported beams with point loads, and loads distributed over all or part of the span. Although partially fulfilling my needs, I thought there must be a better way.

With some help, I discovered that there is a unifying concept, namely the finite element method. Two books have been particularly useful: *Computational Methods for the Solution of Engineering Problems*, C.A.Brebbia & A.J.Ferrante, Pentech Press, and *Finite Element Analysis - From Concepts to Applications*, by D.S.Burnett, Addison-Wesley. Both give numerous Fortran listings, and cover vibrational topics such as Donald Tattersfield's; both also demonstrate the need for curve fitting functions (hence the reference to Sheridan Williams' article). Coupled with the articles on matrices in BEEBUG Vol.7 Nos.3 & 4. BEEBUG has provided much useful information.

Arising from this, I wonder whether any BEEBUG members have converted any of the above Fortran routines to Basic (I don't really have the time, or inclination if it has already been done). Also, how does one get to find out about sources of structural programs such as these. There must be many such around.

Like Mr.Horne, I should be grateful for any help on the subject, and if either my *Moments* or *Heatloss* programs may help or spur anyone to enhance them, the offer is there.

R.J.Lindsell

It is good to know that BEEBUG programs can be put to sensible practical use. If any other reader wishes to contact Mr.Lindsell then they can do so c/o BEEBUG, and we will forward the letter. As far as sources of such programs are concerned I would have thought relevant trade or professional magazines or journals, or associations, might be the most fruitful.

THE SHORTEST ROUTE

I should be grateful if you would tell me if I can obtain a *shortest route* program from anywhere. For example, if I were to enter town names with distances between them, I would want the shortest route to visit them all.

If you do not know of one, perhaps you know of a program to produce anagrams. I am confident that such a program could be modified to achieve my aim above.

John Holt

Unfortunately, we are unaware of any suitable programs for the BBC micro to solve what is also referred to as the ***travelling salesman*** *problem, although such programs certainly exist for larger machines. As far as anagram programs are concerned, Computer Concepts' Spellmaster has this facility, but being in ROM format does not lend itself to modification. If any readers can help, then write to Mr.Holt c/o BEEBUG and we will forward your letters.*

Personal Ads

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

We also accept members' Business Ads at the rate of 30p per word (inclusive of VAT) and these will be featured separately. Please send us all ads (personal and business) to MEMBERS' ADS, BEEBUG, 117 Hatfield Road, St. Albans, Herts AL1 4JS. The normal copy date for receipt of all ads will be the 15th of each month.

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

Master 128 with 3.5 DSDD disc drive, tape, cartridge ROMs, all software discs and full set of manuals. All in first class condition, home use only. Tel. (0283) 31403.

Magazines & books; magazines include: A&B Computing and BBC Acorn User. Tel. 021-449 7211 after 5pm for details.

Wordwise Plus ROM £20, Penfriend 2 ROM £8, BEEBUG Sleuth ROM £9, Procyon ROM £4, CJE Micros Multifont NLQ ROM (incl. 4 font discs) £25, Mini Office II (Master ADFS 40T) £10, Domark "007 - Licence to Kill" (DFS 40T) £11, Silversoft "Bored of the Rings" (tape) £3, EMR Miditrack Performer V1.2S (Master 80T DFS) £10, Acorn Music 500 (Master compatible - can be upgraded to Music 5000) £30, Acorn Teletext Adaptor with Adv. Teletext ROM £50, all originals complete with manuals. Tel. (04853) 34335.

WANTED: Computer Concepts Spellmaster ROM & letter quality 24 pin dot matrix printer, preferably Epson LQ400 or Epson LQ550. Tel. (0772) 424573.

BBC M128, twin 40/80T 5.25" drives, Microvitec monitor, Quest Mouse, AMX Pagemaker, Data Recorder, Welcome Guide, Reference Manuals 1&2, Advanced Reference Manual, Mouse User Guide, Joystick, lots of games £650. Also BBC B 32k, Light Pen, Speech Synthesiser, Data Recorder, £400 of games - £250. Tel. (074488) 4195.

FOR SALE: The following cassettes for BBC model B, some unused, others used 2/3 times: Worwise - C.C.,Dr Soft 747 key & joystick version, Blitzkrieg Software Invasion, Micro User Space Pilot, Micro User Castle of Fear, BBC Welcome pack, Shado/Azimuth Calibration cassette, £2 each. Gemini Word Processor 32k, BBC Soft - Home Finance, BBC Soft - White Knight The Chess Master Mk 12, all unused £5 each. Know your own p.s.IQ - Mirrorsoft, unused £3, Watford Electronics Print ROM, unused £10. Tel. (0344) 56277.

