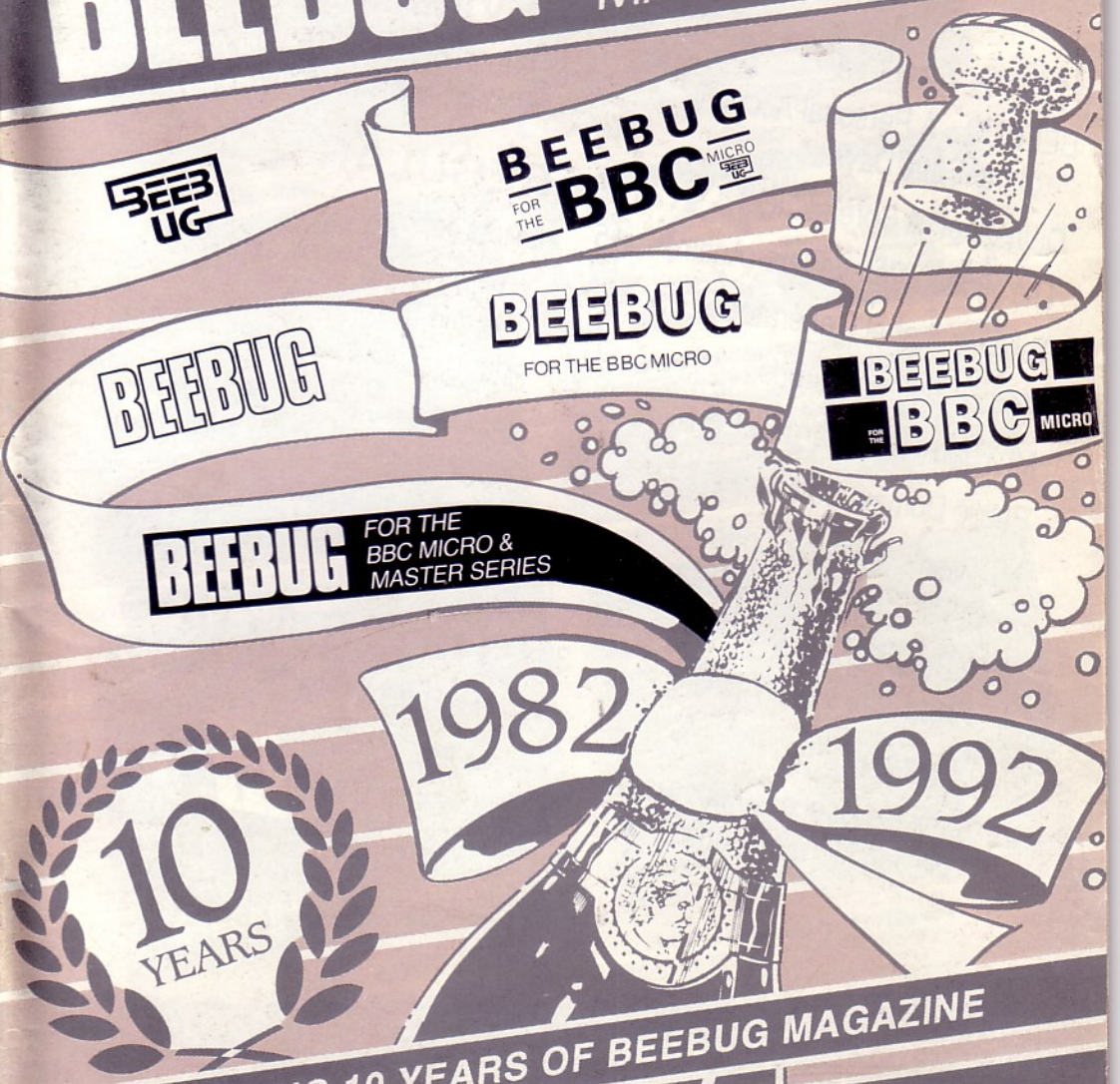
Vol.11 No.1 May 1992

BEEBUG

FOR THE BBC MICRO & MASTER SERIES



CELEBRATING 10 YEARS OF BEEBUG MAGAZINE

SUPPORT ● HELP ● INFORMATION FOR USERS

BEEBUG Vol.11 No.1 May 1992

FEATURES

REVIEWS

REGULAR ITEMS

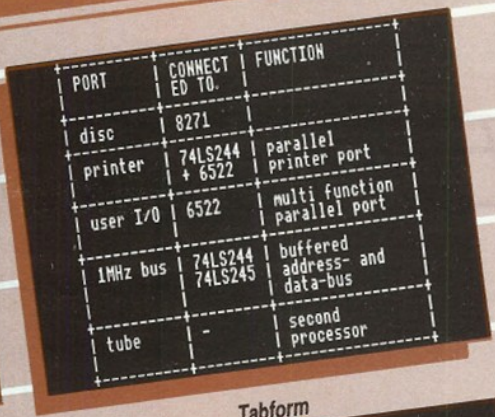
HINTS & TIPS

PROGRAM INFORMATION

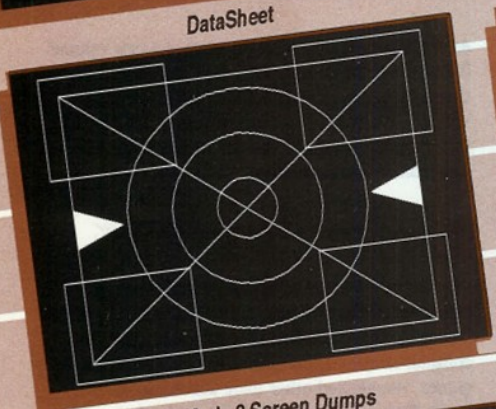
All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly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

DataSheet

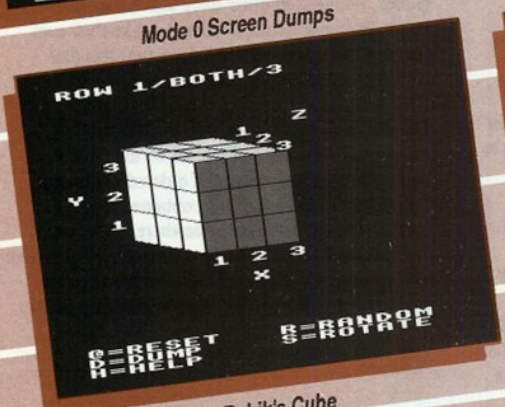


Tabform

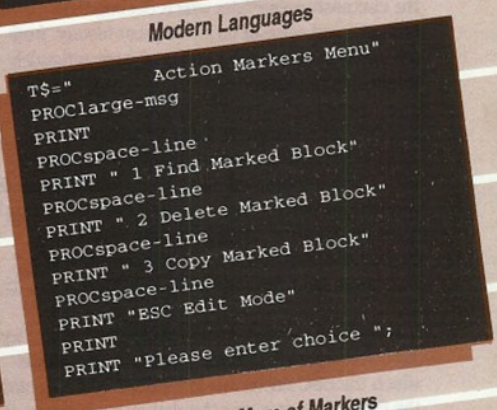


Mode 0 Screen Dumps

Modern Languages



Rubik's Cube



Making More of Markers

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 needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings

BEEBUG CELEBRATES TENTH ANNIVERSARY

Welcome to the Tenth Anniversary issue of BEEBUG magazine. Beebug was involved from the very first announcement of the BBC micro back in 1981 when the BBC announced its Computer Literacy project, and the first issue of the magazine was published in April 1982. To mark this occasion we are publishing an article by Lee Calcraft, who under the pen name of David Graham was one of the founders of Beebug, and editor and contributor to the magazine for several years. The article recounts for the first time the circumstances under which Beebug was founded, and some of its subsequent history, from the personal stand-point of one of those intimately involved at the time.

Lee Calcraft no longer has any involvement with BEEBUG magazine, but in a freelance capacity co-edits and contributes to RISC User, our magazine for Archimedes users. The other co-founder of Beebug is Sheridan Williams, an occasional contributor to Beebug and RISC User, and still very active as a Director of both Beebug and of RISC Developments.

It is doubtful whether either realised just what they were starting back in 1982, nor the huge success which the BBC micro was to become. By developing the BBC micro, to a specification put together by the BBC, Acorn directly and indirectly created an entire world, involving in one way or another millions of people worldwide. It is also true to say that many livelihoods now depend on the continuing success (or otherwise) of the Acorn market. Despite various ups and downs (and Acorn's rescue at one stage by Italian giant Olivetti), Acorn's Archimedes range (itself launched nearly five years ago now) has set the base line for an expanding and successful future for all involved.

VOLUME TEN INDEX

As is always the case with the start of a new volume (Volume 11), we include with each mailed out copy of this issue a free printed index to the whole of Volume 10 arranged for easy reference. We are also including the complete Volume 10 MagScan index on this month's magazine disc.

SPECIAL TENTH ANNIVERSARY DISC

To mark ten years of publication we have compiled a collection of what we consider to be a selection of the best programs published over the years. The disc contains serious applications and utilities, some games, visuals and other more leisure oriented items. Everyone could, of course, make their own personal selection, but we believe that this disc contains something for everyone.

The disc is available now, only to BEEBUG members, at the price of just £4.95 inc. VAT. Post and packing is extra (£1.00 if ordered on its own). Full details of the programs contained on this special disc are published elsewhere in this issue. This disc will be available for a limited period of time only (probably until initial stocks run out) so we recommend you order the disc as soon as possible.

FOUNDER MEMBERS

From time to time some readers have referred to themselves as founder members of BEEBUG. It may be of interest to note that there are still 475 readers whose subscriptions run continuously from Vol.1 No.1. Our many thanks to these members for their long-standing and loyal support of BEEBUG.

M.W.

BEEBUG News

THE DRAGON ROARS

A new game, *Explorer*, has been released for the BBC micro by DragonSoft. And quite a game it is too, comprising a 16K EPROM and two discs. Explorer is essentially an adventure game in which you must penetrate the fire-filled underworld of Hallar, conquer the fiendish Vandar, plunge through icy polar regions, rugged mountains, and danger infested jungles. There are many hazards on the way, but help is sometimes at hand as well, as you seek to redeem your tarnished reputation as a member of the Explorers Club.

Explorer costs £25.99 from Dragonsoft, P.O.Box 22, Whitchurch, Shropshire SY13 2ZZ, tel. (0948) 840522.

MUSIC PUBLISHER RELEASED

Back in BEEBUG Vol.10 No.4 we reported on the imminent release of Hybrid Technology's *Music Publisher* for the BBC micro and Master series. Music Publisher deals with the whole process of music composition and score layout, with support for both 9-pin and 24-pin dot matrix printers. After much delay, this product is now available, and we expect to be publishing a review in the June issue of the magazine. Music Publisher costs £60 ex. VAT from Hybrid Technology Ltd., 88 Butt Lane, Cambridge CB4 6DG, tel. (0223) 861522.

SHOWS FOR COMPUTER USERS

ALL FORMAT FAIRS

Future dates and venues for All Formats Computer Fairs are as follows:

Jun 7 City Hall, Candleriggs, Glasgow.
Jun 14 Haydock Park, Jct 23 M6.

For information and tickets contact John Riding on (0225) 868100.

BBC ACORN USER SHOW

Following the success of last year's event, this year's BBC Acorn User Show will again be held at the Wembley Conference Centre, but this time in the larger hall three of this complex. The show will run from Friday 16th October through to Sunday 18th October. We will bring you more detailed information nearer the time.

COMPUTER SHOPPER SHOWS

The Spring Computer Shopper Show is taking place from 28th to 31st May 1992 at the National Hall, Olympia, while the Christmas Computer Shopper Show is scheduled for the Grand Hall, Olympia from 19th November to 22nd November 1992.

EDUCATIONAL SOFTWARE BECOMES SHAREWARE

John Lyons Computer Software has announced that the majority of its 60 odd educational programs will become shareware. Some programs will also be treated as public domain (see this month's PD column). Shareware discs will be available for £1.50 in the first instance, but users who then continue to use the software will be expected to pay a further registration fee of £7.50. The buyer is then free to make copies as required for their own use, and will be sent full documentation plus any future updates.

By sending £1.50 readers can obtain a sample disc with details of all other programs from John Lyons Computer Software, Freepost, Camberley, Surrey GU15 3BR, tel. (0276) 65275.

ERIC FOR EDUCATIONAL SOFTWARE

We have received a copy of the latest catalogue from ERIC (Educational Resources in Computing). This lists a range of software covering Mathematics, English, Special Needs, French, Latin, History, Science and Geography. All discs cost £14.50 each (inc. VAT and p&p) and are available for the BBC micro, Archimedes and RML Nimbus. For a copy of the catalogue or for more information phone ERIC on (0903) 872400.

Beebug - A personal Ac

by Lee Calcraft

The Beebug story starts back in 1981 when home computing was in its infancy. There were very few affordable machines around: you had to make do with a Sinclair ZX80 or 81, or one of the single-board machines such as the Nascom or the UK101 or the Acorn Atom. For people with more serious budgets there were also early versions of the Apple and Commodore Pet. Into this relative vacuum there appeared the BBC micro - or if not the machine itself, at least the machine's specification.

In that year, the BBC announced the so-called Computer Literacy Project, and after some debate Acorn were chosen as the suppliers of the machine which would spearhead the project. This was an enormous coup for Acorn since it placed them centre-stage in a massive media-based project. They would produce the machine to a spec laid down by the BBC, and the BBC would take a 10% royalty on all machines sold.

The spec was a very full one, and included a vast array of interfaces as well as full colour graphics and sound support. In the autumn of 1981 it occurred to me that not only would this machine be well worth having, it might also provide an opportunity to start a computer user group - an idea which had appealed to me for some time.

In December '81 I proposed the idea as a joint venture to Sheridan Williams. Sheridan lived just around the corner from me at the time, and we had met in connection with work on Personal Computer World magazine. Sheridan ran the popular Computer Answers pages, and I had written a number of articles for PCW on computer interfacing, and he turned up on my doorstep one day with some hardware questions for his column. I remember that this first encounter began with some confusion. Sheridan asked me if I knew a Mr Graham. I said that I didn't, but then recalled that D.E.Graham was a pen name that I had been using in PCW. Since then we had kept in touch.

ount of the Early Days

Sheridan had been involved in running the Research Machines user group, and was very positive about the idea of a BBC user group, and we quickly got things moving. We sent in orders for a BBC micro each, and booked a quarter page ad in Your Computer. This magazine is no longer in circulation, but at the time it was read by 'home users'. Then we just sat back and waited.

A couple of days after the anxiously awaited ad came out, Sheridan dropped round to say that we had received a single membership application. Just one! But the next post brought 30, and the next 30 more, and the next 30 more; and this continued day after day. We began to register each applicant on a membership list (just a big piece of paper), and to bank some of the cheques, but there were just too many.

After we had got several days behind, we enlisted secretarial help to assist with the administration, but after a few weeks we were still getting behind, such was the torrent of applications. At this point we employed the services of a computer bureau to deal with the administration. We forwarded all applications on to them by Securicor, and they maintained a database of members. They would also mail out our magazine, and in due course issue renewal reminders and so on.

The volume of response to the small ads which we had placed shocked us both. The time was clearly right for such a venture, and what worked in Beebug's favour was the contrast between the desirability of the new BBC micro (and the hype surrounding it), and the dearth of information about it. Most of our early members had ordered the machine but had not yet received it, and Beebug seemed to provide a channel of communication that was otherwise missing. The fact that the machine had such potential when it finally did arrive, and that the initial 'provisional' manual was so sparse made Beebug an even more important source of information.

But the lateness of the machine also caused us a problem. Our promotional material was based on a projected first issue of our magazine in April 82, but by late March we had still not received our own BBC Micros. We were rescued from a potentially disastrous situation by the Mehta brothers who run Technomatic. Like other prominent computer dealers, they had been sent early BBC machines for demonstration purposes, and they were kind enough to lend us one for a couple of days. All the articles in the first issue of Beebug were created using this machine in just two or three days of frenetic activity, with the machine shuffling between my house and Sheridan's.

Sheridan entered all the text for this and many subsequent issues into a text editor on his trusty RML 380Z - a machine which actually had disc drives! - and we pasted the whole lot together and handed it to a bemused man at the counter of a local copy shop. Within a few days they shipped the finished magazine to the computer bureau, and some 2000 members received the first issue of Beebug magazine. After breathing a sigh of relief, Sheridan and I sat down to plan the May issue. We had created a kind of monster who required feeding vast quantities of programs, hints, news and other articles on a regular basis, for the forseeable future.

Both Sheridan and I were lecturers - he at Hendon College, and I at Hatfield Polytechnic; and running two full time jobs created a lot of pressure. But it was very rewarding work. Membership continued to grow - by the time that the fourth issue went to press we already had well over 7000 members, and we were getting very positive feedback from the membership. As the year went by we enlisted friends (and relatives) to help with editing, and by the end of the year we had secured some premises, and had employed a part-time editor, as well as a full time technical assistant.

That autumn we branched into software sales, and these proved a very useful addition to revenue. In 1983 my brother Adrian joined the company as software manager, and Mike Williams (no relation to Sheridan) as editor, and as you will know they are still with the company. In 1984 I quit my day job, and Sheridan followed suit a year or so later.

As the years went by the company grew, employing some 25 or 30 people at its zenith, with Beebug attaining a membership of close to 30000. In the summer of 1985 the company moved from its small suite of offices in Marlborough Road, St Albans to new premises at Dolphin Place, St Albans, which also housed a showroom.

In May '86 I reduced my shareholding in the company, and my brother Adrian took up the administrative reins, running the company jointly with Sheridan. This was something of a relief for me as it meant that I could get on and do some computing - as the company had grown over the years, so the amount of hands-on computing that I had time for went down and down. Under the new management the company has continued to do well, bringing out RISC User magazine in response to the launch of the Archimedes in July 1987, and moving into even larger premises in Hatfield Road, St Albans in August 1989. Here's to the next ten years!

To celebrate the tenth anniversary of the founding of BEEBUG magazine, we have put together a selection of the best programs which we have published over the past ten years. This disc is packed with applications, utilities and games most of which have not been available previously other than when first published in BEEBUG. All the programs come with full on-screen help files, which can be printed out as well for permanent reference.

BEEBUG 10th Anniversary Disc

Celebrate BEEBUG's 10th anniversary with this disc at the special low price of £4.95 and you will have something to celebrate too.

DRAUGHTS - An implementation of the classic board game in which you pit your wits against a computerised opponent.

KEYSTRIP DESIGNER - A very well written program to enable the creation, editing and printing of function key strips.

GARP - GARP (Geographical Atlas using Radial Projection) allows views of the globe to be displayed from any point above the Earth's surface.

MULTI-COLUMN PRINTING - This utility formats any text file into columns, and prints the result using an Epson FX-80 or compatible printer.

PERPETUAL CALENDAR - This program can display or print the calendar month by month for any year between 1753 and 5000 A.D. in the United Kingdom, or even earlier in other countries.

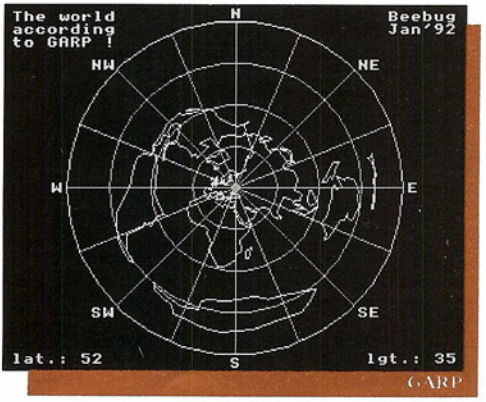
QUAD - Quad is a Tetris-like game, in which you must manipulate falling blocks to slot into each other. Dangerously addictive!

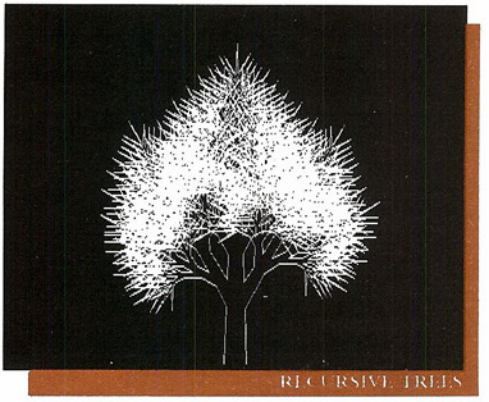
STEALTH - In this game you play against an opponent (or the computer), who sets a number of targets for you to find, and you must use your skill to discover the locations of the targets in as few goes as possible.

RECURSIVE TREES - This fascinating program uses recursion to create an infinite variety of tree-like designs - you choose a set of numbers, and the computer does the rest.

THE WORLD BY DAY AND NIGHT - This program will draw a map of the world showing graphically where the sun is in the sky or where it's night at every spot on earth.

CROSSWORD COMPILER - This program allows crosswords to be designed and the clues compiled.





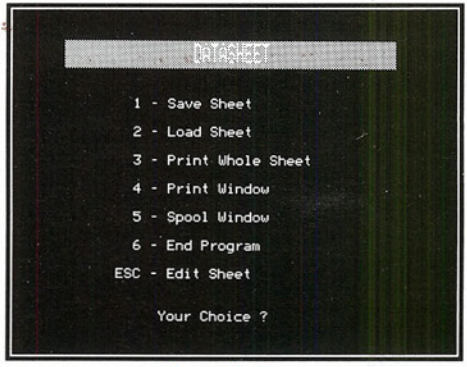
PBB3a 3.5" ADFS £4.95 inc. VAT plus £1 p&p
PBB5a 5.25" DFS 40/80T £4.95 inc. VAT plus £1 p&p

RISC Developments Limited.
117 Hatfield Road, St Albans, Herts AL1 4JS.
Tel. (0727) 40303 Fax. (0727) 860263

DataSheet

Stephen Colebourne presents the first part of his powerful Basic spreadsheet.

DataSheet shows that you can write full-feature business software in Basic. It is an easy to use, general purpose spreadsheet. For the uninitiated, spreadsheets allow you to ask any number of *What If* questions of numerical data. Typical uses are in accounts or stock taking. For instance, by entering details about your income and regular expenditure you could see if you could afford a holiday this year.



