BEEBUG

FOR THE BBC MICRO & MASTER SERIES

GOODBYE!

BEEBUG 1982-1994
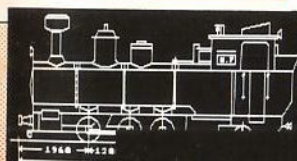
0-6-2 Locom and EVERYTHING

ELITE

THE M BBC MASTER SERIES MICROCOMPUTER

MELVYN WRIGHT'S MUSIC MACHINE

BEEBIOLI

BBC MICRO GRAPHICS AND SOUND

BBC BASIC

ACORN COMPUTER

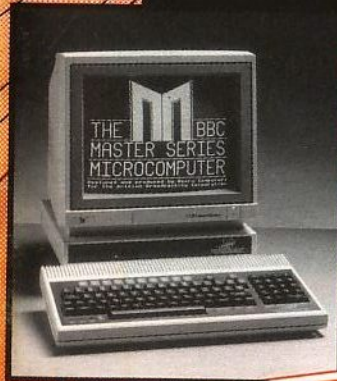- BACK TO THE FUTURE • MONITOR SCREEN SAVER
- INTER-BASE PROGRAMMING GUIDE • CUBERT GAME

BEEBUG

Vol.12 No.10 April 1994

Square Dance

A Simple Disc Menu
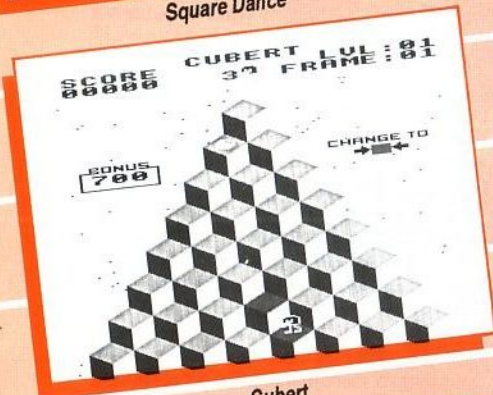
Cubert
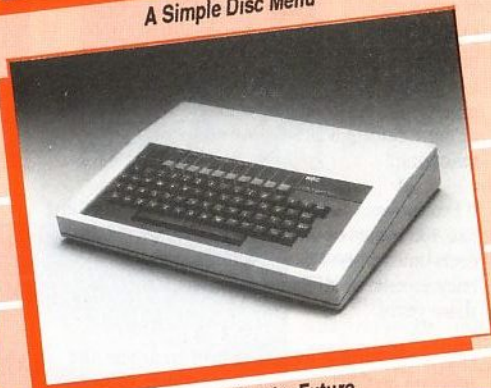
Back to the Future

Back to the Future

Back to the Future

Editor's Jottings/News

A Last Farewell

Ideas for this, my last editorial for BEEBUG, have been going through my mind over the past two weeks. What should I say, when there is so much that has happened over the last ten years or more during which I have edited the magazine?

If I count up, I have ben responsible for something like 105 issues of the magazine, and there have been times when I have felt like a walking encyclopedia to all that we have published in those years.

One of my earliest memories is of an article and program on CAD, with a feature which allowed *Any Size Text, at any Angle, Anywhere on the Drawing*. That was ASTAAD, a very popular program which was updated significantly on two separate occasions, and was eventually published as a stand alone package.

Another visual item I remember was a very realistic rendition of a World War II Spitfire which could be rotated about three axes, made all the more interesting as the author had himself been a Spitfire pilot.

Yet another item which caused a stir was a more technical one showing how screen displays could be split up into sections using different modes. Nobody thought it could be done until the game Elite arrived to prove differently, and then suddenly it was all the rage.

Among other novelties I recall were two competition entries. One, for converting Arabic to Roman numerals showed a man carving the Roman numerals over a Doric entrance to the realistic noises of a stonemason at work. Another, which played typical fairground organ music used the cassette relay to provide added percussion.

Then there were all the games: Hedgehog, Detonator Dan, Truffle Hunt, Kitchen Chaos, Wee Shuggy, George and the Dragon, Brickie Nickie, Builder Bob, Jason Mason, Pitfall Pete, and more recently Robol. In the early years we published two games in every issue.

And in addition to the many articles and programs, there were the reviews, sometimes of now long forgotten products - remember Phloopy, a tape system designed to function like a disc; all those add-on boards for Sideways RAM, Shadow RAM etc. There was just so much being produced in those early years; one wonders where it has all gone to now. Maybe there is somewhere an attic with a forlorn BBC micro and other bits and pieces.

Well there we are. It has been mostly good. We estimate that more than 60,000 people have subscribed to BEEBUG at one time or another, and approximately 150 have been faithful readers from Vol.1 No.1 right through to this final issue.

There are many people I have got to know by phone, by letter and sometimes face to face. Some of our more regular contributors became like personal friends, though many I never met. I wonder what they are doing now. And then there were the staff who have graced (!) the offices of BEEBUG over the years: Alan Webster - BEEBUG's first employee as Technical Assistant, David Fell who initiated our very long running Workshop series, Nige of the rainbow badge, Geoff Bains who went on to edit Acorn User and other magazines, Phyllida and Yolly who handled production in those early years and much more, and not forgetting our current supporting staff - Kristina, Marshal, Paul, Sarah, Sheila, and Tony, and BEEBUG's two founders, Sheridan Williams and Lee Calcraft. To them and to all of you, let me say "Thank you". For us, the end of BEEBUG really is the end of an era, and it is now to time to say a last farewell.
Mike Williams   B

Back to the Future

Mike Williams, editor of BEEBUG for the past ten and a half years takes a nostalgic trip back through the history of Acorn and the BBC micro.

It has been a long time, and much has changed since that time in 1983 when I joined BEEBUG as its first full time editor. However, it is not so much the history of BEEBUG with which I want to concern myself, but rather to reflect back on events both small and large within the Acorn world as a whole, events which I still recall with interest. The early history of BEEBUG itself has already been ably told by one of its founders, Lee Calcraft, in BEEBUG Vol.11 No.1. Lee now co-edits our magazine, RISC User, on a freelance basis, while

Hermann Hauser

Chris Curry

BBC Micro

at that time. Chris Curry and Hermann Hauser, the founders of Acorn were there as well, and I believe a youthful Bob Coates, now Acorn's International Business Director.

In fact, the adoption of Acorn's new computer as the BBC micro had come as something of a surprise to many watching earlier events. The BBC was keen to launch a computer literacy campaign, and wanted a micro with an advanced specification to be the focus of its endeavours. For long the favourite was considered to be the Newbury Newbrain -

BEEBUG's other founder, Sheridan Williams, continues as a Director of BEEBUG and RISC Developments with particular responsibility for BEEBUG and RISC User.

anyone remember that? But at the last minute it was Acorn which walked away with the prize.