Master Compact base system with TV modulator, original packing, £175. Tel. (0703) 445532 (day).

M128 with Reference Manuals 1&2 & cartridge etc. £325, Brother M1109 Printer with Tractor Feed and spare ribbons £120, Z80 co-processor £85, 65C102 Turbo co-processor £85, Plinth mounted 5.25" DS/DD and 3.5" DS/DD disc drives with PSU £130, Phillips green screen monitor £40, Viewstore £25, all excellent and complete (upgrading). **WANTED:** Watford co-processor adaptor. Tel. 051-647 5367.

Acorn Master cartridges £6 each, Original double Acorn joysticks £11, Voltmace Datapad 16, £18, Voltmace Delta 14 joystick + keypad £12, Micropulse External 8socket switchable ROM box £30, ACP Toolkit ROM £20, new Master Reference manuals 1&2 £10 each. Tel. 081-989 2666.

A310 & RISC OS extra software £495. Tel. (0271) 850355.

M512 in Viglen PC type case, with 3.5" & 5.25" DD, Acorn Teletext Adaptor, Watford Mouse, M512 Mouse, Watford User Port Splitter, original Master case & packaging, software available includes GEM suite, selection of PC Shareware, Quest Paint, Dabs MOS-Plus, DataScribe (integrated WP/Database), Vu-type typing tutor, several ROMs and games, all in good working order (have upgraded to A3000). £500, or would consider exchange for Star XB24-15 printer or similar. **Hybrid Music 5000** synthesiser and music 4000 keyboard in good working order, with many pieces of music on disc, AMPLE Nucleus programming guide, and M5000 & M4000 user guides, £200. **Digisound 80 monophonic synthesiser** with 4-octave digital keyboard, fully patchable, includes VCOs, VCA, VCLF, sample & hold, and envelope generator modules, with joystick and portamento, plus 2 non-working modules which require attention, manual. Cost over £400, offers around £150. Tel. (0603) 897511.

BEEBUG C programming language for B, B+ or Master (Care Master cartridge), on two 16k ROMs. Package includes 40T library disc, BEEBUG C Stand Alone Generator, User Guide and Dabhand guide to C by Mark Burgess 2nd Ed. cost £67, offers? Tel. (0245) 71841.

BBC B issue 7, software + data recorder £125. Memoftech MTX 512, manual + software £30. Tel. (0993) 776066.

Cumana (3x) 80T DS SD £70 o.n.o. TRS 80 colour graphic printer £115, £90 o.n.o. Tel. (0372) 375002 9am until 5.30pm.

Archimedes A3000, brand new, perfect condition £525 o.n.o. Tel. 081-560 7310.

Spectravision & Acorn joysticks £10, ROMs: Disc Doctor, Dot Print NLQ, Viewspell; £10 each. Books for BBC B: Advanced User Guide £4, Creative Assembler, Griffiths £2, Creative Graphics, Cownie £1, Assembley Language, Birnbaum £4. Software: Glentop 3D Graphics Development System £15; Viewplot £5. Tel. (0306) 889647 eves.

EPROM Programmer £25, Apollo Modem & software £40, Commstar 1 £15, Printmaster £25, Stop Press, SuperArt, Mouse & software £95, Extra Extra £15, LOGO (Acornsoft) £45, Mailing List £5, Mail Merge £5, 3.5" Elite (Master) £10, 3.5" Music System £15, Master reference guide part 1 £5, New Advanced User Guide £5, BEEBUG mags (Vol. 7-2 to 9-1) £10, all softwar boxed as new. Tel. (092575) 5139 after 6pm.

WANTED: Adventure games, Master compatible on tape or 5.25" discs, also Arkanoid on 5.25" disc. Tel. (0932) 783947.

M512 with GEM software and mouse, Phillips CM8833 colour monitor, Watford dual 40/80T drives and double plinth, joystick, 512 Shareware collection, 512 User Guide & disc, software includes View, Viewsheet, Dump-out 3, Studio 8, Masterfile II and Bank Manager Master. Complete with all manuals/Guides etc. £600. Tel. (04024) 58772.

M128 with 40/80T disc drive and green screen monitor £435, Wordwise/WordAid £25, Master ROM £15, manuals 1&2 £16, all under 18 months old. Tel. 021-351 1389.

WANTED: Listings on 80T SS disc of all 12 programs provided in the BEEBUG Members Pack. Suggest a payment of £4 to cover cost of disc, time involved, post and packing. Tel. (0903) 784328.