The Main Menu

THE PROGRAM

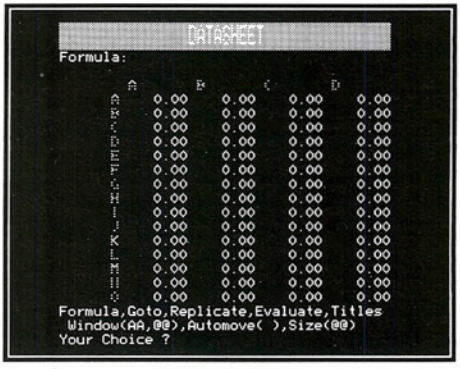
This month's listing is the skeleton version of the program which includes the screen displays and formula routines. This should allow you to get the feel of the program, but you will not be able to save or print anything you create, as these functions will be provided next month along with many others.

Type in the program as listed, making sure that the line numbers are followed exactly. When complete save as *SHEET1*.

CONTROLS

When run, the program first asks for a column width. This is the number of characters in each column of the display. Your choice will affect the number of columns on screen at any time and the largest value which may be handled. When setting up an important spreadsheet this must be considered carefully as it cannot be changed later. For now, press Return which will select the value of 8. Note that this default value can be altered in *PROC USER*.

The main menu has six options, including *Save*, *Load*, *Print* and *Spool*. However, until next month, only option 6, *End Program*, will work. This is the best way to leave the program. Star commands can be issued from the menu; simply type in the command, including the star, at the prompt. To reach the editing screen of the spreadsheet press Escape. Pressing it again will bring you back to the main menu.

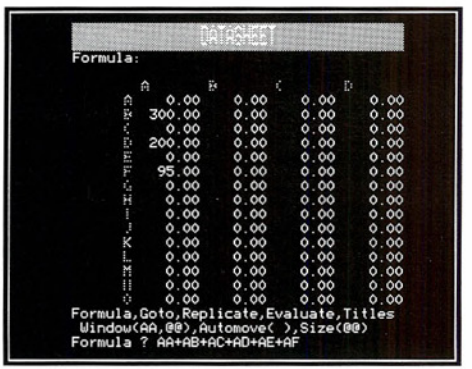


A Blank Page

EDITING

The editing screen should initially consist of 3, 4, or 5 columns of '0.00' - a blank sheet. This will be surrounded by reference letters in green along the top and down the side. Using these letters, each square in the sheet has a unique two letter code. To find this code you

take the letter given at the top for the column, and follow it by the letter given at the side for the row. e.g. Column 1, Row 3 is AC; and Column 25, Row 27 is Y@. Note that @ is the 27th letter in the 'alphabet' in this program, and represents the maximum size allowed.



Entering a Formula

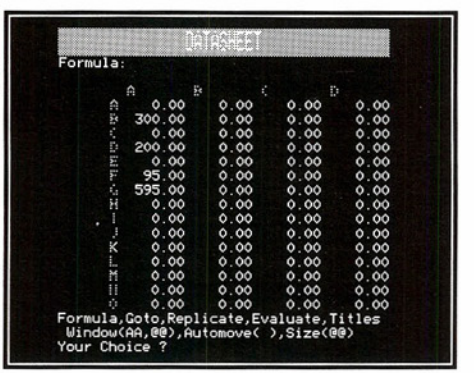
You move around the sheet using the cursor keys. Your current location is shown by the yellow cursor. To enter a value into the current square, simply type in the number. The computer will check it and format it before placing it in the sheet. Any previous value will be overwritten.

FORMULAE

The use of formulae is central to the operation of any spreadsheet. They allow calculations to be made and questions to be asked. To enter a formula press function key f0. A new prompt will appear at which you can type a formula up to 68 characters long. Reference to other squares is by means of the two letter codes described above. Typical formulae might be AA+AB or CC*(AC-BC). Once a formula has been accepted, the cursor will turn blue and the formula will be shown at the top of the sheet. The resulting will not be shown until evaluation.

Other Basic expressions can be entered; mathematical functions such as PI, SIN, COS etc. must be entered between square brackets (these appear as arrows on the screen). If this is not done the computer will treat them as references to squares in the sheet. The program converts all the two letter codes to a form which the computer can understand. Sometimes this conversion may not have the result you wanted - you will only be able to tell when you evaluate the sheet.

Evaluating the sheet will calculate the contents of all the boxes according to their formulae. This is done by pressing the Copy key. Evaluation takes place across each row before proceeding down to the next one, and takes a few seconds.



After Evaluation

OTHER FUNCTIONS

The formula key, f0, also allows access to three other commands, one of which will be added next month. Pressing f0 twice will delete the formula from any square but leave the actual number it last produced in that square. Pressing f0 followed by f2 will clear the present square to zero, whatever it's current state.

As with many programs, the best way to get to know what it can do is to experiment. There is an error handling system which is particularly useful if you type in an incorrect formula. It will show you both the version of the formula it thinks you typed, plus the version it converted it to. Try making an error and see!

NEXT MONTH

The second part of this article will add options that will allow you to load, save and print out your spreadsheets as well as adding many more powerful functions for the processing of your data.

```
   10 REM Program DataSheet
   20 REM Version B 1.0
   30 REM Author  Stephen Colebourne
   40 REM BEEBUG  May 1992
   50 REM Program Subject to Copyright
   60 :
   70 MODE135:PROCSETUP
   80 Q%=2:ONERROR GOTO160
   90 :
  100 REPEAT:PROCMENU:UNTILC$="6"
  110 PRINT'TAB(5)CHR$131"End Program (Y
/N)";
  120 INPUT" ? "C$:IFC$<>"Y" GOTO100
  130 CLS:*FX4,0
  140 END
  150 VDU3:CLOSE#0
  160 IF ERR=17 Q%=Q%-1:GOTO100
  170 VDU7:PROCP(380):*FX21,0
  180 PROCCLS:PRINT'CHR$131"Error:"'
  190 IF Q%<9 REPORT:PRINT" at line ";ER
L:PROCK:GOTO100
  200 PRINT"Bad Formula at square ";A$(V
%);A$(W%);":"
  210 PRINT'SPC2;F$(V%,W%)'SPC2;E$(V%,W%
)
  220 Q%=1:PROCK
  230 GOTO100
  240 :
  250 DEF PROCMENU
  260 IF Q%=1 PROCEDIT:ENDPROC
  270 Q%=2:PROCCLS:FOR Z%=1 TO 6
  280 PRINTTAB(8,2+Z%*2)CHR$134;M$(Z%):N
EXT
  290 PRINTTAB(6,16)CHR$134;M$(0)
  300 PRINTTAB(11,19)CHR$131"Your Choice
";:INPUT"? "C$:Q%=4
  310 REM
  320 REM
  330 REM
  340 REM
  350 REM
  360 IF LEFT$(C$,1)="*" PROCSTAR
  370 ENDPROC
  380 :
  390 :
  400 DEF PROCEDIT
  410 Q%=4:PROCCLS
  420 SX%=1:SY%=1:X%=1:Y%=1:M%=0:N%=0:AM
%=0
  430 PROCSCREEN:PROCMENUBAR
  440 REPEAT:R%=0:S%=0
  450 OX%=X%:OY%=Y%:OSX%=SX%:OSY%=SY%
  460 C$=FNIN("Your Choice",CW%-1)
  470 IF VAL(C$)<>0 ORC$="0" THENPROCVAL
UE
  480 IF C$="<" IFX%>1 X%=X%-1:IFX%-SX%<
1 ANDSX%>1 SX%=X%-1
  490 IF C$="*" IFY%>1 Y%=Y%-1:IFY%-SY%<
1 ANDSY%>1 SY%=Y%-1
  500 IF C$=">" ORR% IFX%<MX% X%=X%+1:IF
X%-SX%>CN%-1 ANDSX%<MX%-CN% SX%=X%-CN%+1
  510 IF C$="?" ORS% IFY%<MY% Y%=Y%+1:IF
Y%-SY%>13 ANDSY%<MY%-14 SY%=Y%-13
  520 IFOSX%<>SX% OROSY%<>SY% PROCSCREEN
  530 IF OX%<>X% OROY%<>Y% PROCPOS
  540 IF C$="F" PROCFORMULA
  550 REM
  560 REM
  570 REM
  580 REM
  590 REM
  600 REM
  610 REM
  620 IF C$="Q" ORC$="E" PROCCALC
  630 UNTILC$="M"
  640 ENDPROC
  650 :
  660 DEF PROCCALC
  670 Q%=9:FORW%=1TOMY%:FORV%=1TOMX%
  680 IF E$(V%,W%)>"" D(V%,W%)=FNCALC2
  690 NEXT,:Q%=4
  700 PROCSCREEN
  710 ENDPROC
  720 :
  730 DEF FNCALC2
  740 V=EVAL(E$(V%,W%)):IFV<VS THEN=VS
  750 IFV>VM THEN=VM
  760 =V
  770 :
  780 DEF PROCVALUE
  790 V=INT(VAL(C$)*100+0.5)/100
  800 IFV<VS ORV>VM THENVDU7:ENDPROC
  810 PROCSCRVAL(V):R%=M%:S%=N%
  820 ENDPROC
  830 :
  840 DEF PROCFORMULA
  850 F$=FNIN("Formula",68)
  860 IFF$="G" PROCSCRVAL(0):ENDPROC
  870 IFF$="F" F$(X%,Y%)="":E$(X%,Y%)=""
:F$=""
  880 IFF$="R" F$="":PROCSREP
  890 IFF$>"" PROCFCONV
  900 PROCPOS
```

```
  910 ENDPROC
  920 :
  930 DEF PROCFCONV
  940 P%=0:E$="":C%=0:B%=0
  950 REPEAT:P%=P%+1
  960 E%=FNPAIR(MID$(F$,P%,2))
  970 B%=B%+(PX$=")")-(PX$="(")
  980 IFPX$="[" C%=3
  990 IFPX$="]" C%=1
 1000 IFPX$="S" IFPY$="(" IFC%=0 E%=5:PR
OCSUM
 1010 IFE%=0 IFC%=0 E%=5-FNFCA
 1020 C%=C%-C%MOD2:IFE%<5 PROCFCC
 1030 UNTILE%=1 ORP%>=LEN(F$)
 1040 E$(X%,Y%)=FNFCB(E$):F$(X%,Y%)=FNFC
B(F$)
 1050 ENDPROC
 1060 :
 1070 DEF FNFCA
 1080 IF LEN(E$)>246 THEN=1
 1090 E$=E$+"D("+STR$(PX%)+","+STR$(PY%)
+")"
 1100 P%=P%+1
 1110 =0
 1120 :
 1130 DEFFNFCB(C$)
 1140 =STRING$(B%*(B%<0),"(")+C$+STRING$
(-B%*(B%>0),")")
 1150 :
 1160 DEF PROCFCC
 1170 IF INSTR(K$(C%),PX$) E$=E$+PX$
 1180 IF INSTR(K$(C%+1),PX$) PROCFCD(1)
 1190 ENDPROC
 1200 :
 1210 DEF PROCFCD(A%)
 1220 F$=LEFT$(F$,P%-1)+MID$(F$,P%+A%)
 1230 P%=P%-1
 1240 ENDPROC
 1250 :
 1260 :
 2000 DEF PROCSREP:VDU7:ENDPROC
 2010 DEF PROCSUM:VDU7:ENDPROC
 5000 DEF PROCCLS
 5010 VDU26:CLS:FORZ%=1TO2:VDU130,157,13
2,141
 5020 PRINTSPC(12);"DATASHEET":NEXT
 5030 ENDPROC
 5040 :
 5050 DEF PROCSCREEN
 5060 VDU28,0,19,39,4,12,26,23,1,0;0;0;0
;
 5070 PROCSCR(SX%,SY%,SX%+CN%,SY%+14,CHR
$130,CHR$135)
 5080 OX%=X%:OY%=Y%:PROCPOS
 5090 ENDPROC
 5100 :
 5110 DEF PROCSCR(JX%,JY%,KX%,KY%,O$,N$)
 5120 PRINT'TAB(0,4);SPC(CW%);
 5130 FOR V%=JX% TO KX%
 5140 IF V%>=WA% IFV%<=WC% PRINTO$;A$(V%
);N$;T$(0,V%);:ELSEPRINT" ";A$(V%);" ";T
$(0,V%);
 5150 NEXT:FOR W%=JY% TO KY%
 5160 PRINT'TAB(0,W%-JY%+5);T$(1,W%);
 5170 IF W%>=WB% IF W%<=WD% PRINTO$;A$(W
%);:ELSEPRINT" ";A$(W%);
 5180 FOR V%=JX% TO KX%
 5190 PRINT,D(V%,W%);
 5200 NEXT:IF N$>" " PRINTTAB(CW%,W%-JY%
+5);N$;
 5210 NEXT:PRINT
 5220 ENDPROC
 5230 :
 5240 DEF PROCSCRVAL(V)
 5250 D(X%,Y%)=V:F$(X%,Y%)="":E$(X%,Y%)=
""
 5260 VDU31,(X%-SX%+1)*CW%,Y%-SY%+5
 5270 @%=AT1%:PRINTRIGHT$(S$+STR$(D(X%,Y
%)),CW%);
 5280 PROCPOS:@%=AT%
 5290 ENDPROC
 5300 :
 5310 DEF PROCPOS
 5320 VDU23,1,0;0;0;0;28,0,3,39,2,12,26,
31,0,2,134:Z%=131:PRINT"Formula:";
 5330 IF F$(X%,Y%)>"" PRINTLEFT$(F$(X%,Y
%),31);CHR$134;MID$(F$(X%,Y%),32,39);:Z%
=134
 5340 PROCPOS2(OX%,OY%,135,32)
 5350 PROCPOS2(X%,Y%,Z%,135)
 5360 VDU31,0,20,134,23,1,1,0,0;0;0;
 5370 ENDPROC
 5380 :
 5390 DEF PROCPOS2(A%,B%,C%,D%)
 5400 VDU31,(A%-SX%+1)*CW%,B%-SY%+5,C%
 5410 IF A%<MX% PRINTSTRING$(CW%-1,CHR$9
);CHR$D%
 5420 ENDPROC
 5430 :
 5440 DEF PROCMENUBAR
 5450 PRINTTAB(0,20);CHR$134;"Formula,Go
to,Replicate,Evaluate,Titles";
 5460 PRINTTAB(1,21);CHR$134;"Window(";W
1$;",";W2$;"),Automove(";AM$(AM%);"),Siz
e(";MS$;")";
 5470 ENDPROC
 5480 :
 5490 DEF FNPAIR(C$)
```

```
 5500 PX$=LEFT$(C$,1):PY$=MID$(C$,2,1)
 5510 IFPX$="" ORPY$="" THEN=1
 5520 PX%=0:PY%=0
 5530 FOR Z%=1TOMX%:IF PX$=A$(Z%) PX%=Z%
 5540 NEXT:IF PX%=0 THEN=2
 5550 FOR Z%=1TOMY%:IF PY$=A$(Z%) PY%=Z%
 5560 NEXT:IF PY%=0 THEN=2
 5570 =0
 5580 :
 5590 DEF FNIN(C$,C%)
 5600 LOCAL A%,X%,Y%:PROCP(200):*FX21,0
 5610 PRINTTAB(0,22)CHR$131;C$;" ? ";
 5620 !BLK%=BUF%:BLK%?2=C%:BLK%?3=32:BLK
%?4=127
 5630 A%=0:X%=(BLK%MOD256):Y%=(BLK%DIV25
6)
 5640 CALL&FFF1
 5650 VDU28,0,24,39,22,12,26
 5660 =$BUF%
 5670 :
 5680 DEF PROCSTAR
 5690 PROCCLS:PRINT'CHR$131;C$
 5700 OSCLI(C$):PROCK
 5710 ENDPROC
 5720 :
 5730 DEF PROCP(A%)
 5740 FOR Z%=1 TO A%:NEXT
 5750 ENDPROC
 5760 :
 5770 DEF PROCK
 5780 PRINT'CHR$131"Press any key";:Z%=G
ET
 5790 ENDPROC
 5800 :
 5810 :
 5820 DEF FNCW
 5830 PROCCLS:PRINTTAB(3,8)CHR$131"Selec
t Column Width (6-10)";:INPUT" ? "C$
 5840 Z%=VAL(C$):IFZ%=0 Z%=CW%
 5850 IFZ%>10 THEN=10
 5860 IFZ%<6 THEN=6
 5870 =Z%
 5880 :
 5890 DEF PROCSETUP
 5900 LOCAL V,PX%,PY%,RX%,RY%,RX2%,RY2%,
RA%,RB%,RC%,RD%,TX%,TY%,OX%,OY%,OSX%,OSY
%
 5910 MS$="@@":MA%=27:MX%=MA%:MY%=MA%
 5920 E$=STRING$(255,"*"):S$=STRING$(10,
" ")
 5930 C$=STRING$(70,"*"):F$=C$:PROCUSER
 5940 CW%=FNCW:PROCCWSET:READO$,N$,PX$,P
Y$
 5950 DIM D(MX%,MY%),F$(MX%,MY%),E$(MX%,
MY%),T$(1,MA%),A$(MA%),AM$(2),K$(3),M$(6
),BLK% 6,BUF% 80
 5960 READ AM$(0),AM$(1),AM$(2),C$,K$(0)
,K$(3)
 5970 K$(2)=K$(0)+C$:K$(1)=K$(3)+C$
 5980 FOR Y%=0 TO 1:FOR Z%= 1TO MX%
 5990 T$(Y%,Z%)=LEFT$(S$,CW%-3+Y%):NEXT,
 6000 FOR Z%=1 TO 27
 6010 A$(Z%)=CHR$(64+Z%MOD27)
 6020 NEXT:*FX4,2
 6030 FOR Z%=0 TO 15:READC$
 6040 OSCLI("KEY"+STR$(Z%)+" |?|?|?|?|?|
?|?"+C$+"|M"):NEXT
 6050 FOR Z%=0 TO 6:READM$(Z%):NEXT:@%=A
T%
 6060 ENDPROC
 6070 :
 6080 DEF PROCCWSET
 6090 CN%=7-(CW%+1)DIV2:PC%=COLS%DIVCW%-
1
 6100 VS=-(10^(CW%-5))+0.01:VM=-VS*10+0.
09
 6110 AT%=&00020200+CW%:AT1%=&01020200+C
W%
 6120 PROCWNSET
 6130 ENDPROC
 6140 :
 6150 DEF PROCWNSET
 6160 W1$="AA":WA%=1:WB%=1
 6170 W2$=MS$:WC%=MX%:WD%=MY%
 6180 ENDPROC
 6190 :
 6200 DATA*,*,*,*,*," ",],v,"QWERTYUIOPASD
FGHJKLZXCVBNM$%&!?,"
 6210 DATA"+-*/.()<>=0123456789","qwerty
uiopasdfghjklzxcvbnm ~#':;\_`{|}""@"
 6220 DATAF,R,G,E,W,A,S,/,:,M,OLD|M,Q,<,
>,?,*
 6230 DATAESC - Edit Sheet,1 - Save Shee
t,2 - Load Sheet
 6240 DATA3 - Print Whole Sheet,4 - Prin
t Window
 6250 DATA5 - Spool Window,6 - End Progr
am
 6260 :
 6270 DEF PROCUSER
 6280 REM**Default Column Width**
 6290 CW%=8
 6300 REM**Insert Printer Codes Here**
 6310 VDU2,3
 6320 REM**No.of Columns across Paper**
 6330 COLS%=72
 6340 ENDPROC
 6350 :
```

The Hidden Persuaders (Again)

Part two of David Holton's article starts with an apology.

I hope you didn't mind Listing 2 of *The Hidden Persuaders* last month. The Beebug front desk has had to pacify scores of angry readers carrying bricks, pick-handles and these funny milk bottles with little damp scarves round their necks, all demanding my address. To calm the punters down, Mr. Williams asked me to apologise and tell you all how I managed to slip it past him, and I found his hands round my throat so persuasive that I could but croak my agreement. Please don't keep asking where I am, though - Salman and I are having a great time, you'll never find us.

Listing 1 was quite genuine, and that list of names really is exactly where I said it was. The name Roger really is at the end of the Basic ROM, too. Listing two was the devious bit. Obviously, the real routine was in those data statements at the beginning. All you have to do to pull this stunt on some poor unsuspecting victim is to write your routine in assembly language, assemble it somewhere handy and use Basic to peek the locations and print them out as hex numbers. You then type them back in as EQUB statements.

MEMORY LANE

Everything in a computer's memory is stored as numbers; it's only the context of those numbers which tells the machine whether you want them to mean an instruction, a byte of data, a character, a Basic keyword or whatever. The machine neither knows nor cares what method you used to put those numbers into memory, it simply acts on them.

&043C is the address of the variable *O%*, which was set to equal P% at the start, so CALL !&043C simply calls the code. I feared that even in the form of numbers, the data "APRIL FOOL!" might be spotted as you typed it in - all those bytes suspiciously in the same range - and a memory-editor would have blown it instantly. Therefore a handy stretch of the MOS ROM was chosen and another Basic routine used to print out the results of EORing the string "APRIL FOOL!" with the bytes of that area of ROM. Those results became the data, and the machine-code routine which you typed in (or not, as the case may be) EORed them again with the same stretch of the ROM to restore the text.

The second part, which was in assembly language, was genuine to the extent that it printed out an error message which exists in the MOS Rom at &E530. The label *acon*, however, was indeed a *con*, as I'm sure you realised; all the code connected with it was window-dressing. &FD34 is in JIM and poking it does nothing unless you are accessing extra memory in cartridges or some such unusual trick; most of JIM is reserved for code to be used by such devices. Doing CALL code, of course, called that second routine.

Next month, Twisted Listings presents a virus with a difference - it doesn't affect the computer, but the user grows another head.

Tabform

Willem van Schaik presents his flexible table formatter.

As soon as you use your word processor for a little more then just writing a letter, there will come a time when you have to include some information which is best presented as a table. Sometimes you go ahead and create it by editing, moving, copying etc., but often you will take the easy track (at least I do) and avoid the issue by describing the information in plain text. The reason is that, while creating a table is not such a big deal, as soon as you start improving and changing the text it is a lot of work to keep the table formatted.