I had my first introduction to the BBC micro in November 1981 in a hotel just outside Cambridge where I spent a fascinating weekend in the company of some 30 or so individuals, many of whom like myself were about to join the government's Microelectronics in Education Programme (MEP). We were let loose on a roomful of BBC micros. None of them had power supplies - groups of four machines were linked up to separate power units! John Coll, author of the original BBC Micro User Guide, was there to tell us how to program the machines, though only the grey covered provisional User Guides were available

In those days, of course, nobody used disc storage - it was all cassette tape - and I can still recall the anguished sounds that were emitted from many cheap cassette recorders as programs were loaded from cassette to memory. There was no way of storing any kind of directory on tape - you had to keep your own written record of what was on there, and record the counter reading too so that you could fast forward or reverse to get near the start point.

Anything up to five minutes was needed to load a long program. For many years with BEEBUG I personally spent hours in

darkened rooms at home late at night photographing the screen to provide the illustrations which we published in the magazine. The worst were usually games. First you had to load the game, usually quite a lengthy process to the accompaniment of the shrieks and groans from the cassette player to which I have already referred.

Acorn Electron

Then, of course, to get a decent picture it was necessary to play the game for a while - a start screen with a score of zero is rather too boring. Then, to freeze the game - no pause key in those days - hold down Break, and if you're lucky press the camera shutter at the same time! If you were unlucky, pressing Break corrupted the picture (and the program), and you had to start loading the game all over again. Too many frustrating evenings were eaten up in anguished frustration in this way!
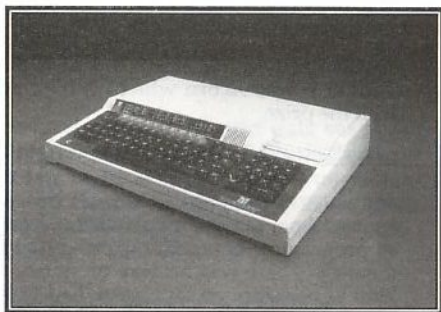

Master 128

Eventually, disc drives did appear, 40T 100K single sided, and 80T 400K double sided. If I remember correctly, an Acorn dual 80T drive (total capacity 800K) cost of the order of £800

- twice the cost of a BBC micro. Compare that with the cost of today's IDE hard disc drives at costs between £100 and £200 for 40Mb of storage or more!

The BBC micro turned out to be a huge success for Acorn. I believe original estimates of potential sales were of the order of 10,000. Eventually over 1.5 million BBC micros were sold, a fantastic achievement by what was originally a very small British company. Acorn's success seemed to know no bounds, and Acorn was soon involved in all manner of enterprises, some of somewhat dubious commercial value. Remember the Acorn sponsored racing car? I remember once being sent complimentary tickets to a race meeting, an invite I was unable to take advantage of at the time.
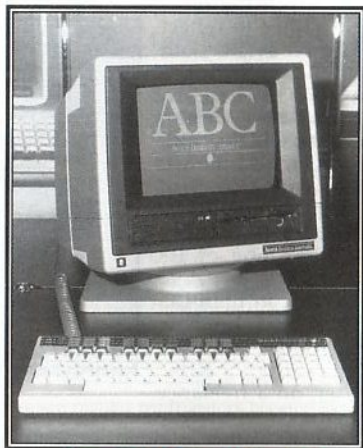
Acorn Business Computer

Those were the days of the lavish press launches: *Aviator* launched at the RAF Battle of Britain museum at Hendon, a database of cocktail recipes at a wine bar in Covent Garden (there was even a special Acorn cocktail in a rather lurid blue I remember), magic programs with Paul Daniels, the launch of Revs at the Steering Wheel Club and more. There was a time when that then doyen of games software houses, Micropower, would annually invite journalists and others to lunch at the Café Royal in London.

But all was not going well for Acorn. In an attempt to meet criticism that its machines

were too expensive Acorn launched the Electron in the summer of 1983. This was a cut-down version of the BBC micro but lacking one vital and essential feature, the ability to operate in mode 7. Before such sophistications as shadow RAM appeared, many programs for the BBC micro used mode 7 to conserve limited memory space. Consequently many of the most popular programs failed to run on the Electron, and despite its glitzy launch it was never a real success. Indeed, rumour has it that many were later dumped at low prices in Holland, and elsewhere.

Acornsoft - Elite

Another significant venture which sunk with even less trace was Acorn's ill-fated entry into the business market, the ABC (Acorn Business Computer). I have a copy of the glossy brochure in front of me as I write: systems ranged from the ABC Personal Assistant with 6502 processor and 640K bytes of floppy disc storage, up to the ABC 310 with 80286 main processor, 1Mb of main memory, 10Mb hard disc and "Desk Top Manager" - sounds not too dissimilar to another Acorn A310 which came along later, except that these were all based on 8-bit machines using 16-bit and 32-bit co-processors. As far as I know, none of these systems was ever sold, although they certainly existed in prototype form, and very classy they looked too. In 1985 Acorn announced the Cambridge Workstation based on the prototype ABC 210. This high specification machine (32016 32-bit second processor, 1Mb RAM, 10Mb hard disc storage) was aimed at scientific and industrial users, but with a price tag of £2700 was not an outstanding success.

One event of an entirely different nature which proved an enormous and long lasting success was the game of Elite released in the autumn of 1984. Written by two Cambridge graduates, David Braben and Ian Bell, Elite not only took the Acorn world by storm but captured the imagination of many other

An original style cassette player

computer users. Many were the hours wasted in battling through the grades of Harmless, Mostly Harmless, through to Dangerous, Deadly and hopefully Elite. It was a constant topic of conversation I remember in the editorial office. Versions appeared for the PC, and it lived on to be resurrected on Acorn's Archimedes range as well. It is difficult, perhaps, for those who did not experience it at first hand to recall the excitement and


Just a few of the many then available books for the BBC Micro

interest Elite generated. Not only was it a superb game, for its time, but it actually made CP/M and MS-DOS users really envious for once of us mere BBC micro mortals.

There were many other games, too, which captured the imagination. Do you remember *Chucky Egg, Frak, Defender* (one of the original Acornsoft games, and one of the best - this also saw an Archimedes version called *Planetoid*) and *Repton*? Then there was a whole rash of so-called adventure games, *Gateway to Karos, Countdown to Doom, Philosopher's Quest* and many more. We had our own adventurer in the pages of BEEBUG too, 'Mitch', who kept us entertained month after month with the escapades of his pet dragon, and much else besides.
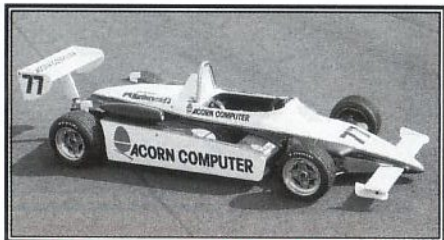
The Acorn sponsored Formula 3 racing car

There was a serious side to the BBC micro as well. Acorn provided not only BBC Basic, but Pascal, Comal, Prolog, Logo, BCPL and Lisp as programming languages - compare that to the current state of affairs for the Archimedes. Then there was the View family, Computer Concepts' rise to stardom with Wordwise (and Wordwise Plus) and then the Inter family, Beebugsoft's range including Exmon, Toolkit and Masterfile and so many more.