Interword ROM, boxed, with full documentation, keystrip and manual £35 o.n.o. Tel. (0454) 260 668.

BBC Master ROMs: ACP Advanced DFS £15, CC Mega 3 £50, CC Interbase £30, PMS Publisher & extra fonts £45, Watford Quest/Conquest & Mouse £60, Wapping Editor & Art disc £50, A&B XBASIC £5, discs & books available. (0452) 506362.

Advanced Control Panel ROM, V1.4 £10, Star Gemini 10X Printer, needs PSU £10, Star Gemini 10X Printer good condition £80, 40T dual D/S disc drive with PSU £80, WE EPROM programmer, less ROM, hence £25, all prices o.n.o. Tel. (0734) 782636.

BBC B issue 7 with DFS, 40T single sided disc drive, Solderless sideways ROM board, 2x16k RAM chips, Interword ROM, SuperArt ROM & mouse, Stop Press ROM and Replay ROM, loads of computer games on disc, total cost well over £1000, bargain price of only £500 o.n.o. Will separate if neccesary. Tel. (0734) 669651 (eves).

BBC Teletext adaptor and TFS £75 o.n.o. Wordwise, Word EX, Printmaster £20 each, Spellmaster £40 offers welcome, Prism Modem 1000 (sorry no book) £25 o.n.o. Tel. 081-886 6475.

Disc Drive: 5.25" 40T, S/S, suit "B" or Master and only £25. Tel. (0679) 62728.

Quest Mouse & Quest Paint (BBC), boxed as new £35, forty original disc games and adventures (Elite, Exile), fifty tapes (mainly adventures) many with hint sheets, light pen, Voltmace joystick, BBC books, BEEBUG mags (Jan '86 - Jan '90) all for £100. Tel. (0227) 374172.

Master 128, monitor, Viewspell, BEEBUG C, Reference manuals, Lightpen, mouse and many discs £510. Will sell separately. Tel. (0427) 890750. After 5pm.

Steel experimenters box designed to fit over the BBC model B.
Will support a monitor and house 3" or 3.5" drives,
only £15 each (incl p&p).
Tel. (0527) 545322.
18 Tanwood Close, Callow Hill, Redditch, B97 5YU

BBC model B with Toolkit, Wordwise Plus and CP ROMs, Taxan colour monitor 40/80T, Pace twin D/D, joysticks & books £255. Tel. (09274) 24063.

WANTED: "The Quill" by Gilsoft (Adventure Generator as reviewed in BEEBUG Vol.5-5) on 5.25" disc, DFS format. Tel. (0473) 723370. eves/wkends.

BBC B (issue 7, Opus DDOS), Opus DS 40/80T disc drive, Voltmace joystick, Replay ROM fitted & several BBC books £300, 50 BBC games cassettes £80, 20 BBC games discs £70, or BBC with all games £400, Electron with 15 games & joystick £40, sensible offers considered. Tel. 061-626 9170.

BBC B Cumana 40T disc drive, black/white TV monitor £500 o.n.o. Tel. (0473) 624297.

WANTED: All other competent BBC fans who have their own BBC Master or BBC B to write to with ideas and own programs. I am 16 and have my own BBC Master, I am looking for intermediate level people; if enough write, who knows we could start a club! Please write to: Michael Scott, 195 Heath Road, Leighton Buzzard, Beds, LU7 8AF. SAE if possible.

BBC Master 512 Computer, with single 80T DS/DD disc drive in double case with power supply, Hitachi hi-res. green screen monitor, Mosplus ROM, 2 ROM cartridges, Reference manuals 1&2, and lots of disc software (DOS& DFS/ADFS) £450. BBC B issue 4 with APL ROM board with battery backup 16k RAM, in Viglen console, with 2 40T DS/DD drives, twin joysticks, Viglen cartridges, Wordwise plus, Wordaid, Spellcheck II & Disc Doctor, plus lots of discs £300. Olivetti JP101 Spark-jet printer, with BBC cable £40, 6502 second processor, Hi-Basic, Hi-WW+ and 6502 Development package (Assembler) £90. Also BCPL package (inc. Maths & Direct access) £30, Lisp (disc) £10, Accelerator (Basic compiler) £20, Floppywise+ £10. Tel. (0483) 39496 (work) or (0273) 605339 (eves.).

Printer, Brother M-1009 with tractor feed, quiet and reliable £70. Tel. (0582) 768328.

Adventures wanted to complete collection: DF's Waxworks, Midwinter, After The Fire and Beyond the Infinite, originals only, reasonably priced please, write to; D Shepherdson, 3 Tarn Villas, Cowpasture Road, Ilkley, West Yorks LS29 8RH.