After 10 years of manual table formatting I finally decided that the BBC should do this hard work for me, and to my surprise writing a program for it appeared to be rather simple. I opted for the method where you store in a separate file the text to be put in the table together with the formatting information. A table formatting program will then read this file, process it and write the formatted table into another file.

The formatting process can best be compared with the way a word processor formats lines. Words that do not fit completely on the line are shifted to the next, while incomplete lines are filled with text from the next line. For our table things work similarly, but in this case everything has to take place within a box. This means that besides moving words we also have to add spaces to fill the box. Further, the box within a row with the most lines of text determines the vertical size of all boxes in that row.

After formatting you can switch back to your word processor document to include the file with the formatted table, loading it in using whatever *insert text* function you have. If you want to make modifications, don't do it in the final table but go through the process of saving your document, then load the table definition file, modify it, run the TABFORM program, load the document again, delete the previous version of the table and include the new one. This looks like going all round the houses, but in the end it will save time, because in practice the minor changes you think of in the beginning always end up into becoming complete re-designs of your table.

The table definition file is created in your word processor and saved out as plain text; its contents are quite simple. First it must define the sizes of the columns, which is done with a ruler. Then the text to be put into the rows and columns of the table has to be separated to provide the program with an indication about when to start a new box.

RULER LINE

The ruler, defining the column widths, will normally be the first line of the definition file, although this is not obligatory. It consists of a line with hyphens, where the sizes of the columns are indicated with either a plus (+) or a bar (|). Using a plus means that you want the boxes in the resulting table to be separated with lines, while using bars in the ruler will indicate that only spaces and blank lines are to be used for separation.

In the ruler you can put spaces around the plus or the bar to define how much space

you want to put between the text and the border lines. In the example given here we use one space between text and lines. To keep the program simple only the number of spaces after the first plus or bar of the ruler will be checked and this value will be used for all the columns. However, it will improve readability when you put the spaces equally around all pluses or bars. In addition you can make the table indented from the left margin by starting the ruler with as many spaces as the table must be shifted to the right.

TABLE TEXT

The main principle of the format to be used for row/column text definitions is the use of an 'equals' sign (=) or a hyphen (-) in the *first* column of the definition file. We will call this the *format character*. When TABFORM encounters a hyphen at the beginning of a line, a new box will be started. The text following the *format character* will be put in a new box; however, when all boxes in a row are used a new line of boxes will be started and the text will go to the first box of that new row. When you want to force the program to start a new row of boxes you only need to use the 'equals' sign instead of the hyphen.

This way of defining boxes permits a high level of flexibility when entering text. For example, after the hyphen, you are free to enter the text immediately on the same line, or you can start on a new line. If you use more spaces (or new-lines) between two words, they will be compressed into one. Therefore blank lines in your text or starting on a new line will be filtered out.

To show all the possibilities of the scanning function of TABFORM it is best to study the following example.

```
+ ——— + ——— + ——————— +
= PORT
- CONNECTED TO
- FUNCTION
= disc
- 8271
= printer
- 74LS244 + 6522

- parallel printer port
- user I/O
- 6522
-        multi    function          parallel port
=
1MHz bus
-74LS244 74LS245
-buffered address-and data-bus
= tube
- -
- second processor
```

The table definition file is not very consistently created, but it illustrates all features of the program and produces the following table.

PORT	CONNECTED TO	FUNCTION
disc	8271	
printer	74LS244 + 6522	parallel printer port
user I/O	6522	multi function parallel port
1MHz bus	74LS244 74LS245	buffered address- and data-bus
tube	-	second processor

Formatted table from definition above

So, let's walk through it from beginning to end:

= **PORT** is the standard notation, in this case the text is on the same line after the format character.

- **CONNECTED TO** illustrates that when a single word is too long for the column it will be cut in two without hyphenation.

= **disc** forces the start of a new row of boxes.

= **printer** also starts a new row, however, in this case it results in an empty last box on the second row.

the line before - **parallel** is blank, which has no further consequences.

- **user I/O** is an example of starting a new row without an equals sign in the first column.

- **multi** shows multiple spaces are no problem, but will be filtered out.

1Mhz bus and on: here the text is put on new lines; to avoid confusion it is best to select either the format above or this way for our definitions and not to use a mix.

- **buffered** illustrates how formatting is done within a box.

- - shows how to put a hyphen (or equals-sign) as text in a box; as long as the format-character is not at the start of the line it will be considered to be standard text.

To learn all the different possibilities of TABFORM it is easiest to start with the example given and check that you get the same table. Then step by step do some experiments like changing the pluses in the ruler into bars, changing column widths, indenting the ruler, etc. to see how that alters the resulting table. The table is displayed on the screen by TABFORM to show you what is in the output file. You can always check the contents of an output file with *TYPE.

WORD PROCESSORS

TABFORM was written and tested in conjunction with View, although there is no reason why it should not work with other word processors. To facilitate this, the program checks first which <CR> and/or <LF> combination is used in the definition file and it will use the same line separator for the output file. Using View you can both *load* and *read* the file with the formatted table. As the file is plain ASCII there should be no problems with any standard word processors.

A problem that could exist is when your word processor starts a file with some type of a header-record, even when exporting pure ASCII. If this is the case you can remove the REM from line 1550. By doing this everything before the first plus or bar in the ruler will be written to the output file. In case a plus or bar character is part of your word processor's header record, you must select another character and change the program accordingly.

And that's it. I hope that this will save you some time and make your documents easier to produce. May all your tabulations be happy ones.

```
  10 REM TABFORM Table Formatter
  20 REM Version B 1.4
  30 REM Author  Willem van Schaik
  40 REM BEEBUG  May 1992
  50 REM Program subject to copyright
  60 :
 100 ON ERROR CLOSE#0:REPORT:PRINT " at
line ";ERL:END
 110 DIM eol%(2)
 120 DIM coltext$(32)
```

```
  130 DIM colsize%(32)
  140 :
  150 VDU 15
  160 PROCopen
  170 PROCanaleol
  180 PROCruler
  190 PROCreadcol(0)
  200 PROCwritedash
  210 REPEAT
  220 PROCreadrow
  230 REPEAT
  240 PROCwriteline
  250 UNTIL rowfini%
  260 PROCwritedash
  270 UNTIL eof%
  280 PROCclose
  290 END
  300 :
  310 :
 1000 DEF PROCopen
 1010 PRINT
 1020 REPEAT
 1030 INPUT "Input file with table defin
ition: " rdfile$
 1040 rdfile%=OPENIN(rdfile$)
 1050 UNTIL rdfile%<>0
 1060 INPUT "Output file with formatted
table: " wrfile$
 1070 wrfile%=OPENOUT(wrfile$)
 1080 PRINT
 1090 ENDPROC
 1100 :
 1200 DEF PROCclose
 1210 CLOSE#rdfile%
 1220 CLOSE#wrfile%
 1230 PRINT
 1240 ENDPROC
 1250 :
 1300 DEF PROCanaleol
 1310 REPEAT
 1320 byte%=BGET#rdfile%
 1330 UNTIL byte%=10 OR byte%=13
 1340 eol%(0)=1:eol%(1)=byte%
 1350 byte%=BGET#rdfile%
 1360 IF byte%<>eol%(1) AND (byte%=10 OR
 byte%=13) THEN eol%(0)=2:eol%(2)=byte%
 1370 PTR#rdfile%=0
 1380 ENDPROC
 1390 :
 1500 DEF PROCruler
 1510 lead%=-1
 1520 REPEAT
 1530 byte%=BGET#rdfile%
 1540 lead%=lead%+1
 1550 REM IF byte%<>ASC("+") AND byte%<>
ASC("|") THEN BPUT#wrfile%,byte%
 1560 IF byte%=10 OR byte%=13 THEN lead%
=-1
 1570 UNTIL byte%=ASC("+") OR byte%=ASC(
"|")
 1580 IF byte%=ASC("+") THEN lines%=TRUE
 ELSE lines%=FALSE
 1590 nrcols%=0
 1600 sep%=-1
 1610 REPEAT
 1620 size%=0
 1630 REPEAT
 1640 byte%=BGET#rdfile%
 1650 IF sep%=-1 AND byte%<>ASC(" ") THE
N sep%=size%
 1660 size%=size%+1
 1670 UNTIL byte%=ASC("+") OR byte%=ASC(
"|")
 1680 nrcols%=nrcols%+1
 1690 colsize%(nrcols%)=size%-1
 1700 byte%=BGET#rdfile%
 1710 PTR#rdfile%=PTR#rdfile%-1
 1720 UNTIL byte%=eol%(1)
 1730 ENDPROC
 1740 :
 1800 DEF PROCreadrow
 1810 c%=1
 1820 REPEAT
 1830 PROCreadcol(c%)
 1840 c%=c%+1
 1850 UNTIL eof% OR c%>nrcols% OR byte%=
ASC("=")
 1860 ENDPROC
```

```
 1870 :
 1900 DEF PROCreadcol(c%)
 1910 coltext$(c%)=""
 1920 REPEAT
 1930 prev%=byte%
 1940 byte%=BGET#rdfile%
 1950 IF EOF#rdfile% THEN eof%=TRUE ELSE
 eof%=FALSE
 1960 IF byte%<>10 AND byte%<>13 THEN co
ltext$(c%)=coltext$(c%)+CHR$(byte%) ELSE
 coltext$(c%)=coltext$(c%)+" "
 1970 IF coltext$(c%)=" " THEN coltext$(
c%)=""
 1980 IF LEN(coltext$(c%))>2 AND RIGHT$(
coltext$(c%),2)="  " THEN coltext$(c%)=L
EFT$(coltext$(c%),LEN(coltext$(c%))-1)
 1990 UNTIL eof% OR ((prev%=10 OR prev%=
13) AND (byte%=ASC("-") OR byte%=ASC("="
)))
 2000 coltext$(c%)=LEFT$(coltext$(c%),LE
N(coltext$(c%))-1)
 2010 ENDPROC
 2020 :
 2100 DEF PROCwritedash
 2110 IF NOT lines% THEN PROCeol:ENDPROC
 2120 IF lead%>0 THEN PROCwritelead
 2130 BPUT#wrfile%,ASC("+"):VDU ASC("+")
 2140 FOR c%=1 TO nrcols%
 2150 FOR i%=1 TO colsize%(c%)
 2160 BPUT#wrfile%,ASC("-"):VDU ASC("-")
 2170 NEXT i%
 2180 BPUT#wrfile%,ASC("+"):VDU ASC("+")
 2190 NEXT c%
 2200 PROCeol
 2210 ENDPROC
 2220 :
 2300 DEF PROCwriteline
 2310 rowfini%=TRUE
 2320 IF lead%>0 THEN PROCwritelead
 2330 IF lines% THEN BPUT#wrfile%,ASC("|
"):VDU ASC("|") ELSE BPUT#wrfile%,ASC("
"):VDU ASC(" ")
 2340 FOR c%=1 TO nrcols%
 2350 IF sep%>0 THEN FOR i%=1 TO sep%:BP
UT#wrfile%,ASC(" "):VDU ASC(" "):NEXT i%
 2360 s%=colsize%(c%)+2-2*sep%
 2370 REPEAT
 2380 s%=s%-1
 2390 UNTIL MID$(coltext$(c%),s%,1)=" "
OR s%=1
 2400 IF s%=1 s%=colsize%(c%)-2*sep%
 2410 t$=LEFT$(coltext$(c%),s%)
 2420 t$=t$+STRING$(255-LEN(t$)," ")
 2430 t$=LEFT$(t$,colsize%(c%)-2*sep%)
 2440 PROCwritetext(t$)
 2450 coltext$(c%)=MID$(coltext$(c%),s%+
1)
 2460 IF sep%>0 THEN FOR i%=1 TO sep%:BP
UT#wrfile%,ASC(" "):VDU ASC(" "):NEXT i%
 2470 IF lines% THEN BPUT#wrfile%,ASC("|
"):VDU ASC("|") ELSE BPUT#wrfile%,ASC("
"):VDU ASC(" ")
 2480 IF LEN(coltext$(c%))>0 THEN rowfin
i%=FALSE
 2490 NEXT c%
 2500 PROCeol
 2510 ENDPROC
 2520 :
 2600 DEF PROCwritelead
 2610 FOR i%=1 TO lead%
 2620 BPUT# wrfile%,ASC(" "):VDU ASC(" "
)
 2630 NEXT i%
 2640 ENDPROC
 2650 :
 2700 DEF PROCwritetext(s$)
 2710 FOR l%=1 TO LEN(s$)
 2720 BPUT#wrfile%,ASC(MID$(s$,l%,1)):VD
U ASC(MID$(s$,l%,1))
 2730 NEXT l%
 2740 ENDPROC
 2750 :
 2800 DEF PROCeol
 2810 FOR i%=1 TO eol%(0)
 2820 BPUT#wrfile%,eol%(i%):PRINT
 2830 NEXT i%
 2840 ENDPROC
 2850 :
```

Mode 0 Screen Dumps

David Stevens gives us two screendump programs especially for lovers of round circles.

Hard copy of a mode 0 graphics screen is often useful, and some programming articles assume that you have a printer dump. If you haven't and your printer is compatible with a 9-pin Epson, one of the programs listed here programs should meet your needs.

They work best in mode 0, but will also cope with mode 4. They are of little use in the other graphics modes, because any non-background pixel is represented by a dot on the paper.

To get close to the proper ratio between width and height, both dumps compensate for the different shapes of screen, printer characters and graphics bytes. Each printer byte is made up of bits 7 to 1 corresponding to screen pixels and a computed bit 0.

The machine code for each dump is just over one memory page long. The source programs, *Sdump1* and *Sdump2* assemble the code at &DD00 if run on a Master, or at &900 on a Model B. You can assemble elsewhere by substituting:

```
1030 code%=&<address in hex>
```

in either program. On the Master with the Tube active, &DD00 is okay if HiBasic is not used.

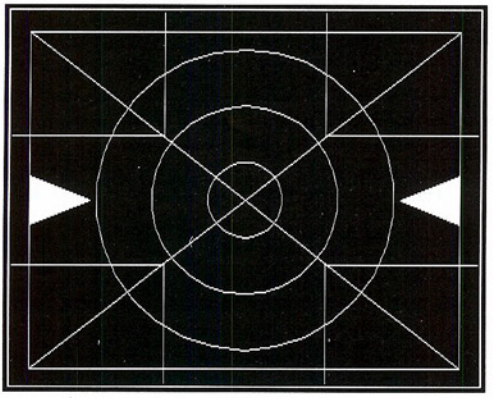
If your printer does not accept the code to set line spacing to n/216" (ESC "3" - check your manual) you will need to substitute:

```
1390 EQUB 27:EQUB 65:EQUB 8
```

in either program.

THE STANDARD DUMP

Type in the first program and save it as *Sdump1*. When you run it type Y in response to the prompt so that the code is saved as *dump1*. Don't change the file name at this stage or the demo program won't work.



Screendump testcard on screen

Now type in the third program and save it as *Dmpdemo*. Run it and choose *dump1*. A diagram will be drawn on the screen and dumped to the printer. The print is about 5" wide and 4" high, and the whole of the screen is reproduced.

THE SIDEWAYS DUMP

Sometimes a larger print is useful. The second program creates a dump with the screen rotated through 90∞ to produce a dump that fills a whole page. Type this in and save it as *Sdump2*. Run it and save the machine code as *dump2*. Now chain *Dmpdemo* and this time choose *dump2*.

Printing takes about four times as long as *dump1*. It reads the pixels in each column, starting at the top right of the diagram, and builds them into lines for the printer. The resulting print is 8" wide and nearly 10" long. It will just fit on A4 or a sheet of 11" continuous paper.

You will notice that the demo diagram does not quite cover the whole screen.

Paper size limits the width that can be dumped. Height is limited because the printer cannot take more than 960 bytes per line in double-density mode. Any screen image to be dumped must therefore fall within the area covered by the demo diagram.

The dump omits a slice 2 characters deep at the top of the screen. If you need the slice omitted to be at the bottom of the screen substitute this line in *Sdump2*:

```
1090 LDA #&FF:STA ylo:LDA #3:STA
     yhi:LDX #0
```

(the printer will ignore any surplus bytes from the bottom of the screen).

USING THE PROGRAMS

If you put the machine code in the same directory as the Basic program that will call it, the program should include the statement *RUN dump1 or *RUN dump2; if the code is in your library directory, the word RUN can of course be omitted.

Either program could be modified to dump only a selected part of the screen. If you try this, bear in mind that the number of bytes per line sent to the printer must match the VDU sequence in line 1400 (of either program). This is VDU27,76,n1,n2, where n1 + n2*&100 gives the minimum number of bytes the printer must receive for each line.

Listing 1

```
   10 REM Program Sdump1
   20 REM Version B 2.0
   30 REM Author  David Stevens
   40 REM BEEBUG  May 1992
   50 REM Program subject to copyright
   60 :
  100 ON ERROR PROCer
  110 MODE0:VDU19,0,4;0;19,1,3;0;
  120 PROCassem:PROCsave
  130 END
  140 :
 1000 DEF PROCassem
 1010 xlo=&70:xhi=&71:ylo=&72:yhi=&73:pi
x=&74:rylo=&75:ryhi=&76:offset=&77:byte=
&78
 1020 osbyte=&FFF4:osword=&FFF1:oswrch=&
FFEE
 1030 A%=0:X%=1:mos%=((USRosbyte)AND&F00
)/&100:IFmos%>2 code%=&DD00 ELSEcode%=&9
00
 1040 FOR pass=0 TO 2 STEP 2:P%=code%
 1050 [OPT pass
 1060 .dump LDA #2:JSR oswrch:LDX #0
 1070 .sloop LDA svdu,X:JSR prt:INX:CPX
#6:BNE sloop
 1080 LDA #8:STA rylo:LDA #4:STA ryhi
 1090 .newline LDX #0:STX xlo:STX xhi
 1100 .lloop LDA lvdu,X:JSR prt:INX:CPX
#4:BNE lloop
 1110 .newbyte LDA #0:STA byte:STA offse
t:JMP esc
 1120 .newbit JSR rdpix:JSR addpix
 1130 CLC:LDA offset:ADC #4:STA offset:C
MP #28:BNE newbit
 1140 JSR byte0
 1150 CLC:LDA xlo:ADC #2:STA xlo:LDA xhi
:ADC #0:STA xhi:CMP #5:BNE newbyte
 1160 LDA #&D:JSR prt
 1170 SEC:LDA rylo:SBC #28:STA rylo:LDA
ryhi:SBC #0:STA ryhi:CMP #&FF:BEQ end
 1180 JMP newline
 1190 .end LDA #27:JSR prt:LDA #64:JSR p
rt
 1200 LDA #3:JSR oswrch:RTS
 1210 :
 1220 .rdpix SEC:LDA rylo:SBC offset:STA
 ylo:LDA ryhi:SBC #0:STA yhi
 1230 LDA #9:LDX #xlo:LDY #0:JSR osword:
JSR chkpix:RTS
 1240 .byte0 LDA xlo:PHA:LDA xhi:PHA:LDA
 pix:BEQ done
 1250 JSR rdpix:BEQ done
 1260 CLC:LDA xlo:ADC #2:STA xlo:LDA xhi
:ADC #0:STA xhi:JSR rdpix:BNE done
 1270 SEC:LDA xlo:SBC #4:STA xlo:LDA xhi
:SBC #0:STA xhi:JSR rdpix
 1280 .done PLA:STA xhi:PLA:STA xlo:JSR
addpix:JSR prt:RTS
 1290 .addpix LDA byte:ASL A:CLC:ADC pix
:STA byte:RTS
 1300 .chkpix LDA pix:CMP #&FF:BNE pixok
 1310 LDA #0:STA pix
 1320 .pixok RTS
 1330 .prt PHA:LDA #1:JSR oswrch:PLA:JSR
 oswrch:RTS
```

```
 1340 .esc LDA #129:LDX #&8F:LDY #&FF:JS
R osbyte:CPX #&FF:BEQ escape
 1350 JMP newbit
 1360 .escape LDA #126:JSR osbyte:JMP en
d
 1370 :
 1380 .svdu EQUB 27:EQUB 108:EQUB 14
 1390 EQUB 27:EQUB 51:EQUB 25
 1400 .lvdu EQUB 27:EQUB 76:EQUB &80:EQU
B 2
 1410 ]:NEXT:ENDPROC
 1420 :
 1430 DEFPROCsave
 1440 f$="dump1":PRINTTAB(2,5)"Do you wi
sh to save the code as ""dump1""? (Y/N)
";:IFNOTFNyes PRINTTAB(43,5)"No    ":INPU
TTAB(2,7)"File name: "f$
 1450 OSCLI"SAVE "+f$+" "+STR$~code%+" "
+STR$~P%:PRINTTAB(2,9)"Code runs from "S
TR$~code%" to "STR$~P%:ENDPROC
 1460 :
 1470 DEFFNyes
 1480 REPEATq%=INSTR("YN",CHR$(GETAND&DF
)):UNTILq%:=q%-2
 1490 :
 1500 DEFPROCer
 1510 ONERRORROFF:PRINT:REPORT:PRINT" at
Line ";ERL:END
```