There were many hardware developments which expanded the original BBC micro way beyond its original capabilities. Sideways and then shadow RAM became almost de rigeur for the serious enthusiast, 6502 and Z80 second processors arrived to expand the Beeb's power and capability, and the Beeb's Teletext adaptor with the facility to download from Ceefax and Oracle was another popular innovation. This was particularly well supported by the BBC with a changing library of programs which users could download from Ceefax, until the BBC subsequently axed this facility.

Acorn eventually suffered a major financial collapse, and was only rescued from potential oblivion by Olivetti. Acorn's founders stayed on for a while but eventually left for relative obscurity, though Chris Curry retained a high profile for some while. While Acorn had stood still others had not, and the BBC micro was increasingly failing to compete. Revamped Acorn's first response was the interim B+ in June 1985, a pig's ear of a machine if ever there was one. Eventually a real replacement appeared in January 1986 in the form of the Master series which showed what the original BBC micro concept was capable of achieving. The Master 128 and later the Master Compact rescued Acorn from near terminal catastrophe, but despite their moderate success many felt, with some justification I think, that Acorn had very nearly left it too late. However, Acorn survived to fight another day, and eventually entered the 32-bit world in 1987 with the Archimedes range which exists today, but that is another story inappropriate to these pages at this time.

Cliff Michelmore and Wendy Craig launching Acorn's new home computer, The Electron.

This trawl through the history books has no doubt been highly selective, and I have probably omitted to mention far more than I have told. It has, however, brought back to me some of the heady excitement which many of us experienced in the early days of the BBC micro. Suddenly everybody wanted to know about computers; everyone wanted to learn how to program. The number of books published on BBC Basic were legion. As far as computing was concerned it was a period of adolescence, of learning, of experimenting, of enjoyment and growing up. The world of computing is older (and maybe wiser) today, and that is the way it should be, but I am glad that I had the opportunity to experience those more pioneering days at first hand.

B

Cubert

Can cuddly Cubert climb around the cubes cannily avoiding the crushing villain of piece as he goes? Tim Thornham sets the scene to his colourful arcade games.

Cubert is a fast, colourful, 3 dimensional, one player game based on the popular arcade game Q*BERT. The idea of the game is to move over a 'pyramid', made up of cubes, and to change the colour of each cube. At the same time you must save Cubert from being flattened by the ball that bounces randomly down the cubes from the top of the pyramid.
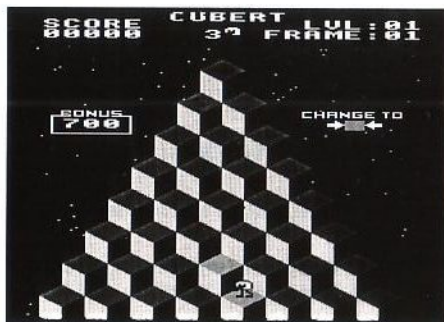

To complete a screen, all of the blocks in the pyramid need to be changed to the desired colour. Once this has been done, a new design of pyramid will appear,

requiring a different strategy to change the colour of the cubes.

Once all eight pyramids have been successfully negotiated, the game restarts from the first pyramid, but each block needs to be stepped upon twice to change it to the required colour. The target colour is displayed on the screen throughout the game.

Cubert can lose a life in one of two ways, either by falling off of the side of the pyramid, or by being flattened by a bouncing ball. Cubert is given five lives

to start the game with, and once these are lost, the game will end.
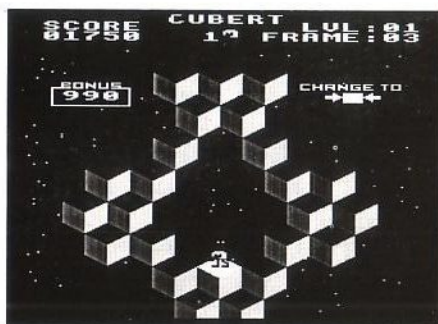
The quicker each screen is completed, the more bonus points are awarded. Every 5,000 points achieved brings an extra life for troubled Cubert in an attempt to reach the top of the high score table. Bonus points are displayed to the left of the pyramid, and count down as time elapses.

To control Cubert, four direction keys are needed. To move up and left (remember you are playing in 3D) you need to press 'A'. Pressing '*' will move Cubert up and

right, 'Z' moves down and left and '?' moves down and right.

If you wish to run Cubert on any system where PAGE is set higher than &E00

(e.g.DFS disc systems), then you will need to include the following movedown routines with the program. Please make sure that the program has been saved first.

```
0 IF PAGE<&E01 THEN 10
1 *TAPE
2 *KEY 0 FOR A%=0 TO (TOP-PAGE) STEP4

:A%!&E00=A%!PAGE:NEXT|MPAGE=&E00|M
OLD|MRUN|M
3 *FX138,0,128
4 END
```