WANTED: DisplayWrite I or II on 5.25" floppy for use on 512 DOS-Plus including documentation. Tel. 081-398 8733.

BBC model B issue 7 plus £250 of software including games, home accounts and word-processing, also manuals etc. plus micronet ROM £165. Tel. 071-836 8333 (day) or (0895) 621953 (eves).

Master 512/1024 (Solidisc PC+) with dual 40/80 D/S D/D, Zenith mono monitor, View etc., cartridges; Master ROM; Dabs M512 Guide, Disc & Shareware, see working. £500 o.n.o. Also BBC B with Torch Z80 disc pack (dual D/S 40/80). BBC MOS: extra RAM and ROM box, Dataplus, Vufile, etc. CPM: Perfect Writer, Speller, Calc, Filer, BBCBASIC Z80. £350 o.n.o. Tel. 021-308 0224.

Z80 second-processor, complete with all software and manuals £50, Prism 2000 modem, with comms software and lead for BBC £20. Tel. (0438) 833575 (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 & RISC USER
£16.90	1 year (10 issues) UK, BFPO, Ch.I	£25.00
£24.00	Rest of Europe & Eire	£36.00
£29.00	Middle East	£43.00
£31.00	Americas & Africa	£46.00
£34.00	Elsewhere	£51.00

BACK ISSUE PRICES (per Issue)

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

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

POST AND PACKING

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

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

BEEBUG
117 Hatfield Road, St.Albans, Herts AL1 4JS
Tel. St.Albans (0727) 40303, FAX: (0727) 60263
Manned Mon-Fri 9am-5pm
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by BEEBUG Ltd.

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

All rights reserved. No part of this publication may be reproduced without prior written permission of the Publisher. The Publisher cannot accept any responsibility whatsoever for errors in articles, programs, or advertisements published. The opinions expressed on the pages of this journal are those of the authors and do not necessarily represent those of the Publisher, BEEBUG Limited.

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) 1990

Printed by **Newnorth Print Limited** (0234) 41111 ISSN - 0263 - 7561

Magazine Disc/Cassette

JULY 1990 DISC/CASSETTE CONTENTS

ENIGMA: A WORLD WAR II CODING MACHINE - An encoding program simulating the Enigma algorithm, and a coded file Freya containing the first message successfully decoded by British intelligence.

PHONE CALL COSTING - Monitor the costs of your phone calls with the help of this program, with versions for the BBC B and the Master.

THANKS FOR THE MEMORY - BAS 128 - Two programs providing a set of procedures to be used with Bas128, and a utility which traps the Break key and allows you to stay in Bas128.

FIRST COURSE: A Data Input Routine - A program demonstrating a data input routine.

A HIGH RESOLUTION GRAPH PLOTTER - This program demonstrates how to plot up to 9 high resolution graphs and use your printer as a continuous data recorder.

PRACTICAL ASSEMBLER (Pt 3) - Two programs, one to provide a print routine for inclusion in your programs, and one for an error trapping routine .

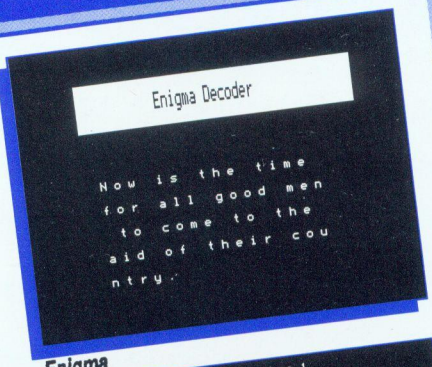
BEEBUG TABLES INVADERS - An educational program which turns the learning of multiplication tables into fun.

USING THE ROM FILING SYSTEM (Pt 3) - The Delete routine from this month's article plus an updated version of RFSload from Vol.8 No.9.

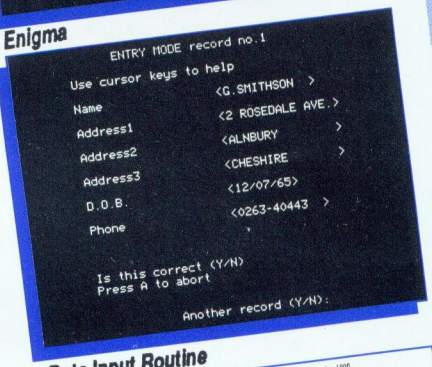
PERMANENTLY COLOURED TEXT - A short program which allows you to keep your text in colour even when you change the screen mode.

AUTOMATING ASSEMBLER INPUT - A handy utility which automatically puts Assembler commands in upper case and variables in lower case.