Listing 2

```
   10 REM Program Sdump2
   20 REM Version B 2.0
   30 REM Author  David Stevens
   40 REM BEEBUG  May 1992
   50 REM Program subject to copyright
   60 :
  100 ONERRORPROCer
  110 MODE0:VDU19,0,4;0;19,1,3;0;
  120 PROCassem:PROCsave
  130 END
  140 :
 1000 DEFPROCassem
 1010 xlo=&70:xhi=&71:ylo=&72:yhi=&73:pi
x=&74:rxlo=&75:rxhi=&76:offset=&77:byte=
&78
 1020 osbyte=&FFF4:osword=&FFF1:oswrch=&
FFEE
 1030 A%=0:X%=1:mos%=((USRosbyte)AND&F00
)/&100:IFmos%>2 code%=&DD00 ELSEcode%=&9
00
 1040 FORpass=0TO2STEP2:P%=code%
 1050 [OPT pass
 1060 .dump LDA #2:JSR oswrch:LDX #0
 1070 .sloop LDA svdu,X:JSR prt:INX:CPX
#6:BNE sloop
 1080 LDA #&C2:STA rxlo:LDA #4:STA rxhi
 1090 .newline LDA #&BF:STA ylo:LDA #3:S
TA yhi:LDX #0
 1100 .lloop LDA lvdu,X:JSR prt:INX:CPX
#4:BNE lloop
 1110 .newbyte LDA #0:STA byte:STA offse
t:JMP esc
 1120 .newbit JSR rdpix:JSR addpix
 1130 CLC:LDA offset:ADC #2:STA offset:C
MP #14:BNE newbit
 1140 JSR byte0
 1150 SEC:LDA ylo:SBC #1:STA ylo:LDA yhi
:SBC #0:STA yhi:CMP #&FF:BNE newbyte
 1160 LDA #&D:JSR prt
 1170 SEC:LDA rxlo:SBC #14:STA rxlo:LDA
rxhi:SBC #0:STA rxhi:BNE nextline
 1180 LDA rxlo:CMP #42:BEQ end
 1190 .nextline JMP newline
 1200 .end LDA #27:JSR prt:LDA #64:JSR p
rt
 1210 LDA #3:JSR oswrch:RTS
 1220 :
 1230 .rdpix SEC:LDA rxlo:SBC offset:STA
 xlo:LDA rxhi:SBC #0:STA xhi
 1240 LDA #9:LDX #xlo:LDY #0:JSR osword:
JSR chkpix:RTS
 1250 .byte0 LDA ylo:PHA:LDA yhi:PHA:LDA
 pix:BEQ done
 1260 JSR rdpix:BEQ done
 1270 CLC:LDA ylo:ADC #1:STA ylo:LDA yhi
:ADC #0:STA yhi:JSR rdpix:BNE done
 1280 SEC:LDA ylo:SBC #2:STA ylo:LDA yhi
:SBC #0:STA yhi:JSR rdpix
 1290 .done PLA:STA yhi:PLA:STA ylo:JSR
addpix:JSR prt:RTS
 1300 .addpix LDA byte:ASL A:CLC:ADC pix
:STA byte:RTS
 1310 .chkpix LDA pix:CMP #&FF:BNE pixok
 1320 LDA #0:STA pix
 1330 .pixok RTS
 1340 .prt PHA:LDA #1:JSR oswrch:PLA:JSR
 oswrch:RTS
 1350 .esc LDA #129:LDX #&8F:LDY #&FF:JS
R osbyte:CPX #&FF:BEQ escape
 1360 JMP newbit
 1370 .escape LDA #126:JSR osbyte:JMP en
d
```

```
 1380 .svdu EQUB 27:EQUB 108:EQUB 0
 1390 EQUB 27:EQUB 51:EQUB 25
 1400 .lvdu EQUB 27:EQUB 76:EQUB &C0:EQU
B 3
 1410 ]:NEXT:ENDPROC
 1420 :
 1430 DEFPROCsave
 1440 f$="dump2":PRINTTAB(2,5)"Do you wi
sh to save the code as ""dump2""? (Y/N)
";:IFNOTFNyes PRINTTAB(43,5)"No   ":INPU
TTAB(2,7)"File name: "f$
 1450 OSCLI"SAVE "+f$+" "+STR$~code%+" "
+STR$~P%:PRINTTAB(2,9)"Code runs from "S
TR$~code%" to "STR$~P%:ENDPROC
 1460 :
 1470 DEFFNyes
 1480 REPEATq%=INSTR("YN",CHR$(GETAND&DF
)):UNTILq%:=q%-2
 1490 :
 1500 DEFPROCer
 1510 ONERROROFF:PRINT:REPORT:PRINT" at
Line ";ERL:END
```

Listing 3

```
   10 REM Program Dmpdemo
   20 REM Version 1.0
   30 REM Author  David Stevens
   40 REM BEEBUG  May 1992
   50 REM Program subject to copyright
   60 :
  100 ONERROR MODE1:PROCer
  110 MODE1:VDU19,0,4;0;19,3,3;0;
  120 PROCprinter
  130 D=FNdump:MODE0
  140 VDU12,19,0,4;0;19,1,3;0;23;10,32,0
;0;0;
  150 IFD=1 PROCscreen1:t0%=TIME:*RUN du
mp1
  160 IFD=2 PROCscreen2:t0%=TIME:*RUN du
mp2
  170 CLS
  180 VDU7,23;10,103,0;0;0;
  190 t%=(TIME-t0%)/100:PRINT'" time: ";
t%DIV60;" minutes ";t%MOD60;" seconds"''
  200 END
  210 :
 1000 DEFPROCscreen1
 1010 PROCrect(0,0,424,340):PROCrect(856
,0,1279,340):PROCrect(0,684,424,1023):PR
OCrect(856,684,1279,1023)
 1020 MOVE64,64:DRAW1216,64:DRAW1216,960
:DRAW64,960:DRAW64,64:DRAW1216,960:MOVE1
216,64:DRAW64,960
 1030 PROCcircle(100):PROCcircle(250):PR
OCcircle(400)
 1040 MOVE64,yc%+64:MOVE64,yc%-64:PLOT85
,224,yc%:MOVE1216,yc%+64:MOVE1216,yc%-64
:PLOT85,1056,yc%
 1050 ENDPROC
 1060 :
 1070 DEFPROCscreen2
 1080 PROCrect(52,0,424,276):PROCrect(84
6,0,1218,276):PROCrect(52,680,424,956):P
ROCrect(846,680,1218,956)
 1090 MOVE84,32:DRAW1186,32:DRAW1186,922
:DRAW84,922:DRAW84,32:DRAW1186,922:MOVE1
186,32:DRAW84,922
 1100 PROCcircle(100):PROCcircle(250):PR
OCcircle(400)
 1110 MOVE84,yc%+64:MOVE84,yc%-64:PLOT85
,212,yc%:MOVE1186,yc%+64:MOVE1186,yc%-64
:PLOT85,1058,yc%
 1120 ENDPROC
 1130 :
 1140 DEFPROCrect(a%,b%,c%,d%)
 1150 MOVEa%,b%:DRAWa%,d%:DRAWc%,d%:DRAW
c%,b%:DRAWa%,b%:ENDPROC
 1160 :
 1170 DEFPROCcircle(r%)
 1180 n=2*PI/30:yc%=512+34*(D=2):MOVE640
,yc%+r%
 1190 FORj%=1TO30:a=n*j%:c=COS(a):s=SIN(
a):x%=r%*s:y%=r%*c:DRAW640+x%,yc%+y%
 1200 NEXT:ENDPROC
 1210 :
 1220 DEFPROCprinter
 1230 REPEATVDU2,1,32,8,3:IFINKEY(10):IF
ADVAL(-4)<63PRINT''" RETURN when printer
 is on line ";:REPEATUNTILGET=13:CLS
 1240 UNTILADVAL(-4)>62:ENDPROC
 1250 :
 1260 DEFFNdump
 1270 PRINTTAB(2,7)"Which dump?";TAB(10,
9)"1  normal,small";TAB(10,11)"2  sidewa
ys, large";TAB(5,13)"Type 1 or 2 ";:REPE
ATd=GET:UNTILd=49 OR d=50:=d-48
 1280 :
 1290 DEFPROCer
 1300 ONERROROFF
 1310 VDU23;10,103,0;0;0;
 1320 REPORT:PRINT" at Line ";ERL
 1330 END
```

Mr Toad's keyboard BEEP ROM

BEEBUG's warty friend makes your Master keyboard pip and squeak.

I wonder how many of you started computing on the Sinclair Spectrum. Because it only had a membrane keyboard, albeit disguised by rubber or plastic keys, the speaker was made to emit a slight click at every press of a key, and this went a long way towards making up for the sloppy feel of the keyboard. There was a system variable known as PIP which you could poke to lengthen the click into a 'beep', and though I've forgotten much of my Speccy lore, I'll never forget POKE 23609,100. When I got a Master I often made use of a facility on the old *AIDS II ROM* which added a beep to key-presses, despite the excellent feel of the Beeb's keys. I gradually let this drop, though, as the sound on offer wasn't ideal and the cartridge always had to be plugged in.

So, I wrote a new facility: in ROM slot eight I keep my own personal go-faster EPROM, the Toad ROM 90, and from time to time I add more facilities and blow a new chip. The listing published here was extracted from the Toad ROM and tidied up into a short self-contained sideways ROM image. Ideally, this assembly text would be tacked onto the texts of other ROMs, as one doesn't want one's working disc cluttered with several different ROMs, all to be loaded in and assembled to separate slots. One day I hope to write an article for these august pages on how to do just that. In the original, labels for the many routines are in various languages, to cut down on accidental duplication; those for this section are in Latin, and they have been left thus, so as to minimise duplication problems if you do tack this listing onto another one. Labels are more fun in Latin, anyway.

GET TYPING

Type in the listing BEEPROM, saving frequently as you go, and run the program to assemble the code. It will find a free slot, if there is one, and tell you so. *ROMS and *HELP should both show its presence since you don't have to press Ctrl-Break to initialise the image after assembling it from this listing. Enter *BP and you should get prompts inviting you to choose the *tone, volume, pitch* and *duration* of the beep. Once running it cannot be eliminated by a Break, hard or soft, but typing *BP again will toggle it off, then on again, until you turn the machine off.

I hope you have some fun with this, and that you find it pleasant and helpful in use, as I do. This article was written partly with PIP, volume 2, pitch 7, duration 1; and partly with SQUEAK,volume 3, pitch 0, duration 1. You'll soon find your own favourite sound.

HOW IT WORKS

The principle is very simple; once you have typed in the parameters, the ROM grabs a vector, RDCHV. Beebs have a number of these vectors which are RAM copies of the addresses of all the major facilities such as input and output to filing systems, key input and so on. Under the standard setup, they are unnecessary intermediate jumps, early in the routines, to the address of the main part of the routine. Being in RAM, the MOS has to copy them in on power-up - it also resets them on Break The user can alter them to point to their own code, so that every time something passes

through the vector it is diverted to the user's own routine. When that routine has done its bit, it must pass control to the address which was in the vector before it was changed unless, of course, the user's code completely replaces the MOS routine, as it would in the case of a printer buffer for example.

ONE SMALL CATCH

One would think that KEYV at &0228 would be the obvious vector to use here, rather than RDCHV, but if you try it, you'll find that each interrupt reads the average key- press several times, and all the repeats go through KEYV - or so one would surmise from the resulting noise, which is like a cracked doorbell. Therefore we use the RDCHV vector. When anything passes through RDCHV, our routine *.fiat* pushes all the registers, flushes the appropriate sound buffer, sends a seven to OSWRCH, which makes the beep, then pops the registers and hands over to the original destination previously held in the vector.

Any machine-code routine can intercept a vector in this way, but the interesting part is the way it is done with sideways ROMs. Problem: if a ROM like Basic or View is paged in, then our ROM isn't. How, then, can it do anything? On the face of it, the ROM would have to do one of two things; if the essential code were short, as here, it could copy out that code to some uncrowded corner of main memory and point the vector at it, after which the ROM itself would be entirely redundant. If the vital code were long, a short routine copied into main memory would have to page the ROM in and pass control to it. Once finished, the ROM would hand over to the external routine, which would page back in whatever ROM had originally been active and then hand over to the original vector destination.

The latter is what actually happens, but the programmer doesn't need to worry, as Uncle Acorn has thoughtfully provided facilities for it to be done automatically. Instead of pointing the vector directly to the ROM, you leave the ROM number and your routine's address in page &0D, at an address calculated from the vector's number, and then point the vector itself at a similarly calculated address in page &FF. The rest is done by the MOS.

A trip down memory lane with a memory editor, starting at &0DB0, will show that Basic (ROM 9) has directed a lot of these vectors to itself (addresses between &8000 and &BFFF, the third parameter being 9). If the BEEPROM is active, &0DB9 will contain the number of the slot it's in, and &0DB7-8 will point to &815D, which is the address of the routine *.fiat*. Look from &FF00 to &FF4E, and you will see that all the extended vectors contain the same instruction, JSR &FF51. Nevertheless, if you use the wrong address you get a crash, so here we jump to &FF18, as we should. Another tempting short-cut is to pass control on exit to the default contents of the vector - in the case of RDCHV it's &E7BC. This is naughty, since another program may already have grabbed the vector. You should store the actual address contained in the vector, using:

```
LDA vector:STA oldVec
LDA vector+1:STA oldVec+1
```

On exit, your routine should pass control to that address, JMP (oldVec), which will be to that other program, if there is one. If your code deselects itself (as here when you turn the beep off), it should put the stored value back into the vector. Mind you, if extended vectors are used, only one ROM can have any one vector, but

it's just barely possible that some program in main memory might be using it, which is why it's done properly here.

You may notice a rather odd way of changing a vector. Interrupts must be disabled during the changeover, else one might be called when one byte had been changed, but not the other; disastrous if the interrupt code uses that vector. It is good practice to have interrupts off for the shortest possible time, and since here we happen to have all registers free at the crucial points, we can load X and Y with the necessary bytes, *then* disable interrupts and store X and Y in the vector - it's a lot quicker. Notice that &0DB7 to &0DB9 are set up with interrupts still enabled - those data won't be used until the vector points to &FF18.

IT'S A FAIR COP

Not everything is 'legal' in this ROM - the page &0D numbers pointing to our code are put in directly, instead of by reading the start with OSBYTE &A8, and the VDU 7 parameters are likewise poked directly, instead of via OSBYTEs &D3 to &D6; the same goes for the buffer start and end indices which are poked with &FF to emulate OSBYTE &15. This ROM can only run on a Master, unless rewritten to use direct vectoring, and no new Master O.S.'s are ever going to be released now, so not everything needs to be scrupulously kosher any more. Most of the above short-cuts save a few bytes; the last is one byte longer than calling Osbyte but is much quicker in a place where speed matters.

Another way in which I break with convention here is to use workspace inside the ROM image. It's by far the best solution in a case where no-one is ever going to blow it into a chip. You could always change the text and use a bit of main RAM, anyway. Four page zero bytes are used, but they can be corrupted between calls. They're in the filing system scratch space, which is stated to be OK for such uses.

The routines which set up the VDU 7 parameters proved to be great fun. First we need to poke &0263 with 0 for the noise channel or 3 for the default channel. Now the ASCII code of capital P is &50, so S is &53. Lower-case p and s are &70 and &73. Names for the 'bell' and 'noise' channels which begin with S and P respectively would therefore simplify the code a lot. 'PIP' and 'SQUEAK' seem suitable, but the wrong way round, as 'SQUEAK' should be the noise channel.

Let's say we'll break the usual convention and get our routine to use ORA #&20 to force bit 5 of the ASCII, thus forcing lower case rather than masking to a capital by AND #&DF. That way we won't have to do a test and a branch to cope with numerals, in all of which bit 5 is set. Setting *p* and *s* as the limits of acceptable entries will therefore leave us with a byte of between &70 and &73; i.e. 011100xx. EOR #3 will invert the last 2 bits so *s* gives the noise channel 0 and *p* the sound channel 3. The user is not prevented from entering *q* or *r* but they will select sound channels 1 or 2, which is acceptable.

The next move seems to be AND #3 to reset bits 4, 5 and 6 - turn &70 into 0 or &73 into 3 - but in fact EORing those bits with 1 will do it, so EOR #&73 is all that's needed to turn *s* into *0* and *p* into *3*. Incidentally, in practice the MOS seems to reset those higher bits for us automatically, but EOR #&73 is as easy as EOR #3, so why not?

Before emitting each beep, we need to poke the locations used by OSBYTE &15 to flush the appropriate sound channel. X has to hold the number of that channel plus 4. After storing the channel number in the variable *chan*, therefore, we add 4 to the number by forcing bit 3 with ORA #4 and store the byte in *flag*. The first purpose of this variable is to tell the ROM whether it's on or off, but for that it only needs to be zero or non-zero, so it can double as a store for that X value to emulate OSBYTE &15. In fact, as we're not calling the actual OSBYTE, we could use any register, but X is all we have free if Y is to be left undisturbed holding the start of the next prompt.

The next parameter is the volume, to be poked into &0264. We need a negative number of the form 1xxx0000, as any bit set in the last 3 bits produces silence - don't ask me why, but it does. We also need the highest number for the quietest beep. Numerals are all ASCII &3x, so if we set the limits at 1 and 5, we will get a byte of the form 00110xxx, where the part that matters is in the xxx.

Again, EOR #7 will take care of the 'highest=quietest' problem, and four shifts left will leave us with a byte 0xxx0000. Fine, except bit 7 should be set. Now, if we write EOR #&0F instead of #7 before the shifts, we get a byte 00111xxx, which after four ASL As ends up as 1xxx0000, the lowest original number now being the highest. The unwanted bits 5 and 4 of the ASCII numerals just drop off the left-hand edge, like ships that tried to sail round the world in the days when it was still flat.

Next comes the pitch; no great problem, except that a change of just one digit in this value makes no perceptible difference. Trial and error showed that a range of 0 to 9, multiplied by 16 (four ASL As again), is a bit low, but adding the ASCII of the original numeral moves it into about the right range.

The last one is the length of the beep. One to three seemed right, so AND #3 is all that's needed to reset those naughty bits 5 and 4, then we poke it into &0266. You can alter this if you like; change the value in line 840 (to one *above* the desired limit) and substitute AND #&0F in line 840.

Finally, I should point out that if you select SQUEAK at the first prompt, certain pitch parameters give funny results; even silence, in two cases. Read the manual on the effects of the the third parameter of SOUND with channel 0, and you'll see why. It's not worth all the code needed to 'fix' that feature, and it's not a bug, because a bug is a feature you didn't know about, and a feature is a bug which you're too idle to fix.

```
 10 REM Program Mr Toad's Beep Rom
 20 REM Version B 1.0
 30 REM Author  Mr Toad
 40 REM BEEBUG  May 1992
 50 REM Program subject to copyright.
 60 :
100 Z%=TOP+&200
110 lo=&B8:hi=lo+1
120 pLo=lo+2:pHi=lo+3
130 rdchvLo=&0210
140 rdchvHi=&0211
150 oswrch=&FFEE
160 osnewl=&FFE7
170 osasci=&FFE3
180 osrdch=&FFE0
190 :
200 FOR N%=4 TO 6 STEP 2
210 P%=&8000:O%=Z%
220 [ OPT N%
230 BRK:BRK:BRK
```

```
240 JMP hicIncipit
250 EQUB &82
260 EQUB longinquitas MOD &100
270 EQUB &91
280 .nomen
290 EQUS "Mr Toad's Magic Beep ROM"
300 .longinquitas
310 BRK:EQUS "(C) BEEBUG May 1992"
320 :
330 .auxilium
340 LDA (&F2),Y
350 CMP #&0D:BNE nuncDimittis
360 JSR osnewl:LDX #&FF
370 .gyrus
380 INX:LDA nomen,X
390 PHP:JSR osasci:PLP
400 BNE gyrus:JSR osnewl
410 :
420 .nuncDimittis
430 PLY:PLX:PLA:RTS
440 :
450 .noliFugere
460 LDA flag:BEQ nuncDimittis
470 JMP aditus
480 :
490 .requiescat
500 LDX oldVec:LDY oldVec+1
510 SEI:STX rdchvLo
520 STY rdchvHi:CLI:STZ flag
530 JMP iteMissaEst
540 :
550 .hicIncipit
560 PHA:PHX:PHY
570 CMP #&27:BEQ noliFugere
580 CMP #9:BEQ auxilium
590 CMP #4:BNE nuncDimittis
600 LDA (&F2),Y:AND #&DF
610 CMP #ASC"B":BNE nuncDimittis
620 INY:LDA (&F2),Y:AND #&DF
630 CMP #ASC"P":BNE nuncDimittis
640 LDA flag:BNE requiescat
650 :
660 LDA #ASC"p":STA lo
670 LDA #ASC"t":STA hi
680 JSR imprimatio
690 EOR #&73:STA chan
700 ORA #4:STA flag
710 :
720 LDA #ASC"1":STA lo
730 LDA #ASC"6":STA hi
740 JSR etSequ:EOR #&0F
750 ASL A:ASL A
760 ASL A:ASL A:STA vol
770 :
780 LDA #ASC"0":STA lo
790 LDA #1+ASC"9":STA hi
800 JSR etSequ:STA pitch
810 ASL A:ASL A:ASL A:ASL A
820 ADC pitch:STA pitch
830 :
840 LDA #ASC"1":STA lo
850 LDA #ASC"4":STA hi
860 JSR etSequ:AND #3:STA dur
870 :
880 JSR osnewl:JSR osnewl
890 LDA rdchvLo:STA oldVec
900 LDA rdchvHi:STA oldVec+1
910 :
920 .aditus
930 LDA chan:STA &0263
940 LDA vol:STA &0264
950 LDA pitch:STA &0265
960 LDA dur:STA &0266
970 LDA &F4:STA &DB9
980 LDA #fiat DIV &100:STA &DB8
990 LDA #fiat MOD &100:STA &DB7
1000 LDY #&FF:LDX #&18
1010 SEI:STY rdchvHi
1020 STX rdchvLo:CLI
1030 :
1040 .iteMissaEst
1050 PLY:PLX:PLA
1060 CMP #4:BNE exeat:LDA #0
1070 .exeat
1080 RTS
1090 :
1100 .imprimatio
1110 LDA #mandata MOD &100:STA pLo
1120 LDA #mandata DIV &100:STA pHi
1130 LDY #&FF
1140 .etSequ
1150 LDA #7:JSR oswrch
1160 .gyrus
1170 INY:LDA (pLo),Y:PHP:JSR osasci
1180 PLP:BNE gyrus
```

continued on page 58

Public Domain Software

Alan Blundell continues his look at the PD scene with a review of educational software.