Please be careful when typing in the character definitions and data from line 2690 onwards, otherwise the screen display may appear garbled.

```
10 REM Program CUBERT
20 REM Version B0.1
30 REM Author   Tim Thornham
40 REM BEEBUG   April 1994
50 REM Program subject to copyright
60 ::S%=0:ON ERROR GOTO 3220
110 MODE2:PROCinit:PROCchars:REPEAT:FR
%=1:PROChigh:PROCnew:REPEAT:IF E%=1 PROC
screen
170 VDU5:E%=1:REPEAT:PROCman:PROCball(
0):IF K>0 PROCball(1)
220 PROCman:IF K>1 PROCball(2)
240 PROCdelay(100):PROCman:J%=J% EOR1:
IF J%=1:PROCbonus
270 PROCscore:UNTIL noleft=0 OR E%=0:I
F E%=0 PROCrestart:ELSE S%=S%+B%:PROCsco
re:PROCtune
300 UNTIL L%=0:PROCdead:UNTIL FALSE:EN
D
340 :
1000 DEFPROCman:IF noleft=0 OR E%=0:END
PROC
1020 IF INKEY(-66):PROCpman(-72,104):EN
DPROC
1030 IF INKEY(-98):PROCpman(72,-104):EN
DPROC
1040 IF INKEY(-73):PROCpman(72,104):END
PROC
1050 IF INKEY(-105):PROCpman(-72,-104):
ENDPROC
1060 PROCdelay(100):ENDPROC
1080 :
1090 DEFPROCpman(x%,y%):VDU5:SOUND2,2,5
0,1:MOVEX%+64,Y%:PRINTCHR$127+B$(3+(POIN
T(X%-8,Y%)>9)+2*(POINT(X%+72,Y%)>9))+CHR
$(POINT(X%,Y%-64))+J$
1120 X%=X%+x%:Y%=Y%+y%:W%=POINT(X%,Y%-4
8):IF W%<8:E%=0:ENDPROC
1150 IF W%<D%:S%=S%+10:GCOL0,W%+1:MOVEX
%-36,Y%-52:MOVEX%+28,Y%-84:PLOT85,X%+28,
Y%-20:MOVEX%+36,Y%-84:MOVEX%+28,Y%-20:PL
OT85,X%+100,Y%-52:IF W%+1=D%:noleft=nole
ft-1
1160 MOVEX%,Y%:PRINTM$:ENDPROC
1180 :
1190 DEFPROCball(F%):IF noleft=0 OR E%=
0:ENDPROC
1210 SOUND1,1,1,1:MOVEB%(F%,0),B%(F%,1)
:GCOL0,POINT(B%(F%,0),B%(F%,1)):PRINTCHR
$250:B%(F%,1)=B%(F%,1)-104:P%=-(POINT(B%
(F%,0)-72,B%(F%,1))>8)-2*(POINT(B%(F%,0)
+72,B%(F%,1))>8):IF P%=0 PROCrepos
1260 IF P%=1:B%(F%,0)=B%(F%,0)-72
1270 IF P%=2:B%(F%,0)=B%(F%,0)+72
1280 R%=0:IF P%=3:R%=RND(2):IF R%=1:B%(
F%,0)=B%(F%,0)-72:ELSE IF R%=2:B%(F%,0)=
B%(F%,0)+72
1290 MOVEB%(F%,0),B%(F%,1):GCOL0,0:PRIN
TCHR$250:IF POINT(B%(F%,0)+32,B%(F%,1)+8
)<9:E%=0
1310 ENDPROC
1320 :
1330 DEFPROCrepos:IF nos%=1:B%(F%,0)=ST
%(1):ELSE B%(F%,0)=ST%(RND(nos%))
1350 B%(F%,1)=844:ENDPROC
1370 :
1380 DEFPROCrestart:FORT=0TO1:SOUND0,-1
5,T,2:NEXT:PROCdelay(1000):L%=L%-1:VDU4:
PRINTTAB(9,2);L%:VDU5:IF L%=0 ENDPROC
```

```
1420 IF W%>8:MOVEX%+64,Y%:PRINTCHR$127+
B$(3+(POINT(X%-8,Y%)>9)+2*(POINT(X%+72,Y
%)>9))+CHR$(POINT(X%,Y%-64))+J$
1430 FOR F%=0 TO K:MOVEB%(F%,0),B%(F%,1
):GCOL0,POINT(B%(F%,0),B%(F%,1)):PRINTCH
R$250:PROCrepos:NEXT:X%=608:Y%=256:W%=10
:MOVEX%,Y%:PRINTM$:ENDPROC
1470 :
1480 DEFPROCbonus:IF B%=0:ENDPROC
1500 VDU4:B%=B%-10:PRINTTAB(xb,yb);"00"
;TAB(xb+3-LEN(STR$(B%)),yb);B%:VDU5:ENDP
ROC
1520 :
1530 DEFPROCdead:VDU4,28,3,19,16,15,12,
26:PROCdbl("GAME OVER",1,3,508):PROCdbl(
"Press <SPACE>",1,3,444):REPEAT:UNTIL GE
T=32
1580 ENDPROC
1590 :
1600 DEFPROCscore:VDU4:PRINTTAB(6-LEN(S
TR$(S%)),2);S%:IF S%>=I%:I%=I%+5000:L%=L
%+1:PRINTTAB(9,2);L%
1630 VDU5:ENDPROC
1650 :
1660 DEFPROChigh:PP=0:VDU4:CLS:FORS=0TO
7:PROCdraw(S*144,800):PROCdraw(S*144,592
):NEXT
1690 FORS=0 TO1:PROCdraw(S*144+72,696):
PROCdraw(S*144+792,696):NEXT
1700 PROCdbl("CUBERT",1,3,784):IF S%>HI
%(8) PROCchange:PP=1
1720 PROCdbl("HIGH SCORES",1,3,500):FOR
T=1TO8:PRINTTAB(2,T+18);T;" ";STRING$(10
,".");TAB(4,T+18);H$(T);TAB(15,T+18);HI%
(T):NEXT:IF PP=1 PROCin
1770 VDU4:PRINTTAB(0,28);SPC(45);:PROCd
bl("<SPACE> to play.",1,3,99):REPEAT:UNT
IL GET=32
1800 ENDPROC
1810 :
1820 DEFPROCchange:FORT=7TO0 STEP-1:IF
S%<HI%(T):P=T+1:T=0
1850 NEXT:IF P=8:GOTO 1880
1870 FORT=7 TO P STEP-1:HI%(T+1)=HI%(T)
:H$(T+1)=H$(T):NEXT
1880 HI%(P)=S%:H$(P)=STRING$(10,"."):EN
DPROC
1900 :
1910 DEFPROCin:PROCdbl("Please enter na
me",1,3,99):PRINTTAB(4,P+18);:*FX15,0
1950 X%=0:Y%=&D:A%=0:!&D00=&200A0A00:?&
D04=127:CALL&FFF1:H$(P)=LEFT$($&A00,10):
ENDPROC
1980 :
1990 DEFPROCtune:FORT=1TO9:SOUND1,-15,1
20+T*4,2
2020 SOUND2,-8,80+T*4,2
2030 NEXT:ENDPROC
2050 :
2060 DEFPROCdelay(TT%):FORt=1TOTT%:NEXT
:ENDPROC
2090 :
2100 DEFPROCcubes
2110 nos%=0:GCOL0,6
2120 FORT=1TO100:PLOT69,RND(1280),RND(9
00):NEXT:READT,noleft,xb,yb,xc,yc:FORU=1
TO T:READA$:xc%=72*EVAL("&"+MID$(A$,1,1
)):yc%=104*EVAL("&"+MID$(A$,2,1)):n%=EVA
L(MID$(A$,3)):IF yc%=728:FORW=1 TO n%:ST
%(nos%+W)=xc%+104+144*(W-1):NEXT:nos%=no
s%+n%
2200 FORV=0 TO n%-1:PROCdraw(xc%+144*V,
yc%)
2220 NEXT:NEXT:ENDPROC
2240 :
2250 DEFPROCdraw(x%,y%):FORT=1TO5:VDU19
,T+10,EVAL(MID$(C$((FR%-1)MOD8),T,1)),0,
0,0:NEXT:GCOL0,12:MOVE68+x%,32+y%:MOVE13
2+x%,y%:PLOT85,68+x%,y%+96:PLOT85,132+x%
,y%+64:GCOL0,11:MOVE140+x%,y%:MOVE140+x%
,y%+64:PLOT85,204+x%,y%+32:PLOT85,x%+204
,y%+96:GCOL0,13
2340 MOVE68+x%,y%+100:MOVE132+x%,y%+68:
PLOT85,132+x%,y%+132:MOVE140+x%,y%+68:MO
VEx%+140,y%+132:PLOT85,204+x%,y%+100:END
PROC
2360 :
2370 DEFPROCnew:L%=5:S%=0:LVL%=0:FR%=0:
D%=14:E%=1:K=0:W%=10:I%=5000:ENDPROC
2410 :
2420 DEFPROCscreen:CLS:FR%=FR%+1:LVL%=(
FR%+7)DIV8:IF (FR%-1)MOD8=0 RESTORE
```

```
2460 K=(LVL%-1)DIV2:IF LVL%>5 K=2
2470 D%=(LVL%-1)MOD2+14:IF LVL%>5 D%=15
2480 PROCdbl("CUBERT",1,3,1020)
2490 VDU4:PRINTTAB(1,1);"SCORE";TAB(1,2
);"00000";TAB(14,1);"LVL:00";TAB(20-LEN(
STR$(LVL%)),1);LVL%;TAB(12,2);"FRAME:00"
;TAB(20-LEN(STR$(FR%)),2);FR%;TAB(9,2);L
%:PROCscore:MOVE640,970:PRINTM$:PROCcube
s:X%=608
2500 Y%=256:FOR F%=0TO2:PROCrepos:NEXT
2540 GCOL0,D%:MOVE572,204:MOVE636,172:P
LOT85,636,236:MOVE644,172:MOVE636,236:PL
OT85,708,204
2550 MOVEX%,Y%:PRINTM$:MOVEB%(0,0),B%(0
,1):GCOL0,0:PRINTCHR$250:GCOL0,3:MOVExb,
yb:VDU230,231,232:MOVExb,yb-24:DRAWxb-32
,yb-24:DRAWxb-32,yb-80:DRAWxb+216,yb-80:
DRAWxb+216,yb-24:DRAWxb+192,yb-24
2580 MOVExc,yc:VDU233,234,235,236,237:M
OVExc+74,yc-36:VDU238,9,239:MOVExc+138,y
c-36:GCOL0,D%:VDU244:xb=(xb+24)DIV64:yb=
(1056-yb)DIV32:B%=1000:J%=0:ENDPROC
2620 :
2630 DEFPROCdbl(A$,C1,C2,H):VDU5:GCOL0,
C1:MOVE640-32*LENA$,H:PRINTA$
2660 GCOL0,C2:MOVE656-32*LENA$,H-8:PRIN
TA$:VDU4
2670 ENDPROC
2680 :
2690 DEFPROCchars:VDU23,224,0,0,0,0,56,
120,240,224:VDU23,225,212,22,19,17,119,1
19,0,0:VDU23,226,0,60,126,86,70,6,14,0:
2700 VDU23,227,60,66,129,169,129,1,1,30
:VDU23,230,0,0,247,148,228,148,247,0:VDU
23,231,0,0,165,181,189,173,165,0:VDU23,2
32,0,0,47,40,47,33,239,0
2770 VDU23,233,0,0,244,132,135,132,244,
0:VDU23,234,0,0,189,165,189,165,165,0:VD
U23,235,0,0,47,168,235,105,47,0:VDU23,23
6,0,0,120,64,112,64,120,0:VDU23,237,0,0,
239,73,73,73,79,0:VDU23,238,8,12,126,127
,126,12,8,0:VDU23,239,16,48,126,254,126,
48,16,0
2840 VDU23,240,240,240,240,240,240,224,
192,128:VDU23,241,15,15,15,15,15,7,3,1:V
DU23,242,0,0,0,0,0,0,24,60,126:VDU23,243,2
55,255,255,255,255,255,0,0
2850 VDU23,244,255,255,255,255,255,255,
255,255:VDU23,250,60,118,251,253,255,255
,126,60:C$=CHR$18+CHR$0:H$=C$+CHR$11+CHR
$240
2920 I$=C$+CHR$12+CHR$241:J$=CHR$242+CH
R$8+CHR$10+CHR$243:B$(0)=H$+CHR$8+I$+CHR
$8+C$:B$(1)=I$+CHR$8+C$:B$(2)=H$+CHR$8+C
$:B$(3)=C$:M$=C$+CHR$3+CHR$226+CHR$8+C$+
CHR$0+CHR$227+CHR$8+C$+CHR$1+CHR$224+CHR
$8+CHR$10+CHR$225:ENDPROC
3000 :
3010 DEFPROCinit:DIM B$(3),B%(2,1),ST%(
7),C$(7),HI%(8),H$(8):ENVELOPE1,0,0,0,0,
0,0,0,126,-8,-7,-7,126,30:ENVELOPE2,2,0,
0,0,0,0,0,127,-10,-5,-2,126,0
3020 RESTORE3190:HI%(0)=1000000:FORT=0T
O7:HI%(T+1)=1500+500*(7-T):H$(T+1)="Beeb
ug":READC$(T):NEXT:VDU23;8202;0;0;0;:END
PROC
3100 :
3110 DATA8,35,132,704,896,704,008,117,2
26,335,444,553,662,771
3120 DATA12,25,132,704,896,704,206,117,
222,A22,331,B31,441,A41,551,951,662,771
3130 DATA12,23,132,800,896,800,602,513,
222,A22,132,B32,242,A42,551,951,662,573
3140 DATA8,23,132,480,896,480,404,513,6
22,731,642,553,464,375
3150 DATA13,39,64,320,960,312,206,315,4
24,335,241,642,C41,153,953,063,A63,172,B
72
3160 DATA12,35,552,576,480,420,404,315,
222,A22,132,B32,042,C42,152,B52,266,375
3170 DATA16,30,64,512,960,512,404,315,2
21,622,C21,332,932,441,A41,355,261,662,C
61,171,771,D71
3180 DATA13,33,64,512,960,512,201,602,C
01,315,424,332,932,441,A41,352,952,266,3
75
3190 DATA46523,53247,65432,23154
3200 DATA73562,12543,73124,72153
3210 :
3220 MODE7:REPORT:PRINT" at line ";ERL:
*FX15,0
3240 END
```