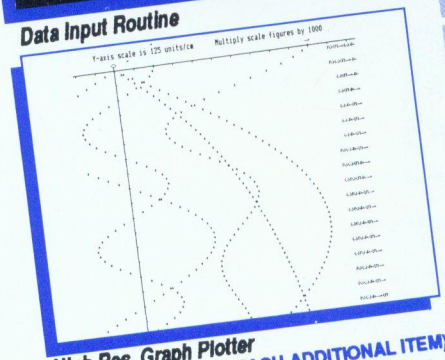
MAGSCAN DATA - Bibliography for this issue (Vol.9 No.3).



Enigma



Data Input Routine



High Res. Graph Plotter

ALL THIS FOR £3.50 (CASSETTE), £4.75 (5" & 3.5" DISC) + 60p P&P (30p FOR EACH ADDITIONAL ITEM)

Back issues (5.25" disc since Vol.3 No.1, 3.5" disc since Vol.5 No.1, tapes since Vol.1 No.10) available at the same prices.

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

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

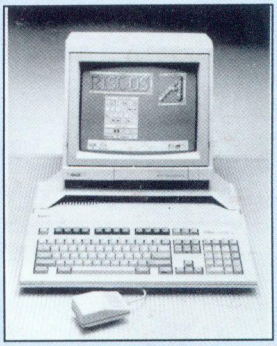
Cassette subscriptions can be commuted to a 5.25" or 3.5" disc subscription on receipt of £1.70 per issue of the subscription left to run. All subscriptions and individual orders to:

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

BEEBUG The Archimedes Specialist BEEBUG

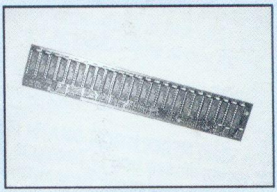
SPECIAL SUMMER PROMOTION

THE A3000 LEARNING CURVE



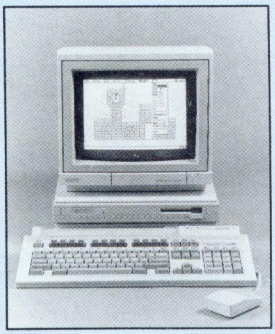
Acorn's new Learning Curve Package consists of an A3000 computer, First Word Plus V2 (the number one Word Processor package for the Arch), the PC Emulator, Genesis (A graphic based database system with a number of sample files). A parent's guide to the national curriculum and a demonstration video (VHS) is also included.

If you purchase an A3000 or Learning Curve from Beebug in addition to the backup service for which we are renowned, we will supply a free Beebug 1Mb RAM board. This board uses surface mount technology to achieve its remarkably small size of 256x46mm. A special feature of the Beebug upgrade is that it is expandable by the user to 3Mb (giving a total of 4Mb), unlike most other products. We believe that this is the best RAM board available for the A3000.



Learning Curve (no monitor)	699.00 (803.85 inc. vat)
Learning Curve (Acorn Monitor)	878.73 (1033.80 inc. vat)
A3000 (no monitor)	585.83 (688.85 inc. vat)
A3000 (Acorn Colour Monitor)	780.98 (918.80 inc. vat)

THE ARCHIMEDES 400/1 SERIES



For a limited period we are offering a number of unbeatable offers on the 400/1 series computer. It is our company policy only to supply what we believe are the best products available for the price. Our upgrades are equivelent or higher in specification to those supplied by Acorn.

OFFER 1 - £1099.00 (£1263.85 inc. VAT)

An Archimedes 410/1 upgraded with either a high quality 20Mb hard drive or a Star LC-10 Colour printer.

OFFER 2 - £1499.00 (£1723.85 inc. VAT)

An Archimedes 410/1 with a 40Mb fast (28ms) hard drive and either a free Samsung Multi-Sync Colour monitor (RRP £459) or an additional 3Mb of RAM to give the maximum 4Mb of user RAM.

OFFER 3 - £2099 (2413.85 inc. VAT)

An Archimedes 410/1 with a 50Mb Hard drive (28ms access time), 3Mb of RAM (giving a total of 4Mb) and a choice of either a Taxan 775+ Multi-Sync Colour monitor or a Canon Bubblejet BJ130e printer offering near laser quality (supplied with a RISCOS printer driver).

Please add £230 (inc. vat) for an Acorn colour monitor or £459.95 for a Taxan 775+ Colour Multi-Sync if required.

Please *add £8.00 for postage & packaging per system.*
Prices & specification subject to change without notice.

BEEBUG Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS
Telephone: 0727 40303 (24 hours) Fax: 0727 60263