Many parents, I suspect, originally bought their BBC micro because it was the same computer as the one their children used at school, with the more or less vague idea that it might give their child some advantage in development. This made sense, as it meant that the child only needed to become familiar with the use of one system and there was the possibility that they could use the same software at home as at school. Because of the traditional links between Acorn and education, a lot of software with educational uses has been produced, covering a wide range of ages and skills. I didn't have that excuse when I first bought my micro, but now have three children aged five and two (twins), so over the past couple of years I have taken a personal interest in educational uses in addition to my general interest, which arose from my wife's work as a remedial class teacher.

Although the Archimedes is becoming more and more common in schools, there are still plenty of BBC micros and Masters in everyday use around the country, and I know from letters which I have received that parents still have an interest in this subject. Although I have no desire to knock more recent micros, good educational software does not necessarily depend on the use of windows, 16 million colours, or the other paraphernalia of modern computing, although these no doubt add to the presentation of software. The most important thing about educational software is its educational content, and this should be as valid for programs for the BBC as it was when they were first written - if a program had educational value 5 years ago, then it still has, national curriculum willing.

Three ranges of educational software will be looked at in this and next month's column, to give a balanced view of what is available in the field. First of all, let's look at the public domain (as distinct from shareware or freeware). Starting with the youngest first, I was particularly pleased to acquire a copy of Jim Stirk's collection of programs for younger children. His *Caroline & Philippa's House* was the first program which really engaged the interest of my two year olds; even better, it kept them happy and occupied (if not quiet) for longer than anything else I could think of! The user interaction consists of nothing more than repeatedly pressing any key on the keyboard, but as the keys are pressed, a house, its rooms and their contents are gradually revealed. The pictures are colourful and well drawn and, from my research sample, generate plenty of interest, excitement and 'discussion'. Jim's collection contains about 20 varied programs, suited to children aged from 2 upwards, many of which may be suitable for use in junior schools.

David Shepherdson (under the alias 'Dragonrider', which has nothing to do with his software) has produced a useful selection of programs aimed at slightly older children. *Chain Letters*, one of his works, is a well produced way of developing spelling skills by trying to outdo the computer in changing one word into another by changing one letter at a time. Three versions are available, for 3, 4 or 5 letter words, and the

difficulty increases in proportion to the number of letters in a word. One potential pitfall of this type of program is a limited vocabulary built into it, but David has neatly got round that problem by storing a range of words in sideways RAM; the 3 letter version I tried seemed to know every 3 letter word I could think of, and a few I couldn't.

Grace Educational Software's *Maths Competition Pack* is a fairly standard mental arithmetic game, using all four basic rules of arithmetic in a race against a clock to answer questions correctly at a chosen difficulty level. Its strengths lie in its real-time clock display, the range of choices available to the supervisor in setting it up, and its competitive element. A number of children can 'play' in turn, and try to reach the top of the league table.

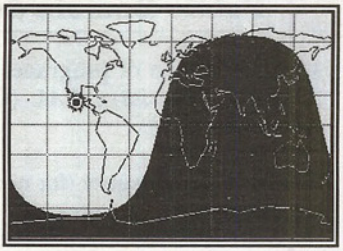
Moving onwards to teenage use, there are several science teaching aids. *Institute of Metals Schools Disc* and *Introduction to the Rutherford Appleton Laboratory* are two examples which particularly appealed to me, from my own general interest. The IoM disc covers materials science at a level suited to A-level physics and chemistry students, and includes subjects such as diffusion and diffraction in solids, and p-n transistor junctions. The RAL disc describes the work done at this famous laboratory, and demonstrates subjects such as molecular structure, Haley's Comet and the Giotto project. Each subject is very well presented and explained, such that the interested amateur (me!) can learn and enjoy. C.A.McGaughey has collected a range of programs which also find a use in the teaching of science, although I should perhaps say that these do not all have a polished user interface. There is also a small collection of programs under the heading *Chemistry CAL* (Computer Aided Learning).

Finally in this category (for now), there is a range of programs such as trigonometric and quadratic equation solvers, other mathematical routines and data handling demonstrations, many of which were developed as aids to teaching A-level computing and so may possibly be useful in schools.

John Lyons was possibly the first commercial software distributor to take an active interest in distributing software via the public domain. *Fidei Defensor* and *Multiple Choice Quiz for Christians* are two packages which he released into the public domain in 1991; both have a similar theme in Christianity, and are of use equally in religious studies and in the home. Also last year, he released a demonstration disc of his range of educational software into the public domain. This gave details of his range of software and included examples from various packages in the range. Now he has decided to re-release almost his entire product range as shareware. I have mentioned shareware before, but briefly the term refers to software which may be distributed by anyone to anyone, but which you must pay for if you find it useful.

That serves as a 'taster' for the subject which I will be covering in the next issue: the remaining two of the three areas which were mentioned at the start of this column are shareware and free software. Unfortunately, there isn't room to cover these this month, so they will be the subject next time. The John Lyons range will be covered, together with a look at the free software available from Peter Davy for use in adult basic education.

Best of BEEBUG



Applications I Disc

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

File Handling for All

on the BBC Micro and Acorn Archimedes

by David Spencer and Mike Williams

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

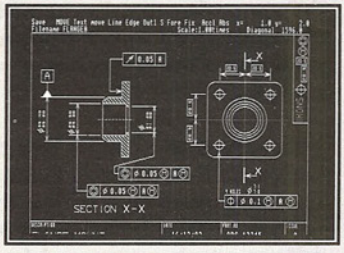
File Handling for All, written by highly experienced authors and programmers David Spencer and Mike Williams, offers 144 pages of text supported by many useful program listings. It is aimed at Basic programmers, beginners and advanced users, and anybody interested in File Handling and Databases on the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

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

The topics covered by the book include:

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

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



ASTAAD

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

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

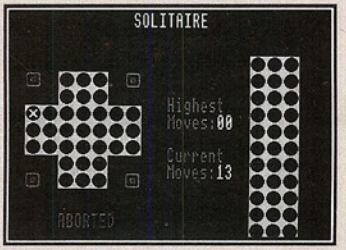
	Stock Code	Price		Stock Code	Price
ASTAAD (80 track DFS)	1407a	£5.95	**ASTAAD** (3.5" ADFS)	1408a	£5.95
Applications II (80 track DFS)	1411a	£4.00	**Applications II** (3.5" ADFS)	1412a	£4.00
Applications I Disc (40/80T DFS)	1404a	£4.00	**Applications I Disc** (3.5" ADFS)	1409a	£4.00
General Utilities Disc (40/80T DFS)	1405a	£4.00	**General Utilities Disc** (3.5" ADFS)	1413a	£4.00
Arcade Games (40/80 track DFS)	PAG1a	£5.95	**Arcade Games** (3.5" ADFS)	PAG2a	£5.95
Board Games (40/80 track DFS)	PBG1a	£5.95	**Board Games** (3.5" ADFS)	PBG2a	£5.95

All prices include VAT where appropriate

RISC Developments Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS.

Best of BEEBUG

Board Games



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

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

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

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

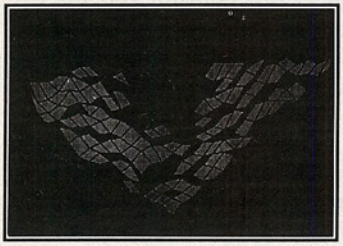
CRIBBAGE - an authentic implementation of this very traditional card game for two, where the object is to score points for various combinations and sequences of cards.

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

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

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

Applications II Disc



CROSSWORD EDITOR - for designing, editing and solving crosswords

MONTHLY DESK DIARY - a month-to-view calendar which can also be printed

3D LANDSCAPES - generates three dimensional landscapes

REAL TIME CLOCK - a real time digital alarm clock displayed on the screen

RUNNING FOUR TEMPERATURES - calibrates and plots up to four temperatures

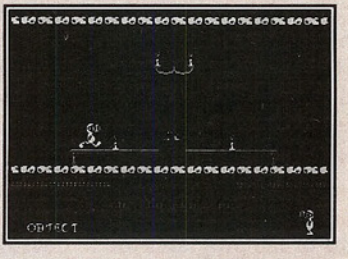
JULIA SETS - fascinating extensions of the Mandelbrot set

FOREIGN LANGUAGE TESTER - foreign character definer and language tester

LABEL PROCESSOR - for designing and printing labels on Epson compatible printers

SHARE INVESTOR - assists decision making when buying and selling shares.

Arcade Games



GEORGE AND THE DRAGON - Rescue 'Hideous Hilda' from the flames of the dragon, but beware the flying arrows and the moving holes on the floor.

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

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

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

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

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

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

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

	Stock Code	Price
File Handling for All Book	BK02b	£ 9.95
File Handling for All Disc (40/80T DFS)	BK05a	£ 4.75
Joint Offer book and disc (40/80T DFS)	BK04b	**£ 11.95**

	Stock Code	Price
File Handling for All Disc (3.5" ADFS)	BK07a	£ 4.75
Joint Offer book and disc (3.5" ADFS)	BK06b	**£ 11.95**

P & P	UK	Europe	Americas, Africa Mid. East	Elsewhere
a	£ 1.00	£ 1.60	£ 2.40	£ 2.60
b	£ 2.00	£ 3.00	£ 5.00	£ 5.50

Carriage is donated by the letter after the stock code. When ordering several items use the highest price code, plus half the price of each subsequent code.

Tel. (0727) 40303 Fax. (0727) 860263

BEEBUG Education

Modern Languages

by Mark K. Sealey

The Scottish Council for Educational Technology has recently released an enterprising batch of packages which will form the basis of reviews over the next issue or two of Beebug Education.

To begin with, we will examine some software for a curriculum area to which we have not given much attention previously in this column: modern languages.

Product	**En Ville, In der Stadt**
Supplier	**Scottish Council for Educational Technology, 74 Victoria Crescent, Glasgow G12 9JN.**
Price	**£15.00 ex. VAT.**

Firstly, En Ville and In der Stadt are French and German versions, respectively, of a simulation. The scenario is a shopping expedition by someone who has been staying with a penfriend in the relevant country.

AIMS

The aims of the programs are to develop language skills, provide substitute experience of shopping in a French or German town, and consolidate reading skills. I use the word 'consolidate' because it is clearly necessary for pupils (probably of middle secondary age) to have some facility in the language - the suite will not really teach any aspect of French or German from scratch.

The package could also usefully fit into a cross-curricular project on shopping, and be employed to direct pupils' language specifically to use intentional and planning registers - maybe in groups away from the computer.

USE

Decision making is at the core of the program, and this is evident from the first screen after booting and the title page. Here, participants are required to choose how they will get to the shopping area... on foot, by taxi or bus etc.

All the instructions are in French/German. Thus by working with the programs pupils ought to acquire familiarity with the equivalent expressions for menu handling and other housekeeping tasks such as pressing the space bar to continue, as instructions in the computer program are not in English.

Since both French and German have characters, accents and umlauts, that do not appear in the English alphabet, use has been made of the function keys to accommodate this. The manual has cardboard strips for the purpose. It is important because the user may have to type in sentences or word answers instead of selecting a number against the correct string. This can be determined by the teacher (see later).

Wherever this is the case, exact spelling is necessary. Children often fail to distinguish between a and ä, say, and this feature should help to reinforce the point.

After arriving at the shopping centre, successive screens representing one of 15 shops are presented; each has a chunky but effective graphic, two time-displays (the current time and the start-out time), and the simple menu panel.

COMPLETION OF THE TASK

If the user needs to consult the shopping list and/or establish how well or badly they are doing in comparison with the tasks set, they can exit the shopping area and obtain a summary before re-entering to make further purchases. This happens by means of simple selection from a menu, and the computer prompts by asking whether each item is a gift or for personal 'consumption'.

The application ends when the pupil has arrived back home for feedback on their shopping day, and points are awarded for how well they handled things in terms of times, budget and items chosen.

TEACHERS' EDITOR

One of the most useful features of En Ville/In der Stadt is the teachers' and pupils' editors, permitting a range of user-parameters to be set. Chief amongst these is the facility that will require users to enter complete answers to cues on screen, not merely to select the number corresponding to an answer string.

Data can also be updated - by pupils too - so that they could research the current prices of items for sale in the shops and, by entering these via the editor, work from up-to-date and real information. The same is true of general data about the country concerned (currency symbol, exchange rates and so on), which means that for French users the shopping trip could be set in Corsica or Zaire as well as Metropolitan France, for example.

Amusingly, the screen colour for the pupil editor changes to red "so that the teacher can see from a distance when it is being used"!

CONCLUSIONS

It is a pity that states reached in the simulation cannot be saved. On the plus side, decisions have to be taken when a bus is late or there is the option to purchase the ticket on the train: further realistic language practice.

The documentation is adequate, if a little slim. The program is easy to use, and could spawn a number of related activites. As such, it deserves to have a specific and well defined place in modern language and continuing education classes for some time to come.

TELETEL FOR MASTER 128

Product	**Teletel**
Supplier	**Scottish Council for Educational Technology, 74 Victoria Crescent, Glasgow G12 9JN.**
Price	**£15.00 ex. VAT (Parts 1, 2 & 3).** **£5.00 ex. VAT (Metz data).** **£5.00 ex. VAT (Dieppe data).** **£6.00 ex. VAT (Dieppe extended data file).**

Teletel is the French equivalent of Prestel, and is widely acknowledged as being superior to the BT service, which - in fairness - pioneered videotext in this country in the early 1980s.

The second package to be reviewed this month is a simulator for Teletel. Despite

the existence of Comms packages (such as the excellent ArcComm II for the Acorn 32 bit range, which specialises in European terminals), it would be financially prohibitative to provide pupils with regular access to the real network. Hence a package - with associated training and sample material - to emulate it.

The complete SCET package consists of two programs - the simulator itself and an editor, sample databses (on Dieppe and Metz) and an in-service training manual on the use of the editor.

Loading of the databases is straightforward. They then behave much as the real Teletel service itself, although clearly not so many services can ever be simulated (there are over 12,000). The package nevertheless gives a clear and easy to use idea of what it is like to use the database.

TECHNICALITIES

In each of the programs nearly all those operations that you would wish to alter can be easily set to ensure smooth operation: printer configuration, time delay for display, access to relevant O.S. commands from within the program. To make full use of the suites, access to a network, a mouse and twin 80 track disc drives is mandatory. In any case, you must have at least a Master and 80 track drives.

THE EDITOR

Designed to handle about 100 frames to be stored on a single disc (a twin disc system is envisaged), this part of the software can actually log onto Teletel (though for this a separate (Aldoda) ROM must be purchased and installed) to capture frames. These are then to be edited - thus providing an emulation of the more complete service. As with the real thing, frames can be routed and linked, the system can be made fully interactive - only the graphics and scope of what the user can access will be different.

The documentation for this part of the package is the clearest and most satisfactory, though it lacks an index.

PLACE IN THE CURRICULUM

Teletel is a well thought out and easy to use - if specialist - package. Plenty of resource sheets, cribs, appendices with background and technical information have been assembled, and a realistic and accessible short cut to experiencing aspects of European culture as well as the world of comms has been found.

Those using this package will be able to practice research skills and make the sorts of inquiry, perhaps, that would equip them to update En Ville (current prices, for example). Many of the procedures and skills in handling this software effectively lie in the domain of Information Technology as much as language; at the same time, the appeal of the package is slightly more limited than that of the shopping simulations.

To sum up: a lot of thought and development time has clearly gone into both the products under review this month; as specific answers to specific requirements in the secondary curriculum, they are worthy of serious consideration.

Mr Toad's Machine Code Corner 2

This month our slippery friend looks at screen printing.

In some issues, Machine Code Corner will be aiming to treat topics aimed frankly more at the beginner than at the experienced programmer, and this month's column is one such: Mr Toad wants to look at printing from assembly-language.

Some simple applications may not need to print to the screen, but most do. Many need their own screen, or screens, but these are no harder to implement than simple prompts once your print-routine is set up. For any normal purposes, the MOS routines OSWRCH and OSASCI are quite fast enough; these take care of a huge amount of work for you, such as expanding bytes in some modes to take account of colour, keeping track of the cursor position and scrolling.

Generally speaking, you place your data in a neat section of the listing (Mr Toad prefers the end), heading each section with a label and ending with an end-marker byte, which is most likely to be zero, though we shall see in a moment that there are other possibilities, for example:

```
.toadText
EQUS"Mr Toad rules crOaK":BRK
```

Note that BRK assembles a zero, since zero is the opcode for BRK; this is shorter than EQUB 0.

Now choose OSWRCH or OSASCI for your print routine. The latter does a new line as well as a carriage-return when it gets a &0D sent to it, and so is generally handier. A, X and Y emerge unchanged after the call, but the flags (P) cannot be guaranteed to do so, which is a nuisance.
At the top of the listing most people like to put oswrch=&FFEE or osasci=&FFE3, as labels are easier to remember. Pick a couple of page-zero addresses - I like to use &B0 to &BF, as these are free for transitory use. Write something like: printLo=&B0 (or wherever), printHi=&B1. The actual print-routine is like this:

```
.print
STY printHi
STX printLo
LDY #&FF
.loop
INY
LDA (printLo),Y
PHP
JSR osasci
PLP
BNE loop
RTS
```

You push the flags so that when the LDA picks up the end-marker and the zero-flag is set, the PLP guarantees that it is still set after the call to OSASCI - which may have corrupted it - so the BNE fails and the routine exits.

By beginning with Y=&FF we can tuck the INY away at the start of the loop where its effect on the flags won't upset things; this is worth remembering in other kinds of loop, too. It's not the only way, of course. The end- marker, zero, is sent to OSASCI but is ignored. If you don't want the end-marker to be 'printed', you have to write a BEQ after the LDA as well as a JMP at the end:

```
.loop
INY
LDA (printLo),Y
BEQ end
JSR osasci
JMP loop
.end
RTS
```

This is untidy. JMP takes three bytes, but BRA (only available on the Master) will take an extra cycle if a page-boundary is crossed, so Mr Toad invariably jumps. What use has a reptile for a BRA, anyway?

Call your print routine by putting the address of the label indicating the start of your text into XY:

```
LDY toadText DIV &100
LDX toadText MOD &100
JSR print
```

The page zero locations don't need to be preserved between calls. There are other approaches but this method is as good as any.

There are times when we don't use zero as an end-marker, however, even though this then involves a CMP# instruction in the print loop. If you are only printing short prompts throughout the program, &0D is a handy marker: you get a new line after each prompt. In such cases, though, Mr Toad often uses a 7 as marker, since a nice beep is produced each time. The loop is now as follows:

```
.loop
INY
LDA (printLo),Y
JSR osasci
CMP #&0D \ (or #7)
BNE loop
RTS
```

When we go into sideways ROM headers, we shall see that there are cases when neither zero nor &0D nor 7 will do - here Mr Toad always uses &1B, which does nothing when 'printed'; a 6 (enable VDU) is also harmless.

Say you want to begin a completely new screen. The new screen had better set the desired mode first; send &16 to OSASCI, then the mode - let's say, shadow mode 135.

```
.newScreen
EQUB &16:EQUB &87 \ (i.e. 135)
```

Note that mode 135 on the Master looks the same as mode 7 on the Beeb.

Now call your print-routine as above. Bingo - a nice clear mode 7 screen with the cursor at 0,0. We can do a TAB next using EQUB &1F followed by the column, then the row (as with TAB). Want page-mode on? Include EQUB &0E. A beep? EQUB 7. Printer on or off at some point? EQUB 2 or 3. Thus, armed with nothing more esoteric than pages 181 to 187 of the Master's Welcome Guide (the VDU codes), any or all possible effects can be achieved more easily and neatly than in Basic.

If the screen is mode 7 you can include colour codes in the listing with the function keys, as in Basic. Don't forget the end-marker.

AND FINALLY

You could include the following subroutine in your listing if OSWRCH/OSASCI are called often:

```
.oswrch
PHP
JSR &FFEE
PLP
RTS
```

Very structured, is that.

For those with a Spectrum, good Z80 assemblers allow DEFB (like EQUB) followed by any number of bytes separated by commas. Why couldn't we have that? After all Basic's DATA allows it. Still, you can tidy up listings by using EQUW for 2 bytes of data or even EQUD for four: EQUW &8716 sets mode 135, as EQUW assembles the bytes in reverse order.

You can kill the cursor (VDU 23,1,0;0;0;0;) - which Mr Toad always does because nothing looks scruffier than a white flashing line in the middle of a lovely artistic screen - with EQUW &0117 (that's 23,1), EQUD 0:EQUD 0 (that's eight noughts).

EQUW followed by a one-byte number assembles that number followed by a zero - handy for the old end-marker: EQUW 7 = 'beep and end', or EQUW &0D = 'new line and end'.

Now for this month's quiz - what does this lot do?

```
.beebugAd
EQUW &8716:EQUW &0117
EQUD 0:EQUD 0
EQUW &041F:EQUW &070B
EQUB &88
EQUS"Beebug kills all known bugs"
EQUW &0D
```

The sender of the first wrong entry opened wins a weekend in Bognor with Mr Toad's children.

Next month: bolting a dynamo onto your exercise-bike - how to cut down your electricity bills and get fit as you program.

BEEBUG Function/Procedure Library (11)

by R.W. Smith

This month we come to the final part of R.W. Smith's procedures and functions. These deal with sideways RAM and will give you the opportunity to explore, from Basic, the extra power it supplies. This has been a substantial contribution and we hope that you have found it useful.

THE FUNCTION/PROCEDURE LIBRARY (PART 11)

Routine 30: Sideways memory access: Get character

Type: FUNCTION
Syntax: FNgc(K%,N%) Purpose: To call characters from sideways memory into working memory.
Parameters: *K%* is the sideways memory absolute byte number. *N%* is the sideways memory bank number.
Notes: See later routines for placing the data into sideways memory.
Related: Uses FNpk
Example:

```
10 N$=FNgc(&8020,4)
```

Routine 31: Get integer from sideways memory.

Type: FUNCTION
Syntax: FNgn(K%,N%)
Purpose: To get an integer from sideways memory and to place it into an integer variable.
Parameters: *K%* is the sideways memory absolute byte number. *N%* is the sideways bank number.
Notes: Creates the integer from four bytes as placed into sideways memory by Routine 40
Related: Uses FNpk
Example:

```
10 K%=&8010
20 N%=4
30 Y%=FNgn(K%,N%)
```

Routine 32: Get a string from sideways memory.