Mr Toad's View Envelope Printing ROM

Mr T keeps you posted.

It's quite a while now since I bought a new printer and wrote a new printer driver for it, which was published in BEEBUG Vol.12 No.4. It has 'paper parking' - almost all printers do, nowadays - but it was only recently that I decided to use this facility to print the envelopes for business letters. View is the word processor I always use, so I confidently set Marker 1 on the first character of the address at the bottom of a letter - it was to Beebug, actually - set Marker 2 on the space after the last character, inserted the envelope and confidently typed PRINT 1 2. 'File not found'. What's that, you stupid *!?**? Oh, of course! PRINT 12. 'File not found'. Oh, sh** ***** *******. (Now, now, Toad, calm down - Ed.) Right! PRINT12. 'File not found'. (Mr T shovels yellow pills into mouth.) Would you Adam and Eve it, blooming View has no facility for printing out just part of a page - or if it does, I can't find it. (If it does, you give us our money back - Ed.) (What? All 50 pence? - T.)

I laboriously deleted everything between the highlight codes at the top and the address itself, typed PRINT and... dammit, I forgot to delete the 'double height' and 'double width' codes which I use to print the address of Toad Hall at the top. Hang on, though, the address looks good in big type and the post-office people will probably find it a help. That bit's OK, but there must be an easier way than this to do the job.

So I sat down and wrote one. It had to be a Sideways ROM, of course, so as to be easily callable from View, so out came my standard small ROM header and a simple prototype was soon working. Install the ROM, print the letter and then put Marker 1 on the first character of the recipient's address. If the address is last thing on the page then that's it, but if you prefer it on the top before the 'Dear Sir,' then set Marker 2 one space after the last character, as you would for any of View's functions. In with the envelope (don't forget to increase the printer head gap, unless it's a label you're printing), go to the command screen and type *EN. The part between Marker 1 and either Marker 2 or the end of the document will be printed. Type *ENB ('B' for 'big') and it will come out in double size and double width.

There's not the time now to test this utility as exhaustively as I would wish, the Hour of Doom is too near upon us for that, but it's been fine with me so far. The following points occur to me.

Do you put the recipient's address on the right? If so, and you add spaces to get it there, the spaces will be printed on the envelope. If you do it by a LM command, the command will be ignored, as will any TAB setting. I would simply move the envelope to the right of the carriage, myself, but in fact I always put the address at the bottom left.

As listed here the ROM prints at the left margin. That's what I want, because I put the envelope a wee bit left of the '0' line, knowing that printing will start on that mark. If you want it to go a bit to the right, I suggest adding the following subroutine at the end somewhere:

```
.addSpaces LDA #&20
JSR osPrinter
RTS
```

Repeat the JSR osPrinter statement once for each extra space you want at the left of the printing, and call it by adding:

```
955    JSR osPrinteR
1035   JSR osprinter
```

If I'd added this to my listing, there would have had to be a prompt 'Enter number of spaces ...' etc, and pretty soon the whole thing would have become a pain to use and too long for the mag. It's better if each user customises the code to suit. On that subject, look at the printer control codes at the end. I've put ESC + "x" + 1 for 'LQ', then ESC + "k" + 2 for 'Courier font'. The second set, ESC + "w" + 1 and ESC + "W" + 1 set 'double height, double width'. You may have to customise these, too, according to your printer's conventions and what you wish to achieve. The left margin spaces coded above might well be achieved more easily by a 'set left margin' sequence: mine would be ESC + "l" + n, where n is the number of spaces - try adding that as line 1765.

Mr T hopes you find this a useful final offering. Had we but world enough and time...

```
 10 REM Program ENVROM
 20 REM Version B1.o
 30 REM Author Mr Toad
 40 REM BEEBUG April 1994
 50 REM Program subject to copyright
 60 :
100 osasci=&FFE3
110 osnewl=&FFE7
120 mk1lo=&53:mk1hi=&54
130 mk2lo=&55:mk2hi=&56
140 zpLo=&8E:zpHi=zpLo+1
150 size=&60:ESC=&1B
160 :
170 FOR N%=4 TO 6 STEP 2
180 Z%=?2+&100*?3-&400*(N%=4)
190 P%=&8000:O%=Z%
200 [ OPT N%
210 :
220 BRK:BRK:BRK
230 JMP whatsUpNow
240 EQUB &82
250 EQUB offset MOD &100
260 EQUB &94
```

```
270 .title
280 EQUS "View Envelope Address ROM"
290 EQUW &0D00
300 EQUS "  EN or ENB"
310 EQUW &070D
320 .offset
330 BRK
340 EQUS"(C) Beebug April 1994"
350 BRK
360 :
370 .help
380 JSR osnewl
390 LDX #title MOD &100
400 LDY #title DIV &100
410 JSR print
420 :
430 .noClaimExit
440 PLY:PLX:PLA
450 RTS
460 :
470 .whatsUpNow
480 PHA:PHX:PHY
490 CMP #9
500 BEQ help
510 CMP #4
520 BNE noClaimExit
530 LDA (&F2),Y
540 AND #&DF
550 CMP #ASC"E"
560 BNE noClaimExit
570 INY
580 LDA (&F2),Y
590 AND #&DF
600 CMP #ASC"N"
610 BNE noClaimExit
620 INY
630 LDA (&F2),Y
640 CMP #&0D
650 BEQ small
660 AND #&DF
670 CMP #ASC"B"
680 BEQ big
690 JMP noClaimExit
700 :
```

```
 710 .wrong
 720 LDX #message MOD &100
 730 LDY #message DIV &100
 740 JSR print
 750 JMP out
 760 :
 770 .big
 780 LDA #&0D
 790 STA size
 800 JMP start
 810 :
 820 .small
 830 STZ size
 840 :
 850 .start
 860 LDA &028C
 870 CMP #&0E \ View
 880 BNE wrong
 890 LDA mk1hi
 900 BEQ wrong
 910 JSR sendStartCodes
 920 LDA mk1lo
 930 STA zpLo
 940 LDA mk1hi
 950 STA zpHi
 960 .loop
 970 LDA (zpLo)
 980 BEQ allDone
 990 JSR osPrinter
1000 CMP #&0D
1010 BNE update
1020 LDA size
1030 JSR osPrinter
1040 .update
1050 INC zpLo
1060 BNE compareMarkers
1070 INC zpHi
1080 .compareMarkers
1090 LDA zpLo
1100 CMP mk2lo
1110 BNE loop
1120 LDA zpHi
1130 CMP mk2hi
1140 BNE loop
```

```
1150 :
1160 .allDone
1170 LDX #finishCodes MOD &100
1180 LDY #finishCodes DIV &100
1190 JSR allCodes
1200 :
1210 .out
1220 LDA #3
1230 JSR osasci
1240 PLY:PLX:PLA
1250 LDA #0
1260 RTS
1270 :
1280 .sendStartCodes
1290 LDA #2
1300 JSR osasci
1310 LDX #startCodes MOD &100
1320 LDY #startCodes DIV &100
1330 .allCodes
1340 STX zpLo
1350 STY zpHi
1360 LDY #&FF
1370 .loop
1380 INY
1390 LDA (zpLo),Y
1400 BEQ endOrNot
1410 CMP #7
1420 BEQ ret
1430 JSR osPrinter
1440 JMP loop
1450 .endOrNot
1460 LDA size
1470 BNE loop
1480 .ret
1490 RTS
1500 :
1510 .osPrinter
1520 PHA
1530 LDA #1
1540 JSR osasci
1550 PLA
1560 JMP osasci
1570 :
```

The Inter-Base Programming Guide

Chris Robbins reviews a useful tome.

One of the largest, and perhaps the most powerful, ROM based application available for the BBC and Master series is Inter-Base. It's also the last, possibly the finest, effort made by Computer Concepts for those platforms. It was certainly a long time coming, but well worth it in the end, despite the limitations imposed by the Beeb and Master's hardware.

As Inter-Base users will already be aware, it's one of the ROM-LINK series of packages that provides (amongst other things) facilities for combining the operations of all the other members in the series i.e. Inter-Word, Inter-Sheet, and Inter-Chart either via simple ROM-LINK commands or the Inter-Base Programming Language (IBPL). Alternatively, it can be used in stand-alone fashion as a straightforward database package.