Type: FUNCTION
Syntax: FNgs(K%,N%)
Purpose: To get a string from sideways memory and place it in a string variable.
Parameters: *K%* is the sideways memory absolute byte number. *N%* is the sideways bank number.
Notes: The string must be ended with a carriage return character (ASCII 13) when placed into memory.
Related: Uses FNgc
Example:

```
10 K%=&8020
20 N%=5
30 N$=FNgs(K%,N%)
```

Routine 33: Pick a byte in sideways memory.

Type: FUNCTION
Syntax: FNpk(K%,N%)
Purpose: To read a byte from sideways memory using the Operating System routine - OSRDSC.

Parameters: *K%* is the absolute address in sideways memory. *N%* is the sideways bank number.
Notes: None.
Related: Used by routine 30, 31 and 32.
Example:

```
10 K%=&8000
20 N%=6
30 Y%=FNpk(K%,N%)
```

Routine 34: Poke a block of bytes into sideways memory.

Type: PROCEDURE
Syntax: PROCpo(K%,N%,L%)
Purpose: To place a block of bytes into sideways memory using OSWORD 66. The block can be of varied data.
Parameters: *N%* is the address in sideways memory which must be in absolute notation - &8000 to &BFFF. *K%* is the sideways bank number. *L%* is the number of bytes in the block.
Notes: The block of bytes to be transferred must be in free memory starting at &B00.
Related: Uses PROCpp: Used by routine 39 and 40
Example:

```
10 A$="This is the name of the game"
20 $&B00=A$
30 PROCpo(5,&8000,LEN(A$))
```

Routine 35: Pick a block of bytes from sideways memory.

Type: PROCEDURE
Syntax: PROCpi(K%,N%,L%)
Purpose: To transfer a block of bytes from sideways memory to free memory starting at &B00. Uses OSWORD 66. The block of bytes can be string or numeric data which must be identified after the transfer. See later routines which deal with this.
Parameters: As Routine 34.
Notes: None.
Related: Uses PROCpp.
Example:

```
10 PROCpi(5,&8000,20)
20 A$=LEFT$($&B00,20)
```

Routine 36: OSWORD routine for block transfer to/from sideways memory.

Type: PROCEDURE
Syntax: PROCpp
Purpose: To use the OSWORD Routine to pick or poke blocks of bytes from free memory to sideways memory.
Parameters: None. Taken from related routines.
Notes: The routine uses absolute or pseudo addressing.
Related: Used by PROCpo, PROCpi, PROCpop, PROCpip.

Routine 37: Poke block to sideways memory using pseudo addressing.

Type: PROCEDURE.
Syntax: PROCpop(4,N%,L%)
Purpose: As routine 34 but using pseudo addressing.
Parameters: Pseudo address - *N%* being from 0 to &FFBF, *L%* being number of bytes in block. The first parameter is always 4 indicating that the first byte 0 is in bank 4.
Notes: Pseudo addressing mode must have be set within program by *SRDATA *N%*
Related: Uses PROCpp

Example:

```
10 *SRDATA 5
20 A$="This is the Data Block"
30 $&B00=A$
40 K%=4
50 N%=&100
60 L%=LEN(A$)
70 PROCpop(K%,N%,L%)
```

Routine 38: Pick block to sideways memory using pseudo addressing.

Type: PROCEDURE
Syntax: PROCpip(K%,N%,L%)
Purpose: As routine 35 but using pseudo addressing.
Parameters: As routine 37.
Notes: Pseudo addressing mode must have been set using *SRDATA.
Related: Uses PROCpp.
Example:

```
10 *SRDATA 5
20 K%=4
30 N%=&100
40 L%=30
50 PROCpip(K%,N%,L%)
60 A$=LEFT$($&B00,L%)
```

Routine 39: Put block of string data into sideways memory.

Type: FUNCTION
Syntax: FNpokes(K%,N%,N$)
Purpose: Routine 34 can be used when the length of the strings to be put in sideways memory is fixed. When variable length strings are to be stored, it is necessary to make provision for the length of the string to be available when recalling from memory. This routine adds the length of the string as the first character to the string before storing into memory. Additionally the function returns the next memory byte address at which the next block of data commences.
Parameters: *K%* is the bank number. *N%* is the sideways absolute address - &8000 to &BFFF. *N$* is the string or string variable.
Notes: The string is placed in Free memory at &B00 before being stored. If the parameter *N%* is used repeatedly, it will not need to be reset after each call if used as the function call variable.
Related: Uses PROCpo.
Example:

```
10 N%=&8000
20 K%=4
30 A$="THIS IS THE FIRST STRING"
40 B$="THIS IS THE SECOND OF THE
STRINGS"
50 N%=FNpokes(K%,N%,A$)
60 N%=FNpokes(K%,N%,B$)
```

Routine 40: Put an integer into sideways memory.

Type: FUNCTION
Syntax: FNpoken(K%,N%,L%)
Purpose: To store integers into sideways memory.
Parameters: *K%* is the bank number *N%* is the absolute memory address. *L%* is the integer or integer variable.
Notes: The integer is placed in free memory at &B00 before storing. The function returns the next sideways memory address.
Related: Uses PROCpo
Example:

```
10 K%=4
20 N%=&8000
```

```
30 Z%=1234567
40 Y%=890123
50 N%=FNpoken(K%,N%,Z%)
60 N%=FNpoken(K%,N%,Y%)
```

Routine 41: Get a variable length string from sideways memory.

Type: FUNCTION
Syntax: FNpicks(K%,N%)
Purpose: To recall a variable length string from sideways memory, placed there by routine 39.
Parameters: *K%* is the bank number. *N%* is the absolute memory address
Notes: Uses free memory at &B00. The memory address is not updated during the function and is the responsibility of the programmer.
Related: Uses PROCpi
Example:

```
10 K%=4
20 N%=&8000
30 A$=FNpicks(K%,N%)
```

```
40 N%=N% + LEN(A$) + 1
50 B$=FNpicks(K%,N%)
```

Routine 42: Get an integer from sideways memory.

Type: FUNCTION
Syntax: FNpickn(K%,N%)
Purpose: To recall an integer from sideways memory as placed there by routine 40.
Parameters: *K%* is the bank number. *N%* is the absolute memory address.
Notes: Uses free memory at &B00. Note that this routine is complementary to routine 31(FNgn) and is included to retain compatibility with routine 41
Related: PROCpi
Example:

```
10 K%=4
20 N%=&8000
30 Z%=FNpickn(K%,N%)
40 N%=N%+4
50 Y%=FNpickn(K%,N%)
```

```
30500 REM Sideways Memory Access.
30510 :
30520 REM Get Character from Sideways Me
mory.
30530 :
30540 DEFFNgc(_%,`%): =CHR$(FNpk(_%,`%))
30550 :
30560 REM Get an Integer from Sideways M
emory.
30570 DEFFNgn(_%,`%)
30580 =FNpk(_%,`%)*&1000000+FNpk(_%+1,`%
)*&10000+FNpk(_%+2,`%)*&100+FNpk(_%+3,`%
)
30590 :
30600 REM Get a String from Sideways Mem
ory.
30610 :
30620 DEFFNgs(_%,`%) : _$=""
30630 REPEAT: _$=_$+FNgc(_%,`%): _%=_%+1
30640 UNTIL FNgc(_%,`%) =CHR$13
```

```
30650 =_$
30660 :
30670 REM Get Byte from Sideways Memory.
30680 :
30690 DEFFNpk(!&F6,Y%):=USR(&FFB9)AND&FF
30700 :
30710 REM Poke & Pick Sideways Memory.
30720 :
30730 REM Poke.
30740 :
30750 DEFPROCpo(_%,`%,__%): ?&80=128: PR
OCpp: ENDPROC
30760 :
30770 REM Pick.
30780 :
30790 DEFPROCpi(_%,`%,__%): ?&80=0: PROC
pp: ENDPROC
30800 :
30810 REM Sideways Memory Access Routine
```

Continued on page 46

Direct Memory Access (1)

by Alan Wrigley

This month First Course moves on to a new topic. *Direct memory access* refers to the practice of reading from, and writing to, specific addresses in the computer's memory. Over the next couple of issues we will consider why you might want to do this, how to do it, and how you decide exactly which part of memory to access, though not necessarily in that order since all three aspects are to some extent bound up with each other.

Reading from a memory location is sometimes referred to as "peeking", while writing to a location is known as "poking". Many dialects of Basic do in fact have two keywords, PEEK and POKE, which perform these very functions. Generally they only operate on one byte at a time and are therefore somewhat limited in their usefulness. However, BBC Basic has a more powerful set of commands for memory access, allowing you to read and write strings and four-byte integers as well as single bytes, as we shall see in a moment.

WHY ACCESS MEMORY DIRECTLY?

Many Basic programs can perform their duties quite happily without ever needing to know what goes on in the computer outside the space allocated to the program itself. BBC Basic in particular is a well-developed variant of the language with many powerful features that were missing in earlier Basics; for example, with some older versions of Basic, it was even necessary to write directly to memory locations to produce sound or graphics. Nevertheless, there are still many reasons why a program might want to address memory directly, and these will gradually become apparent as this series progresses.

INDIRECTION OPERATORS

First of all, though, we need to know exactly how to access memory directly. BBC Basic has a set of commands for this purpose which are known as *indirection operators*. The reason for this rather grand piece of jargon is simply this: referring to data held at a memory location is an indirect reference to that data rather than a direct one. For example, if your program refers to a variable X% which holds a value to be used by the program (123 or -20, say), then you are using the contents of X% directly. But if X% holds the address of a memory location, such as &701C, then you are using X% not as a value itself but as an indirect reference to the value held in that location (note that memory addresses are normally referred to in hex, as here). So an indirection operator signifies "the contents of" rather than "the value of".

There are three indirection operators available on the BBC micro. These consist of the single characters ?, ! and $, and they operate on a single byte, four bytes and a string respectively. If you need to refer to them verbally they are known as query, pling and dollar. They can all be used on either side of an expression, i.e. for both reading from and writing to memory, and they can operate on either a constant or a variable. To take a simple example, suppose that you want to put a value of 123 into location

&701C (don't worry about why for the moment), you would use the byte operator as follows:

```
?&701C=123
```

and to read the value:

```
val%=?&701C
```

Equally, you could specify the location in the form of a variable:

```
loc%=&549A
?loc%=1
```

i.e. the location is &549A, and a value of 1 is placed at that location. Because we are using the query operator here, the range of values possible is 0 to 255 (which is the maximum value that can be held in 1 byte).

Before we go on to discuss the other indirection operators, we can explore the use of query with an example or two. When you type in a Basic program, or load one from disc, it is stored in memory, starting at PAGE (if you're not sure of the significance of PAGE, see *First Course*, Vol.10 No.9). This means that it is possible to use the indirection operators to look at the way the program is actually stored in memory. This is a complex topic whose full details are beyond the scope of the present article, but it does offer a useful way to test the use of the indirection operators. First of all, type in the following simple program:

```
10 PRINT "Hello"
20 END
```

A Basic program in memory starts with a carriage return (ASCII 13) marking the beginning of the program. This is followed by the first line, starting with two bytes which hold the line number (high byte first), a further byte giving the total number of bytes taken up by that line, and then the contents of the line itself, with keywords tokenised, and a carriage return to end the line. This is repeated for every subsequent line in the program, and finally the end of the program is signalled by a byte of 255. The function TOP will return the address of the first byte above the end of the program, so TOP-1 should contain 255. If you type in:

```
PRINT ?(TOP-1)
```

255 should be returned as the result. Now type in the following:

```
?(TOP-1)=123
```

which places a value of 123 in (TOP-1). You have now interfered with the end-of-program marker, so if you try to list the program you will get a "Bad program" error. Restore the marker with:

```
?(TOP-1)=255
```

and you can list it properly again.

If you have typed in the program correctly (i.e. with the spaces exactly as shown), the H of "Hello" should be located at PAGE+8. Type:

```
?(PAGE+8)=74
```

and list the program again; you should now see that line 10 is listed as:

```
10 PRINT "Jello"
```

because the value of 74 that you have inserted into that location is the ASCII code for J.

Note that your programs should *never* be written in such a way that they modify themselves - this is extremely bad programming practice. We have only included this example here in order to demonstrate the use of indirection operators.

If you are using a variable as the operand to the query operator, then there is a useful extension to the syntax which allows you to add an offset to the variable. For example, whereas *?loc%* accesses the contents of *loc%* as we have seen, *loc%?1* refers to the contents of

loc%+1, loc%?2 to the contents of *loc%+2* and so on. This is a simpler way of saying *?(loc%+1), ?(loc%+2)* etc. This comes into its own when the offset itself is a variable, as in:

```
loc%?offset%
```

This provides the means for referencing tables of data in memory, where the offset of the byte required is calculated elsewhere in the program. For example, you could read every tenth byte of a 100-byte table as follows:

```
FOR offset%=0 TO 99 STEP 10
PRINT loc%?offset%
NEXT
```

A thoroughly practical example of using an indirection operator in this way was the movedown routine for the model B described in the Vol.10 No.9 *First Course* referred to above. To recap, this was designed to move a program down in memory once loaded to allow more space for it to run (if you are unsure about the reasons for doing this, you should refer to the article). The routine programs a function key to move the program down, reset PAGE to a lower value, and run the program at the new value of PAGE. The actual code given in Vol.10 No.9 used the pling operator to move the program by four bytes at a time, but we can just as easily do it one byte at a time using the query operator. To move it down from the current value of PAGE to a new value of, say, &1400 we would use:

```
FOR A%=0 TO (TOP-PAGE)
A%?&1400=A%?PAGE
NEXT
```

The length of the program is returned by (TOP-PAGE), and so A% is being used as an offset starting at zero and going up to the program length. Each time through the loop, the value at (PAGE+offset) is transferred to (&1400+offset), which results in the entire program being moved down to start at &1400. Once PAGE has been set to the new value, the program can be run just as if it had always been there.

FOUR-BYTE VALUES

Being able to insert a single byte into memory, and read from a single location, is very useful for a number of purposes. However, BBC Basic, unlike some other dialects of the language, can handle four-byte integer variables with potential values far greater (-2147483648 to +2147483647) than can be placed into a single byte. It would therefore be extremely useful to be able to put a whole four-byte integer into memory at one go, and this is the purpose of the pling operator. It is used in exactly the same way as query, as for example:

```
!loc%=2000
```

but in this case the value is stored as four consecutive bytes starting at *loc%*, with the least significant byte first. Note that the value is *always* stored as four bytes when pling is used, even if it could have fitted into one byte (i.e. 255 or lower).

Pling can also make use of the same syntax extension as query to enable offsets to be used. This is done in exactly the same way, i.e.:

```
loc%!offset%=1234
val%=loc%!offset%
```

You can try out the use of pling with the following program:

```
10 DIM loc% 3
20 !loc%=987654321
30 FOR i%=0 TO 3
40 PRINT loc%?i%:NEXT
```

You should get the values 177, 104, 222 and 58, representing the values held in the four bytes starting at *loc%*. These values do not make a lot of sense at first

sight, but if you change line 20 to:

```
20 !loc%=&3ADE68B1
```

(which is the hex representation of 987654321) and force the values to be printed in hex by altering line 40 to:

```
40 PRINT ~loc%?i%:NEXT
```

you will get B1, 68, DE and 3A. You can immediately see that these are the four bytes of the number in reverse order. This is why you will usually find it very much easier to work in hex when handling four-byte numbers. Note the use of DIM in line 10 to reserve a block of memory for *loc%* - we will discuss how and when to do this later in the series.

Now alter line 20 to:

```
20 !loc%=25
```

and run the program again. Now you will see that the four bytes pointed to by *loc%* contain 25 (or &19 in hex), 0, 0 and 0. This demonstrates the point that a single-byte number still occupies the full four bytes.

There is much more we could say about pling, and in the course of these articles we will cover a number of uses for it, but just for the moment we will finish this month by looking again at the movedown routine referred to above. We used query to move the program down a byte at a time, but using pling is much more efficient as you can move four bytes at a time. All you need to do is substitute one operator for the other and alter the loop to increment by four, as follows:

```
FOR A%=0 TO (TOP-PAGE) STEP 4
A%!&1400=A%!PAGE
NEXT
```

B

BEEBUG Function/Procedure Library (continued from page 42)

```
30820 :
30830 DEFPROCpp
30840 !&81=&B00: ?&85=__%: !&86=0: ?&88=
`%: ?&89=`% DIV 256: ?&87=_%
30850 A%=66: Y%=0: X%=&80: CALL&FFF1
30860 ENDPROC
30870 :
30880 REM Pseudo Addressing Mode Poke.
30890 :
30900 DEFPROCpop(_%, `%, __%): ?&80=192:
 PROCpp: ENDPROC
30910 :
30920 REM Pseudo Addressing Mode Pick.
30930 :
30940 DEFPROCpip(_%, `%, __%): ?&80=64:
PROCpp: ENDPROC
30950 :
30960 REM Put Data to Sideways Memory.
30970 :
30980 REM String to Sideways Memory (Abs
olute Addressing)
30990 :
31000 DEFFNpokes(_%, `%, _$)
31010 $&B00=CHR$(LEN(_$))+_$: PROCpo(_%,
 `%, LEN(_$)+1)
31020 =`% +LEN(_$)+1
31030 :
31040 REM Integer to Sideways Memory (Ab
solute Addressing)
31050 :
31060 DEFFNpoken(_%, `%,__%):!&B00 = __%
31070 PROCpo(_%, `%, 4):  =`%+4
31080 :
31090 REM  Get a variable length string
from Sideways Memory
31100 :
31110 DEFFNpicks(_%,`%)
31120 PROCpi(_%,`%,1)
31130 `%=`%+1:__%=?&B00
31140 PROCpi(_%,`%,__%)
31150 ?(&B00 +__%)=13
31160 =$&B00
31170 :
31180 :
31190 DEFFNpickn(_%,`%)
31200 PROCpi(_%,`%,4)
31210 =!&B00
31220 :
```

B

512 Forum

by Robin Burton

This month we'll continue our look at the outside world for a while; after all the weather's steadily improving and the nights are getting lighter.

CHANGING TIMES

It's becoming apparent from my postbag that a steadily increasing number of 'new' 512 users are reading the Forum, which probably means that a proportional number of existing readers must be upgrading their hardware and moving on.

This is hardly startling or unexpected, but it has been made more pointed by the fact that a good many of the query topics I've had recently are items that have long since been covered in 512 Forum, some as long ago as two or three years.

As I mentioned last month, things aren't standing still in the PC world, and naturally it's just the same for Acorn users too. Over time, some BBC micro/512 users migrate to later systems, either to the Archimedes or as in quite a number of cases I've heard of, to a PC.

In both upgrade paths there's plenty to cater very well for the needs of 512 users, regardless of whether they see themselves primarily as DOS or BBC enthusiasts. Given the wealth of PC software, for 512 owners who have spent most of their time in DOS rather than BBC mode, the move to a PC can only improve all their existing DOS facilities and add a vast number of new opportunities.

At the same time, many of these users will have some of their own BBC Basic programs which they don't want to lose. These will have been previously catered for either by the BBC micro itself, or perhaps by 512BBCBASIC. Even for those with quite an extensive Basic program library however, a switch to M-Tech's PC version, BBCBASIC(86) (reviewed in the Forum in early 1989 and similar to 512BBCBASIC) offers a simple and virtually painless transition.

For those who choose the other obvious upgrade path, the support for BBC Basic and even machine code programs and ROM software is well known, while the Archimedes PC emulator can cater for DOS needs, although it must be acknowledged that it's pretty slow, even compared to the humble 512. Still, it's not expensive and it does do the job. However, as you might have seen, DOS options for Archimedes users have now been dramatically enhanced by the recent announcement of the 386SX based Archimedes expansion card by Aleph One.

The processor clock speed is 20MHz, so this upgrade now offers the chance of 'real' PC performance for the Archimedes. While it may not be quite as fast as an equivalent speed 386 PC, the expansion board will obviously show a clean pair of heels to the 512 since it is, in effect, a much faster co-processor than a 512 attached to a much faster host than an eight-bit BBC micro.

However, on the negative side it must be said that at a price of £495 for a 1 Mb board and £625.00 for 4 Mb of RAM (prices ex. VAT), the expansion board is by no means the bargain of the century. In fact the expansion board alone is a substantial proportion of the cost of a

complete 386 PC system, without considering the cost of the Archimedes. For someone who mainly uses DOS therefore, a PC might still be a more attractive proposition, given that BBCBASIC(86) can be purchased for a PC at well under £100.00 to fill in the only obvious gap.

UPGRADING?

Don't get the idea I'm particularly trying to recommend an upgrade to a PC. Even given what I've said above, plus last month's comments about current PC processor power and cost, anyone who has seen Microsoft Windows (even 3.1) will know that it's still not a patch on the RISC OS Desktop. Added to that, there are of course a great many improvements and additions to many aspects of the system in RISC OS 3 too.

The best that can be said for MS Windows in my opinion, bearing in mind that in general I'm not a mouse fan no matter who breeds it, is that the software can only improve with time, but it still has a fair way to go.

Of course the other ingredient in making a decision on which way to go, if or when you feel it's time to upgrade, is the range and price of software for your intended new system. Certainly the range of applications available for a PC isn't a problem as there are literally tens of thousands of packages, but while the price of some Archimedes software may come as a bit of an unwelcome surprise to long time BBC users, they're actually generally quite reasonable. By contrast the prices asked for certain PC packages are so high it's difficult to take them seriously.

As an example, I recently noticed in a PC magazine that one particular software package (for a single user system by the way, not a site licence) is over £1250.00. Still, it was the April issue of the magazine, so maybe that's the explanation!

As I said, I'm not trying to promote PCs in particular as the best upgrade path for 512 users, nor the Archimedes range for that matter either. The justification for this discussion is simple. It's no good adopting the ostrich mentality, thinking that if you bury your head in the sand and ignore progress it will go away. We haven't looked at general developments in 512 Forum very frequently in the past, if at all, but obviously it is an area of interest to at least some 512 users. That's why it was featured as part of last month's Forum and again now.