'Straightforward' is perhaps the wrong term. Certainly the built-in and ready-to-use card index style database, is simplicity itself in use (providing a simple data structuring facility similar to several other Beeb/Master database products). But that's only a part of the story. Where it differs from the others, and the real strength (and major part) of the package is in its powerful and comprehensive Basic-like programming language. This gives it a capability that puts it on a par, in some respects, with commercial database systems.

That, sadly, is also the area in which the accompanying manual is rather deficient. While it fulfils the role of reference manual adequately, it fails to provide the comprehensive guidance that most 'DIY programmers' would need to enjoy the full benefits of IBPL. That's where Martin Pickering's guide comes in.

DEALING WITH THE HARD STUFF

For a start, the 'front end' card index of Inter-Base is ignored; this aspect is covered quite satisfactorily in the original manual. Instead, the guide concentrates on two areas i.e. programming, and what amounts to a completely revamped version of the reference section of the original manual. This latter may not sound very important, but in fact even its adoption of an alphabetical ordering, rather than functional grouping of keywords, makes it far easier to use as a work of reference. To digress slightly for a moment - I've never understood the logic of grouping in terms of function, in a reference manual that is. It's a bit like organising a dictionary so that all the words relating to a subject such as gardening say, are grouped together!

In addition, the reference section of the guide contains, in most cases, far better explanations and useful examples. The keyword SOUND for instance takes up about a page and a half in this guide whereas the original manual has just three lines, merely giving an abbreviated syntax and saying "As in BBC BASIC. ..." etc. In like fashion, RAM SPACE gets seven lines in the manual, but once again, nearly a page and a half in Martin Pickering's guide.

The somewhat shorter section on programming in the guide adopts a gentle approach to learning to use the language, starting with a very short (just 4 lines of code) program to accept a number typed in by the user, and then to display it. Trivial as this may sound, it obeys the 'thin end of the wedge' teaching principle. In other words, introducing a large and complex subject in as simple a manner as possible to avoid putting people off.

BUILDING UP SKILLS

From this straight forward beginning, the reader is gradually introduced to additional keywords and language features via further examples that get longer and more involved as they work through the section. In other words this is the beginners' teach-yourself part of the book. It covers things such as strings, program loops, calculations, sub procedures, dates etc, before going on to the next chapter dealing with simple database programs. This is followed by a chapter on displaying, printing and editing records, the use of sideways RAM to hold IBPL programs, converting programs to token form, and so on. There are also chapters covering communications with other ROM-LINK packages i.e. Inter-Word, Inter-Chart, but not Inter-Sheet.

Throughout, considerable use is made of examples to illustrate and explain, many of which are also available (at a modest extra fee) on disc to accompany the book. It's worth getting the disc just to avoid the tedium of typing in the longer examples. There are, in fact, some very useful programs on the disc e.g. a PC to BBC file transfer program, · program to retrieve records from a corrupted database and, at no extra cost, the complete Inter-Mail package (mail shots, correspondence organiser etc. - reviewed in BEEBUG Vol. 12 No. 8) including ROM images and documentation. The disc also contains the corrections for errata in the guide itself, along with some additional hints and tips.

A small criticism is the lack of a cross reference between the contents of the disc and the guide. Although most of the stuff on disc is referred to in the body of the text, it would be nice to be able to relate text and programs via an index of some kind (a suitable case for IBPL treatment perhaps?).

NEARLY PERFECT

The only real criticism I have of the book is that the introductory chapters that go to make up the 'teach-yourself' part would, I believe, have benefited from a more extended treatment. But, having said that, what that part of the book lacks in size is made up for, to a certain extent at least, by the enhanced reference section.

To sum up, this is a good introduction to IBPL, a "most excellent" accompaniment to the standard reference manual, and well worth the money, especially if you buy the example disc at the same time.

If you've got Inter-Base and haven't turned your hand to programming it, now's the time to try with the help of this guide. And if you haven't got Inter-Base, why not? It's also obtainable from SYNECTICS at a knock-down bargain price.

Monitor Screen Saver

Howard L Smith comes to the rescue.

How often have you forgotten to switch off the TV/monitor showing a screenful of information while you nip out to make a cup of tea, and come back two hours later to find the display still there, slowly burning its way into the phosphor? Well, this utility will solve the problem for you.

You may select any of the two-colour modes, 0, 3, 4, or 6, or the corresponding Shadow mode if you are running a Master series, select foreground and background colours, and choose a timeout delay from 1 to 9 minutes, after which time your screen will be blanked out. The display reappears immediately you press a key. You may switch off the utility by pressing Ctrl-@. You may optionally save the machine code with the currently selected options to disc with a filename of your choice, and re-install them at any time by typing *filename. If you do this you will not need to re-run the Basic program unless you want to change the options.

The Basic source program is quite long - BBC B owners should load it in mode 7. PAGE will probably be at &1900 with a DFS fitted. The program will relocate to &1100, assemble and run the code, then reset PAGE to &1900. The machine code generated is only 365 bytes, and resides at &900, although you may assemble it elsewhere if you wish.

HOW IT WORKS

The screen blanking is achieved by setting the foreground colour to the background. Hence, it appears to 'disappear', but is still there ready to be recovered by resetting the foreground colour.

The BBC micro's event handling routine is used - it senses when a key is pressed, starts a timer, and then uses the start of vertical sync to update the timer.

PROCEDURES AND FUNCTIONS

Procedures have been used extensively, and are generally self-explanatory. However, the Function, FNvdu, is worthy of explanation. It takes data from the DATA statements, and uses them to synthesize VDU strings within the assembly code. This means that a single assembly statement such as OPT FNvdu(6) may be used to read six VDU parameters instead of a long list of LDA and JSR oswrch statements. Any variables within the VDU DATA statements are read as d0, d1, d2 etc, and stored within the string array vdupars$ before assembly takes place.

USING IT

Run the program and answer the prompts. If a text mode is chosen, the background defaults to black, as otherwise, the display would be in horizontal bands of colour, defeating the point of a screen blanker. If the monitor is monochrome, the background defaults to black, the foreground to white, which gives optimum contrast.