Having been prompted to think about possible upgrades by some of my recent mail, the truth is that I find one particular aspect ironic and, to be honest, just plain amusing. After all, the single group of Acorn users least well served (i.e. ignored) for years, both by Acorn and its hundreds of third party suppliers (with so few exceptions you can count them on one hand) now find themselves 'spoilt for choice' when it's time to consider an upgrade.

That there are two completely different routes to choose from, both of which offer very simple and direct growth paths for much if not all of a user's existing software is remarkable if not unique. Perhaps there is justice after all.

As to advice, the only suggestions I'd offer to would-be upgraders from the BBC/512 is common sense. First, decide exactly what gains you expect from the change and decide objectively (not forgetting costs) whether it's justified right now, or whether you should wait a while longer. When you do decide it's time to do something choose the growth path that you think best suits not only

your current plans, but your likely future needs too.

Don't forget to add into your decision list associated factors, such as the range and cost of the types of application that interest you most, the typical cost of extra hardware you might want to add sooner or later, such as a hard disc, a scanner, a digitiser, CD-ROMs, MIDI equipment and so on. The initial hardware cost of upgrading is most certainly not the only item to weigh up. The same extras often don't cost the same for different systems.

CONSTRAINTS

Naturally, for most 512 users who sell up and move on, there's another, usually existing BBC micro user, who buys the equipment and finds a completely new world of opportunity opening up, with usually a fair number of new problems too. This changing user base poses a problem for me too in the Forum, so I'm going to ask you for your input.

My difficulty is that, while I've always tried not to repeat topics previously dealt with in the Forum, this inevitably gets harder all the time. For one thing, this is (I think) the thirty-eighth 512 Forum, so a lot of ground has been covered over the past four years. On top of this, the 512's software, especially the operating system (but applications too for the reasons given last month), and the hardware (except for a memory expansion) is pretty well static. In consequence the new topics that continually crop up for current systems don't exist for the 512.

Another problem is that old hands are, by definition, experienced, so it's difficult to find new items to interest them anyway. At the same time new users are finding things in the 512 or DOS as much of a mystery as they once were to many of us. Of course many of these 'new user' topics we've already covered in past issues.

My difficulty therefore, is trying to find a balance in the Forum. The ideal is that on the one hand experienced users who aren't upgrading (yet) don't become bored, while on the other, new 512 owners can learn some of the things which will help them get more out of the system.

What I'd like *you* to do therefore, whether you're a new user or not, is to let me know the sort of things you'd like to see in the Forum. Some of you will know I've made similar requests from time to time in the past, but let me offer some guidelines.

One frequently repeated request is an up to date software compatibility list. I understand how useful this could be, but such a list would take a lot of effort and time to compile. It also assumes that the necessary user input is forthcoming. It usually isn't, or at least not in enough volume (or detail) to make a useful list of a wide range of applications.

If anyone wants to contribute to such a list, I'll publish the information in the Forum in batches, but I can't do it without you. For this sort of exercise, nothing is worse than single pieces of information that arrive over a lengthy period of time. Equally, the information is virtually useless to others if it isn't complete.

The necessary information is obvious, but all too frequently at least one vital item is missing from what users tell me, so I'll list the points which are needed if details are to be useful to others.

Often a program isn't given its full and correct name, or there's no version number. If that part's OK, the source of supply (including shareware or public domain outlets) or the correct name of the publisher for commercial software

often isn't included. Less vital, but useful all the same, is the price of a package. Very rarely have I been given that information for commercial software.

Another point which others need to know is whether a program runs in a standard memory 512, or needs expanded memory. Finally, very few mention which version of DOS Plus they're using; it's not much help if I to have to guess whether it's 1.2 or 2.1.

QUICKIES

The final two points for this month are old and well known facts for most 512 users. However, they are causing problems again for a few new users.

It's been quite a surprise to me to find out just how many 512 owners have recently purchased a board complete with a copy of DOS Plus that turns out to be version 1.2.

This means that the previous owner simply hadn't bothered to upgrade and didn't bother to mention it to the new owner either, or perhaps didn't even know that there is a later version. There is, it's 2.1 and it can still be obtained from Acorn Customer Services at Fulbourn Road, Cherry Hinton, Cambridge, tel. 0223 245200.

If you're still using 1.2 you should upgrade. There's no charge and application compatibility will be much improved.

Another old problem that's catching out a number of new users is the 'Can't find command.com' message on leaving an application. This is easy to cure if you know how, but many are confused because they think that setting 'PATH' ought to fix it. It doesn't.

First, make sure that COMMAND.COM is in a current path, which means that it must be on one of the discs that's in the system at the time you get the problem. If this path isn't the root directory of the default (current) drive at the time, you need to tell DOS where it is by using the 'SET' command.

For example, suppose you're running a program from drive B:, but COMMAND.COM is in the root directory of drive A:. Before you run the program that causes the problem, you should issue the command:

```
SET COMSPEC = A:\COMMAND.COM
```

This tells DOS plus, no matter which disc or directory you're in at the time, that COMMAND.COM should be loaded from the root directory of drive A: when required. If you haven't sufficient space on floppies for it, you can put COMMAND.COM in the RAM disc (which has speed advantages too) so long as you remember to tell DOS about it using 'SET'. B

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

THE GAME OF ZEUS (Vol.10 No.9)

In certain circumstances some blocks will not disappear quite correctly. To overcome this, add the following line to *Zeus2*:

```
2355 IF playfield%(CX%,CY%,0)<>CB% =0
```

Thanks to Albert Gardner for pointing this out. The version on that month's magazine disc includes this line.

MUSICAL MUSKRATS (Vol.10 No.10)

An error crept into the lines needed to be added to the MUSTRAN program on page 11 column 1. The correct line 220 should be:

```
220 REPEAT:PROCchoosesym:IF EP%=1 THE
N 230 ELSE UNTIL FALSE
```

B

Wordwise User's Notebook:
Making More of Markers

Chris Robbins adds some powerful features to Wordwise Plus.

One of the most powerful features of word processing packages in general is the ability to manipulate text in bulk; moving whole sections around, copying them, deleting them, and so on. In Wordwise (WW) and Wordwise Plus (WW+), blocks of text are marked by pressing the function key f3 to put a square blob on screen at the start and end of a section to be manipulated.

Adequate though this is, it's only possible to work with one block or section of text at a time. Wouldn't it be nice to get round this, and have more than just a measly two markers?

ADDITIONAL MARKERS

For WW+ users there's no great difficulty in implementing this; the segment program at the end of this article (*Marksb* on the disc) provides an additional 10 pairs of markers (numbered 0-9) which can be used in much the same way as the standard issue f3 variety. It makes use of the little known but highly useful fact that anything between Green (f1) and White (f2) markers is treated as an embedded command, so that although the extra markers can be seen in Edit mode, they do not affect either previewing or printing. They have the added advantage over standard f3 markers in that, like all embedded commands, they are saved with the text.

These extra markers have the form f1<n>f2, where <n> denotes a numeric character in the range 0-9. I've stuck to this range for the sake of simplicity, but there's no reason why it can't be increased (or reduced) by changing the appropriate sections of the program.

USING THE PROGRAM

Having loaded the program into a suitable segment, it can be invoked at any time from Edit mode by holding down Shift and pressing the appropriate function key e.g. if loaded into Segment 0, then Shift-f0. This will bring up the Main Menu which gives four options; *Set Markers, Delete Markers, Action Markers,* and *Collect Markers.* If nothing takes your fancy you can escape from this (as from all menus) naturally enough by pressing Escape, which will take you back to Edit mode.

Set Markers: Selecting this produces a display of the known current settings of the additional markers, whether one or both of a pair are in use, or free to be used. The emphasis on *known* is important since the additional markers can be manipulated just like ordinary text and, for instance, deleted without the marker system being aware of it. But more of that later.

Assuming one or both markers of a pair are available, pressing the appropriate numeric key (0-9) inserts the associated marker in the text at the current cursor position and returns you to Edit mode.

Delete Markers: This option also displays the known current settings. Pressing a numeric key (0-9) removes all the markers with the selected number, thus clearing a marked section and making the markers available for use elsewhere. It also comes in handy for removing the odds and ends of markers

that might be left over after a complicated sequence of block edit operations.

Action Markers: This option brings up a further menu providing facilities to *Find*, *Delete*, or *Copy* blocks. No *Move* facility has been included, since this would mean yet more code, and can in any case be achieved by a judicious application of *Copy* followed by Delete. Simply select the required operation then, when prompted, the marker number, and the operation will be performed.

Collect Markers: This, unlike the other Main Menu options, isn't so obvious, and perhaps requires a little more explanation.

I mentioned earlier that any additional markers left lying around in text are saved when the text is saved. But, when reloading a file containing these, they won't necessarily (if the Beeb has been turned off in the meantime, for instance) be *known* to the multiple marker system. That's where the *Collect* option comes in. Selecting *Collect*, either after a cold start using a saved file containing additional markers, or at any time during editing, will ensure that the status of any and all additional markers is collected and made known to the system.

Collect is also unlike the other options, in that once collection is complete, it leaves the cursor at the end of text, rather than where it was found.

HOW IT WORKS

Apart from the additional markers f1<n>f2, the program also makes use of two other special embedded commands.

f1@f2 is used to mark the cursor position during block operations so that it can be restored to its rightful place in text once an operation has been completed.

f1Bf2 is used to mark the positions of any real WW+ markers during block operations so that they can, if present, also be restored.

Because of this you should be careful not to leave the cursor or any WW+ markers in text to be manipulated.

PROCEDURE DEFINITIONS

You'll notice that all of the procedures have been defined prior to the main body of the program. This gives a slightly faster program; of no great importance in this application perhaps, but it's a point worth bearing in mind when developing your own segment programs.

INITIALISATION

When first invoked, by pressing Shift and the relevant function key, the variables used by the program are initialised, and the first time used flag, U%, is set to TRUE. When invoked subsequently, this flag stops the variables from being re-initialised.

If for any reason you need to re-initialise the marker system, simply type:

```
:U%=FALSE
```

from Menu mode, but be aware that this will also produce an acute attack of marker amnesia. This is easily cured by using the Collect Markers facility to find and record any markers in the text.

BELLS AND WHISTLES

Perhaps I shouldn't say this as it's rather tempting fate, but the program is reasonably fail-safe. It would be possible to include more error checking and correcting, but at the cost of extra code, and a slower program. I believe I've achieved the right balance between safety and size, but if you feel like adding your own features, go ahead.

For instance, as set up, the system works on text in the main text area. An obvious enhancement would be to make the working area selectable at run-time. Another extra might be a Delete All Markers option. However, each extra feature means more code and less space for word processing.

I have, however, included one or two bells and whistles, which strictly speaking do nothing extra for the program, except make it slightly more interesting to use. Error messages are displayed at the bottom of the screen for a brief period in a contrasting display of blue on white, and one of three possible attention getting sounds; either the standard VDU 7 beep, a gentle alarm, or a real 'wake-em-up!' effect. The required sound is determined by setting N% to either 0, 1, or 2.

PROGRAM SIZE

The complete program as printed here occupies some 5K bytes of WW+ workspace. I've also included two other versions of the program on this month's disc. The first, MARKSA, has extensive REMs to explain how it works. The second, MARKSC has been compressed as much as possible to save space and run faster.

```
IF U%=TRUE THEN GOTO main-program
U%=TRUE
F%=0
L%=0
S%=0
M%=0
M$=""
A%=0
G$=CHR$(2)
W$=CHR$(7)
C$=G$+"@"+W$
B$=G$+"B"+W$
N%=0
GOTO main-program
.warning
IF N%=1 THEN GOTO beep
IF N%=2 THEN GOTO whoop
VDU7
ENDPROC
.beep
REPEAT
*FX214,4
*FX213,150
VDU7
*FX213,120
VDU7
TIMES 2
GOTO warning-end
.whoop
*FX214,1
REPEAT
I%=100
REPEAT
OSCLI("FX213,"+STR$(I%))
VDU7
I%=I%+1
UNTIL I%=120
TIMES 2
.warning-end
*FX214,7
*FX213,100
ENDPROC
.message
VDU31,39,23
VDU131,157,132
PRINT T$;
PROCwarning
TIME=0
REPEAT
UNTIL TIME=250
VDU31,?126,11
ENDPROC
.large-msg
REPEAT
VDU131,141
PRINT T$
TIMES 2
ENDPROC
.space-line
PRINT
```

```
VDU134
ENDPROC
.restore-cursor
CURSOR TOP
REPLACE C$,""
ENDPROC
.exit
*FX21,0
*FX138,0,27
*FX138,0,27
VDU23;11,255;0;0;0
*FX229,0
END
ENDPROC
.set-and-value
A%=1
IF M%=0 THEN ENDPROC
REPEAT
A%=A%*2
TIMES M%
ENDPROC
.not-found
E%=TRUE
T$="Marker "+M$+" not found"
PROCmessage
ENDPROC
.find-marker
E%=FALSE
FIND P$
IF EOT THEN PROCnot-found
ENDPROC
.test-marker
S%=0
PROCset-and-value
IF (F% AND A%) >0 THEN S%=S%+1
IF (L% AND A%) >0 THEN S%=S%+1
ENDPROC
.validate-markers
PROCtest-marker
IF S%=2 THEN GOTO validate-find
IF S%=0 THEN T$="No markers "+M$
IF S%=1 THEN T$="Only one marker "+M$
PROCmessage
PROCexit
.validate-find
TYPE C$
CURSOR TOP
S%=0
REPEAT
PROCfind-marker
IF E%<>TRUE THEN S%=S%+1
IF E%=TRUE THEN GOTO validate-find-
restore
CURSOR RIGHT
TIMES 2
.validate-find-restore
PROCrestore-cursor
IF S%<>2 THEN PROCexit
ENDPROC
.set-first
F%=F%+A%
TYPE P$
PROCexit
ENDPROC
.set-last
L%=L%+A%
TYPE P$
PROCexit
ENDPROC
.select-marker
PRINT
PRINT "Select Marker (0-9)";
REPEAT
M%=GET
IF M%>=48 THEN M%=M%-48
UNTIL M%>=0 AND M%<=9 OR M%=27
IF M%=27 THEN PROCexit
M$=STR$(M%)
P$=G$+M$+W$
ENDPROC
.display-markers
PRINT
M%=0
REPEAT
PROCspace-line
PRINT "Marker "+STR$(M%)+"    ";
PROCtest-marker
PRINT STR$(S%)+" set"
M%=M%+1
UNTIL M%>9
PROCselect-marker
ENDPROC
.set-markers
CLS
T$="         Set Markers Menu"
PROClarge-msg
```

```
PROCdisplay-markers
PROCtest-marker
IF S%=0 THEN PROCset-first
IF S%=1 THEN PROCset-last
T$="Both markers "+M$+" already set"
PROCmessage
GOTO set-markers
ENDPROC
.delete-markers
CLS
T$="       Delete Markers Menu"
PROClarge-msg
PROCdisplay-markers
TYPE C$
CURSOR TOP
REPEAT
REPLACE P$,""
UNTIL EOT
PROCset-and-value
IF (F% AND A%) >0 THEN F%=F%-A%
IF (L% AND A%) >0 THEN L%=L%-A%
PROCrestore-cursor
GOTO delete-markers
ENDPROC
.mark-real
TYPE C$
CURSOR TOP
B%=0
REPEAT
FIND MARKERS
IF EOT THEN GOTO mark-real-loop
B%=B%+1
DELETE AT
TYPE B$
.mark-real-loop
UNTIL EOT
ENDPROC
.restore-real
IF B%=0 THEN ENDPROC
S%=0
CURSOR TOP
REPEAT
FIND B$
IF EOT THEN GOTO restore-real-loop
S%=S%+1
DELETE AT 3
IF S%<3 THEN FKEY3
.restore-real-loop
UNTIL EOT
ENDPROC
.mark-block
CURSOR TOP
REPEAT
PROCfind-marker
DELETE AT 3
FKEY3
TIMES 2
ENDPROC
.copy-block
CLS
T$="         Copy Marked Block"
PROClarge-msg
PROCselect-marker
PROCvalidate-markers
PROCmark-real
PROCmark-block
PROCrestore-cursor
FKEY9
TYPE C$
CURSOR TOP
REPEAT
FIND MARKERS
DELETE AT
TYPE P$
TIMES 2
PROCrestore-real
PROCrestore-cursor
PROCexit
ENDPROC
.delete-block
CLS
T$="        Delete Marked Block"
PROClarge-msg
PROCselect-marker
PROCvalidate-markers
PROCmark-real
PROCmark-block
DELETE MARKED
PROCrestore-real
PROCset-and-value
F%=F%-A%
L%=L%-A%
PROCrestore-cursor
PROCexit
```

```
ENDPROC
.find-block
CLS
T$="        Find Marked Block"
PROClarge-msg
PROCselect-marker
PROCfind-marker
PROCexit
ENDPROC
.action-markers
CLS
T$="        Action Markers Menu"
PROClarge-msg
PRINT
PROCspace-line
PRINT " 1 Find Marked Block"
PROCspace-line
PRINT " 2 Delete Marked Block"
PROCspace-line
PRINT " 3 Copy Marked Block"
PROCspace-line
PRINT "ESC Edit Mode"
PRINT
PRINT "Please enter choice ";
REPEAT
R%=GET
IF R%>=48 THEN R%=R%-48
UNTIL R%>0 AND R%<4 OR R%=27
IF R%=27 THEN PROCexit
IF R%=1 THEN PROCfind-block
IF R%=2 THEN PROCdelete-block
IF R%=3 THEN PROCcopy-block
ENDPROC
.collect-markers
F%=0
L%=0
T$="Looking for markers - please wait"
PROCmessage
CURSOR TOP
REPEAT
FIND G$
IF EOT THEN GOTO collect-end
CURSOR RIGHT
M%=VAL(GCT$)
IF EOT THEN GOTO collect-end
IF M%>9 THEN GOTO f1-search
M$=GCT$
IF M$<>W$ THEN GOTO f1-search
PROCset-and-value
IF (L% AND A%)=0 THEN GOTO f1-first-
check
T$="Too many markers "+STR$(M%)+"  (15
   spaces)  "
PROCmessage
T$="Looking for markers - please wait"
PROCmessage
GOTO f1-search
.f1-first-check
I%=(F% AND A%)
IF I%=0 THEN F%=F%+A%
IF I%<>0 THEN L%=L%+A%
.f1-search
UNTIL EOT
.collect-end
PROCexit
ENDPROC
.main-program
VDU23;11,0;0;0;0
SELECT TEXT
CLS
*FX229,1
T$="    Multiple Markers Main Menu"
PROClarge-msg
PRINT
PROCspace-line
PRINT " 1 Set Markers"
PROCspace-line
PRINT " 2 Delete Markers"
PROCspace-line
PRINT " 3 Action Markers"
PROCspace-line
PRINT " 4 Collect Markers"
PROCspace-line
PRINT "ESC Edit Mode"
PRINT
PRINT "Please enter choice ";
REPEAT
R%=GET
IF R%>=48 THEN R%=R%-48
UNTIL R%>0 AND R%<5 OR R%=27
IF R%=27 THEN PROCexit
IF R%=1 THEN PROCset-markers
IF R%=2 THEN PROCdelete-markers
IF R%=3 THEN PROCaction-markers
IF R%=4 THEN PROCcollect-markers
```

HINTS and tips HINTS and tips HINTS and tips HINTS and tips HINTS and tips

Please keep sending in your hints on anything relevant to the BBC and Master computers. Don't forget, we pay for all hints we publish.

TEXT FILES VERSUS TEXT TO BASIC

N.P.Toft

Without wishing to diminish Jack Phillips' ingenious *Text to Basic* program in the March BEEBUG, may I submit my own, possibly simpler solution to the problem of displaying word processed text from a program.

Spooling text out from a word processor produces a text file that can be displayed from a program using the *TYPE command. This method avoids having to juggle around with the Basic program, and has the advantage that the text takes up no memory in Basic. However, there must be a disc present when the text is required, which can be a drawback.

For 40 column modes, a line length of 39 characters is required (to accommodate the end of line character), and 80 column modes require lines of 79 characters. Embedded commands should be omitted, as well as any printer highlight codes.

VIEW HINTS

Elaine Kemp

In View, the Format command reformats all the text. If you wish to protect addresses on the left hand side from being formatted then you can use the LJ stored command. If, however, you have set a left margin, then LJ sets the characters to the extreme left, not to the set margin. To overcome this problem, the View manual suggests you insert a margin without end markers above the text, but an easier method is not to use the LJ command and to put a Tab character at the end of each line that you don't want formatted.

If you wish to use the graphics character set on your printer, first select the alternative font by sending the appropriate code (as described in your printer manual), and then send the code for setting the top bit of data sent to the printer. You can then incorporate graphics symbols into your text files.

If you use the ADFS filing system, then it is a good idea to include the following key definition in your !Boot file:

```
*KEY 0 *DIR $|M*CAT|M
```

This will mount and catalogue the current drive. If you have a disc toolkit ROM such as ADT, then *CAT can be more usefully replaced by *AMENU (for example).

CLEARING THE ASH TRAY

David Fairhurst

Having 18 BBC computers in one classroom causes several problems. One of these is the frequency with which the children push sweet papers and other objects into the "ash tray" speaker grilles. I tried quite a few ways of bunging them up until I hit upon the idea of using dead phone cards.

Cut about 6mm off the long edge of the card. Clean all the paint off with steel wool. Scour the back of the card with the steel wool and then stick it over the hole using a good quality super glue. The problem no longer exists, but do remember that the volume of the sound will be greatly reduced.

VIEW PROFESSIONAL MEETS THE MASTER ROM

J.Scott

In BEEBUG Vol.6 No.2 there were some loaders published for use with the Master ROM. As View Professional was not included, the following loader will work. Create the following file using *BUILD !MENUVP:

```
MODE 0
VDU 19,1,2,0,0,0
OSCLI("KEY0 *VP |M\L "+$&100+"|M")
*FX 138,0,128
```

The second line is optional - it simply gives green writing on a black background. On the BBC B !MENUVP will reside in the $ directory, and on the Master with ADFS it must reside in every directory in which there are View Professional files. B

Mr Toad's Keyboard Beep ROM (continued from page 29)

```
1190 .lectio
1200 JSR osrdch:ORA #&20
1210 CMP hi:BCS lectio
1220 CMP lo:BMI lectio
1230 JSR oswrch:RTS
1240 :
1250 .fiat
1260 PHA:PHX:PHY
1270 LDA #&FF:LDX flag
1280 STA &02D7,X:STA &02E0,X
1290 LDA #7:JSR oswrch
1300 PLY:PLX:PLA
1310 JMP (oldVec)
1320 :
1330 .mandata
1340 EQUB &0D
1350 EQUS"Select PIP or SQUEAK "
1360 BRK:EQUW &0D0D
1370 EQUS"Select beep VOLUME 1-5 "
1380 BRK:EQUW &0D0D
1390 EQUS"Select beep PITCH 0-9 "
1400 BRK:EQUW &0D0D
1410 EQUS"Select beep DURATION 1-3 "
1420 BRK
```

```
1430 :
1440 .oldVec
1450 BRK:BRK
1460 .chan
1470 BRK
1480 .vol
1490 BRK
1500 .pitch
1510 BRK
1520 .dur
1530 BRK
1540 .flag
1550 BRK
1560 ]:NEXT
1570 :
1580 FOR N%=7 TO 4 STEP-1
1590 IF N%?&2A1 NEXT:PRINT'"Sorry - no
free SRAM slot.":END
1600 OSCLI "SRWRITE "+STR$~Z%+" "+STR$~
(O%+1)+" 8000 "+STR$N%
1610 N%?&2A1=&82
1620 PRINT'"BEEP ROM running in slot ";
N%
1630 N%=4:NEXT:END
```

POISSON DISTRIBUTION

Since I last wrote, I have been doing some work with Poisson random number generators. Knuth's book *Semi-Numerical Algorithms*, 2nd Ed., Addison-Wesley, gives several algorithms for generating Poisson distributed random variates. I have also found a method of implementing the cumulative distribution method for the Poisson distribution which is quite fast. This uses the recursive relations for the Poisson probabilities (p(n), and the cumulative Poisson probabilities P(n):

$$p(0)=\exp(-\mu)$$
$$p(n)=p(n-1).\ \mu\ /n$$
$$P(0)=p(0)$$
$$P(n+1)=P(n)+p(n)$$

Ron Larham

The BEEBUG Workshops in Vol.10 No.4 on Random Sampling, and Vol.10 Nos.5 to 8 on Simulation Modelling referred to various random distributions. No algorithm was given for the Poisson distribution, so Ron Larham's letter makes a useful contribution to this series. See also the Postbag pages in some of the same issues for further reader input.

FAMILY TREE PROGRAMS AND DATABASES

In response to Mr. Pope's letter in Postbag Vol.10 No.9 I would like to point out that I have recently sent off a disc called *Ancestry* to BBC PD (who advertise in BEEBUG). There will be a utilities disc soon to go with it. Ancestry is compressed and the disc contains a program to decompress the application which then requires four discs.

Silas Brown

I use Masterfile II for general sorting of records and find it efficient and easy to use. I use it for lots of genealogical applications (parish register entries, Mormon index, etc.) and for general purpose record keeping.

With regard to the Micro-Aid program, *The Family History System*, I use it as my main 'tree' program for producing family trees. It is simple to use, and has a wide range of printouts, although as a sortable database it is less impressive. On a model B, it holds 200 family groups with 350 people, and on a Master 400 family groups with 750 people.

With regard to a more complex application, including free format searchable text, I use *Texbase* from BEEBUG Vol.10 Nos.4, 5 & 6. It is extremely flexible with 21 lines of text space per page, from which keywords can be selected and used by the search procedure. I use it for all transcription work and general notes.

Finally, is there a program available which will automatically grab DFS disc information and save it as part of a database disc catalogue?

Robert Clayton

We publish an indexing system which might be of help to Mr. Pope's indexing problem. *JISYS* is a journal indexing system, but could well be applied to other items such as slides. This costs £45.00 and further details are available from ourselves.

Kenneth Spencer
KAS Software, 74 Dovers Park, Bathford,
Nr. Bath, Avon BA1 7UE, tel. (0225) 858464.

Note, Masterfile II is available from Beebug for £22.48 inc. VAT, plus £2.00 p&p. Micro-Aid are at Kildonan Courtyard, Barrhill, S. Ayrshire, Scotland KA26 0PS, tel. (0465) 82288. The complete Texbase program is available on BEEBUG magazine disc Vol.10 No.6 for £4.75 inc. VAT plus £1.00 p&p. For DFS disc catalogue information we recommend Indexing DFS Format Discs in BEEBUG Vol.7 No.10, or Disc File Identifier in Vol.8 No.6. B

RISC USER

The Archimedes Magazine & Support Group

RISC User continues to enjoy the largest circulation of any subscription magazine devoted solely to the Acorn Archimedes range of computers. Its in-depth, authorative approach appeals to all users, whatever their interest and level of expertise, and the lively mixture of articles, programs, reviews and news is carefully selected to cater for all Archimedes owners from beginner to expert.

Existing BEEBUG members, who want to find out more about the Archimedes range, may either transfer their existing subscription to RISC User (at no extra charge), 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 developers for both the BBC micro and the Archimedes range. RISC User is the magazine for enthusiasts and professionals at all levels.

The Archimedes Magazine & Support Group

Here are some articles and series published in the most recent issues of RISC User:

ARMED AND READY FOR BATTLE
A fascinating investigation of the latest developments in RISC technology and the ARM processor.

ALEPH ONE PC BOARD
A review of the first 386 PC card for the Archimedes.

INFORMER
A RISC OS compliant full feature card index database.

GRAPHBOX PROFESSIONAL
A review of the most comprehensive and sophisticated graph application for the Archimedes.

COLOUR PRINTING WITH THE DESKJET 500C
A look at what is being heralded as a revolution in low cost colour printing.

CREATING DRAW FILES IN BASIC
A two-part arfticle on how to use Basic to create Draw files.

EXPLOITING THE WIMP
A series on some practical aspects of WIMP programming.

DESKTOP ANIMATOR
A tutorial mini-series on how to program moving images within a Desktop window.

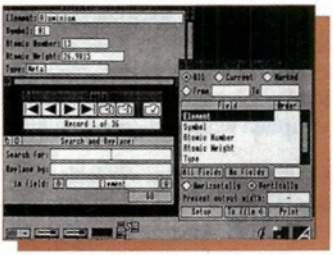
WRITE-BACK
New readers' section of RISC User for comment, information, help - a magazine version of a bulletin board.

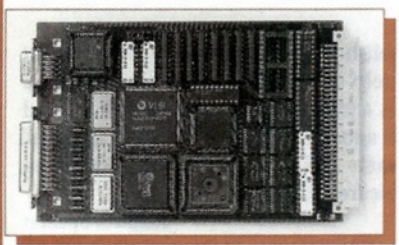
USING ANSI C
A series of articles on programming Desktop applications in C.

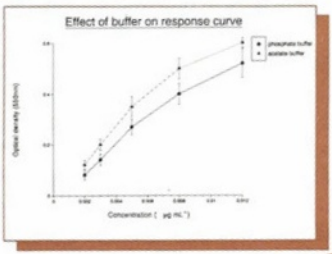
WP/DTP
A regular column on using different DTP and WP packages.

INTO THE ARC
A regular series for beginners currently treating the subject of using shared resources.

ARCADE
A look into the latest games for the Archimedes.







SUBSCRIPTION DETAILS

As a member of BEEBUG you may extend your subscription to include RISC User for only:

Destination	*Additional Cost*
UK,BFPO &Ch Is	£ 10.50
Rest of Europe and Eire	£ 15.40
Middle East	£ 19.60
Americas and Africa	£ 21.90
Elsewhere	£ 33.00

Personal Ads

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

We also accept members' Business Ads at the rate of 40p per word (inclusive of VAT) and these will be featured separately. Please send all ads (personal and business) to MEMBERS' ADS, BEEBUG, 117 Hatfield Road, St. Albans, Herts AL1 4JS.

Mexican built BBC B with View, Speech, NFS, 40/80T disc drive, CUB colour monitor, bound manuals, joystick as new £300, CST Procyon IEEE interface £15, 12" NEC mono monitor £30, A310 base unit & keyboard £380, Wordwise ROM £12, mouse as new £30, BeebDOS and BeebPC £20 each. Tel. (0483) 480632.

BBC Master 128 £250, Turbo board (65c 1002 second processor) £50, Master 512 second processor with Essential Software's memory upgrade, can be used with master or BBC, Dos Plus version 1.2 and 2.1 Gem software and mouse £175, Dabs Press 512 shareware collections 1&2 £10, Dabs Press Sidewriter (5.25 40/80T) £5, Dabs Press Hyperdriver ROM and disc £15, C.C Print Master ROM £25, C.C Interchart ROM £15, C.C Interbase ROM £30, C.C Spellmaster ROM £25, StarBase Database ROM and utilities disc £10, Acorn Overview 1&2 (all the View family plus manuals for Master or BBC with View and Viewsheet) £50, Acorn View printer driver generator £5, Acorn View index (5.25 40/80T) £5, Three Master ROM cartridges (full height) £5 each, BBC Soft Vu-Type Professional £7.50, Fourth Dimension Holed Out Golf £7.50, Acorn Let's Count (5.25 40/80T) £2, Acorn Juggle Puzzle (5.25 40/80T) £2, Acorn Workshop (5.25 40/80T) £2, Acorn Picture Maker (5.25 40/80T) £2, 10 BBC B games on tape £5, BBC User Guide £2.50, Master reference manuals 1&2 £5 each, Advanced reference manuals for BBC Master £7.50, Dabs Press Master operating system £5, Dabs Press Master 512 User guide & disc £7.50, Dabs Press Master reference guide & disc £7.50, Mastering Dos Plus £5, Glentop Dos Plus reference guide £10. All software is original and both software and hardware comes with guides and manuals, will sell separately or as a job lot. Make me an offer. Tel. 081-684 9340 eves & w/ends.

BBC 512 co-processor & Essential's memory expansion, Original system discs & Essential's CLmouse driver, RAMdisc, FSTboot & miscellaneous discs, also Tull mouse driver, Shibumi Problem Solver & Essential's CPFS ROM, complete with User Guides £200, Morley BBC SCSI hard disc control card & utilities disc £60. Tel. 071-543 7800.

BBC B issue 4, with 128 solidisc SWR, 40T SS DD plus 40/80 DS drive, both with own PSU, BBC teletext adaptor, Music 500 (5000), dozens of ROMs, 100's of discs, 100's of books and magazines, all for £450, Ferguson TX14" TV with RGB input £90, can deliver within reasonable area, otherwise plus carriage. Tel. 041-887 7200.

M128 reference manuals 1&2, View & Viewsheet guides £12, Micropulse ROM-Box £15, dual BBC/Master 128 Voltmace joysticks £12, Master Compact joystick (Voltmace) £7, MAX 16k EPROM £2, four used 32k EPROMs £2.50 each, BBC ROM extension lead, works on Master 128 or Compact £8. Tel. (0332) 572009.

BBC B Microvitec med. res. colour monitor, Opus 40/80T DD with own PSU, AMX Mouse, AMX Pagemaker, AMX Super-Art, View, View -Store, ViewSheet, DataBase, User Guide, 30 Hour Basic, Elite £300. Tel. (0249) 816463.

M128 computer and disc drive £250, Canon BJ10ex printer £200 o.n.o. BEEBUG internal modem for M128 £60, BEEBUG magazines £6 a volume, Micro User £8 a volume, Mini Office II (BBC 80T) £6, Dabs View book with disc £11, Dabs Master OS book £6. **WANTED:** Ground control teletext adaptor. Tel. 051-606 0289.

CommStar ROM, Fontaid ROM/discs, Sanyo DR101, Extra! Extra!, Genie ROM, Masterfile II, Graphics discs, Gamma System ROM/discs, printer KP810, Publisher ROM, Signwriter 10 fonts, Viewsheet, Viewindex, Viewstore ROM, ROM cartridges, plus manuals. Tel. (0293) 535229.

WANTED: BBC JP101 printer in working order with cable for BBC micro. Tel. (0747) 55006.

512 co-processor for Master 128 with mouse, DOS 2.1, in excellent condition £100 o.n.o. Tel. (0324) 38816 after 6pm.

WANTED: Acorn ADFS book. Tel. (0293) 529129 after 6pm.

Master 512, Microvitec medium resolution monitor, twin 40/80T drives with PSU, teletext adaptor, mountains of software (games, education, business), ROMs, all manuals plus many extra books £450. Tel. (0222) 865248 for full details.

Modem - Pace Nightingale V21/23 with BEEBUG Command communications ROM (scrolling text and Viewdata), plus data cable for BBC computer, everything in mint condition, with original instruction manuals £40, inc. p&p. Tel. (0294) 52250.

Morley teletext adaptor £30, Spellmaster £20, Printmaster (EPROM) £7, Watford NLQ (EPROM) ROM £7, Master control panel and 1770 DFS ROMs by ACP £7 each, Signwriter software including Icon and Xmas font discs £12, Full set BEEBUG £40, Master reference manuals £12 pair, complete BBC user book £5, Mastering Assembly Code £5, BEEBUG vols 2 to 5 plus part vols 1 and 6. Tel. (0276) 20193 after 6pm.

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
£18.40	1 year (10 issues) UK, BFPO, Ch.I	£28.90
£27.50	Rest of Europe & Eire	£42.90
£33.50	Middle East	£53.10
£36.50	Americas & Africa	£58.40
£39.50	Elsewhere	£62.50

BACK ISSUE PRICES (per issue)

Volume	Magazine	5"Disc	3.5"Disc
6	£1.00	£3.00	£3.00
7	£1.10	£3.50	£3.50
8	£1.30	£4.00	£4.00
9	£1.60	£4.75	£4.75
10	£1.90	£4.75	£4.75
Binders	£4.20		

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	£ 1.00	£ 0.50
Europe + Eire	£ 1.60	£ 0.80
Elsewhere	£ 2.40	£ 1.20

BEEBUG
117 Hatfield Road, St.Albans, Herts AL1 4JS
Tel. St.Albans (0727) 40303, FAX: (0727) 860263
Manned Mon-Fri 9am-5pm (for orders only 9am-6pm and 9am-5pm Saturdays)
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by RISC Developments Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Technical Editor: Mark Moxon
Editorial Assistance: Marshal Anderson
Production Assistant: Sheila Stoneman
Advertising: Sarah Shrive
Subscriptions: Sue Baxter
Managing Editor: Sheridan Williams

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

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

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

In all communication, please quote your membership number.

Printed by Arlon Printers (0923) 268328 ISSN - 0263 - 7561

Magazine Disc

MAY 1992
DISC CONTENTS

DATA SHEET - The first part of this powerful Basic spreadsheet program, which implements the main formula and cell functions.

TABLE FORMATTER - A program that enables tables to be produced automatically from Basic, making table production much easier than in a word processor.

MR TOAD'S BEEP ROM - A program to assemble a sideways ROM image which enables each keypress to emit a beep, thus making fast typing easier. There are a number of sounds that can be used, all accessed through one star command.

FUNCTION/PROCEDURE LIBRARY 11 - A set of routines for use with sideways RAM, to enable integers, strings and blocks of memory to be stored and retrieved in this useful extra memory.

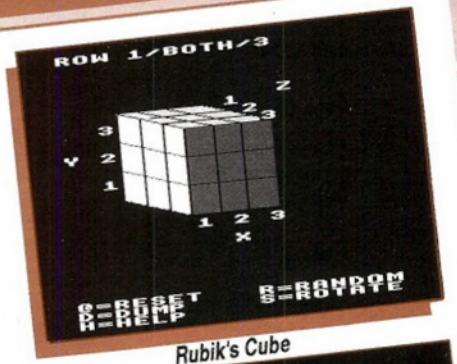
WORDWISE USER'S NOTEBOOK - Three Wordwise Plus segment programs that implement extended markers for use when word processing. The segment programs all do the same, but one is compressed to save memory, and one has extensive comments to help explain the methods used.

MODE 0 SCREEN DUMPS - Two screen dump programs for printing mode 0 screens on a printer. One prints the image the correct way up, and the other prints the image sideways.

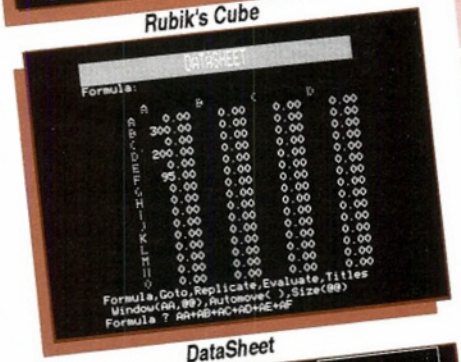
RUBIK'S CUBE - A bonus item for Master owners that implements the Rubik's cube puzzle. The cube is shown on screen, and can be jumbled up to provide a real challenge.

BEEBUG VOLUME 10 BIBLIOGRAPHY - The complete Volume 10 bibliography file for use with MagScan.

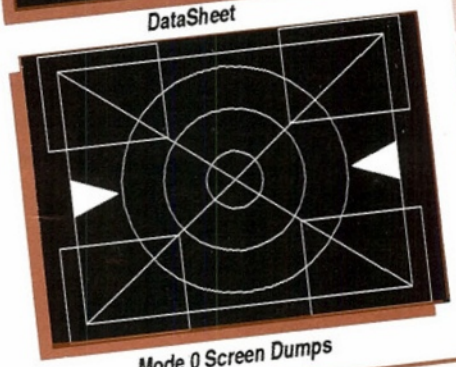
MAGSCAN DATA - Bibliography for this issue (Vol.11 No.1)



Rubik's Cube



DataSheet



Mode 0 Screen Dumps

ALL THIS FOR £4.75 (5.25" & 3.5" DISC) + £1 P&P (50p FOR EACH ADDITIONAL ITEM)

Back issues (5.25" and 3.5" discs from Vol.5 No.1) available at the same prices.

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

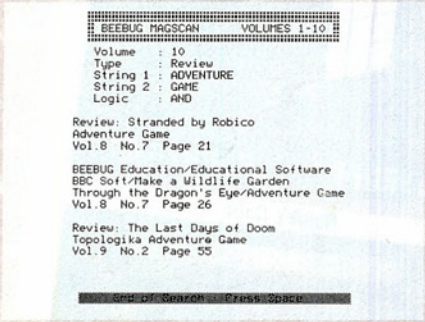
Prices are inclusive of VAT and postage as applicable. Sterling only please

RISC Developments, 117 Hatfield Road, St.Albans, Herts AL1 4JS

Magscan

Comprehensive Magazine Database for the BBC Micro and the Master 128

An updated version of Magscan, which contains the complete indexes to all BEEBUG magazines from Volume 1 Issue 1 to Volume 10 Issue 10



Magscan with disc and manual **£9.95**+p&p

Stock codes:	0005a	5.25"disc 40 track DFS
	0006a	5.25"disc 80 track DFS
	1457a	3.5" ADFS disc

Magscan update **£4.75** +p&p

Stock codes:	0011a	5.25"disc 40 track DFS
	0010a	5.25"disc 80 track DFS
	1458a	3.5" ADFS disc

Magscan allows you to locate instantly all references to any chosen subject mentioned anywhere in the 95 issues of BEEBUG magazine to date.

Just type in one or two descriptive words (using AND/OR logic), and you can find any article or program you need, together with a brief description and reference to the volume, issue and page numbers. You can also perform a search by article type and/or volume number.

The Magscan database can be easily updated to include future magazines. Annual updates are available from BEEBUG for existing Magscan users.

Some of the features Magscan offers include:

- full access to all BEEBUG magazines
- rapid keyword search
- flexible search by volume number, article type and up to two keywords
- keyword entry with selectable AND/OR logic
- extensive on-screen help
- hard copy option
- easily updatable to include future magazines
- yearly updates available from BEEBUG

Special Offers to BEEBUG Members May 1992

1407a	ASTAAD3 - 5" Disc (DFS)	5.95
1408a	ASTAAD3 - 3.5" Disc (ADFS)	5.95
1404a	Beebug Applics I - 5" Disc	4.00
1409a	Beebug Applics I - 3.5"Disc	4.00
1411a	Beebug Applics II - 5" Disc	4.00
1412a	Beebug Applics II - 3.5" Disc	4.00
1405a	Beebug Utilities - 5" Disc	4.00
1413a	Beebug Utilities - 3.5" Disc	4.00
0005b	Magscan Vol.1 - 8 40 Track	9.95
0006b	Magscan Vol.1 - 8 80 Track	9.95
1457b	Magscan Vol.1 - 8 3.5" ADFS	9.95
0011a	Magscan Update 40 track	4.75
0010a	Magscan Update 80 track	4.75
1458a	Magscan Update 3.5" ADFS	4.75
PAG1a	Arcade Games (5.25" 40/80T)	5.95
PAG2a	Arcade Games (3.5")	5.95
PBG1a	Board Games (5.25" 40/80T)	5.95
PBG2a	Board Games (3.5")	5.95

All prices include VAT where appropriate

1600a	Beebug magazine disc	4.75
0077b	C - Stand Alone Generator	14.56
0081b	Masterfile ADFS M128 80 T	16.86
0024b	Masterfile DFS 40 T	16.86
0025b	Masterfile DFS 80 T	16.86
0074b	Beebug C 40 Track	45.21
0075b	Beebug C 80 Track	45.21
0084b	Command	29.88
0073b	Command(Hayes compatible)	29.88
0053b	Dumpmaster II	23.76
0004b	Exmon II	24.52
0087b	Master ROM	29.88
1421b	Beebug Binder	4.20

P & P	UK	Europe	Americas, Africa Mid. East	Elsewhere
a	£ 1.00	£ 1.60	£ 2.40	£ 2.60
b	£ 2.00	£ 3.00	£ 5.00	£ 5.50

Carriage is donated by the letter after the stock code. When ordering several items use the highest price code, plus half the price of each subsequent code.

Have you got your BEEBUG Binder for Volume 11?

Only £4.20

RISC Developments Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS. Tel (0727) 40303 Fax (0727) 860263