LIMITATIONS

The utility will be most useful in applications such as word processing, spreadsheets, and databases, but may be used with any program that does not use screen mode or screen colour changes during execution. Also, any program that uses keyboard or vertical sync events will obviously run into problems if used with this utility. Finally, remember that the code is running at &900, the area used by the RS423 port, cassette port, speech and sound commands (envelopes 5-16). If you need to use these, the code will have to be assembled elsewhere. Here's hoping this doesn't arrive too late for your monitor.

```
 10 REM Program BLANKER
 20 REM Version B1.0
 30 REM Author  Howard Smith
 40 REM BEEBUG  April 1994
 50 REM Program subject to Copyright
 60 :
100 REM BBC B Users - Load in MODE 7
110 IFPAGE>&1100 PROCreloc
120 DIMcolour$(7),vdupars$(5)
130 ON ERROR PROCerror:END
140 *TV0,1
150 MODE3
160 PROCinit
170 PROCscrn_save_pars(Bver%)
180 PROCassmbl
190 IFsave OSCLI("SAVE "+fname$+" 900
A6D 900")
200 CALLtimeout
210 PRINTTAB(8,14)"Screen Saver Instal
led"
220 PRINTTAB(8,16)"Timeout period is "
;delay% DIV(50*mult%);dim$
230 PRINTTAB(8,18)sh$;"MODE ";mode%;"
selected"
240 VDU23,1,1;0;0;0;
250 PRINTTAB(0,6)STRING$(79," "):VDU31
,0,20
260 IFBver%=2 THENPAGE=&1900:M%=PAGE:!
M%=&FF0D:PRINTTAB(0,6)STRING$(79," "):VD
U31,0,20:END
270 END
280 :
1000 REM VDU data statements:-
1010 REM Set screen mode
1020 REM Set backgound colour
1030 REM Set foregound colour
1040 REM Set foregnd to backgnd
1050 REM Switch cursor ON
1060 REM Reset foregnd
1070 REM Switch cursor OFF
1080 :
1090 DATA 22,d0
1100 DATA 19,0,d1@,0@
1110 DATA 19,1,d2@,0@
1120 DATA 19,1,d1@,0@
1130 DATA 23,1,0@,0@,0@,0@
1140 DATA 19,1,d2@,0@
1150 DATA 23,1,1@,0@,0@,0@
1160 :
1170 DEFPROCinit
1180 Bver%=?(&8015)-48
1190 IF(Bver%=1 OR Bver%=3) CLS:PRINTTA
B(21,16)"Utility will not run with BASIC
 1 or 3":END
1200 osword=&FFF1:osbyte=&FFF4
1210 oswrch=&FFEE:eventV=&220
1220 event%=&FC:Hmem%=&07:mc%=&900
1230 ENDPROC
1240 :
1250 DEFPROCscrn_save_pars(ver%)
1260 PRINTTAB(25,0)"Screen Blanking Tim
eout Utility"
1270 REPEAT:*FX15,3
1280 PRINTTAB(15,2)"Enter time-out peri
od required (1 to 9 Minutes): ";
1290 delay%=GET-48
1300 UNTIL(delay%>0 AND delay%<=9)
1310 PRINTSTR$delay%:mult%=60:REM mult%
=60 for mins, =1 for secs
1320 IFdelay%=1 s$="" ELSE s$="s"
1330 IFmult%=60 dim$=" minute"+s$ ELSEd
im$=" second"+s$
1340 delay%=delay%*50*mult%
1350 del_lo%=delay% MOD 256
1360 del_hi%=delay% DIV 256
1370 IFver%>2 PROCshadow ELSE shad=0:sh
$="":opt$="0,3,4,6 "
1380 PRINTTAB(15,6)"Enter Screen mode "
;opt$;" (0,3,4,6): ";
1390 REPEAT:*FX15,3
1400 M$=GET$
1410 UNTILM$="0" ORM$="3" ORM$="4" ORM$
="6"
1420 mode%=VALM$+shad:PRINTSTR$mode%
1430 IFmode%=0 him%=&30
1440 IFmode%=3 him%=&40
1450 IFmode%=4 him%=&58
1460 IFmode%=6 him%=&60
1470 IFmode%>=128 him%=&80
1480 vdupars$(0)=STR$mode%
1490 M%=mode% AND &7F
1500 IFM%=3ORM%=6 tx=TRUE ELSEtx=FALSE
1510   1520 PRINTTAB(15,8)"Do you have a
 Colou
r Monitor? (Y/N): ";
1530 ans$=FNget_ans
```

```
1540 PRINTans$
1550 IFans$="Y"PROCcol(tx)ELSEPROCmono
1560 PROCexit
1570 ENDPROC
1580 :
1590 DEFPROCshadow
1600 PRINTTAB(15,4)"Do you want a Shado
w screen mode? (Y/N): ";
1610 ans$=FNget_ans:PRINTans$
1620 IFans$="Y" shad=128:sh$="SHADOW ":
opt$="128,131,132,134 " ELSEshad=0:sh$="
":opt$="0,3,4,6 "
1630 ENDPROC
1640 :
1650 DEFPROCassmbl
1660 FORpass=0TO2STEP2
1670 P%=mc%:RESTORE1090
1680 [OPTpass
1690 .timeout        \ Initial Timeout.
1700 lda #144        \ Switch interlace
1710 ldx #0          \ off.
1720 ldy #1
1730 jsr osbyte
1740 EQUS FNvdu(2) \ Set mode.
1750 lda himem       \ Reset HIMEM.
1760 sta Hmem%
1770 EQUS FNvdu(4) \ Set BGnd.
1780 EQUS FNvdu(4) \ Set FGnd.
1790 lda #14         \ Enable Start of
1800 ldx #4          \ Vertical Sync
1810 jsr osbyte      \ Event.
1820 lda #14         \ Enable Character
1830 ldx #2          \ Entering Buffer
1840 jsr osbyte      \ Event.
1850 lda #del_lo%    \ Preset delay
1860 sta delay       \ timer and
1870 sta counter     \ counter.
1880 lda #del_hi%
1890 sta delay+1
1900 sta counter+1
1910 lda #1          \ Set timer flag.
1920 sta timer_flag
1930 lda #intrupt AND255 \ Place the
1940 sta eventV      \ intrupt address
1950 lda #intrupt DIV256 \ in EVNTV
1960 sta eventV+1         \ vector.
1970 rts
1980 .intrupt
```

```
1990 sta event%
2000 php:pha:txa:pha:tya:pha
2010 lda event%        \ Check that Event
2020 cmp #4            \ is due to Start
2030 beq elapse        \ of Vertical Sync
2040 cmp #2            \ or Keyboard
2050 beq keypress \ Buffer Event.
2060 rts
2070 .elapse           \ Decrement counter
2080 sec               \ timer, and check
2090 lda counter       \ if a timeout has
2100 sbc #1            \ occurred.
2110 sta counter
2120 bcs skip
2130 dec counter+1
2140 .skip lda counter : bne end
2150 lda counter+1 : bne end
2160                   \ Clear  Screen.
2170 EQUS FNvdu(4) \ Set foreground
2180         \ to background colour.
2190 EQUS FNvdu(6) \ Cursor OFF.
2200 lda #0 : sta timer_flag
2210 .end jmp exit
2220 .keypress         \ Check if still
2230 lda timer_flag \ timing out.
2240 cmp #1            \ If it is, skip
2250 beq reset         \ screen restore.
2260 EQUS FNvdu(4) \ Reset fore-
2270                   \ ground colour.
2280 EQUS FNvdu(6) \ Cursor ON.
2290 .reset lda #145 \ Get character
2300 ldx #0            \ from keyboard
2310 jsr osbyte        \ buffer. If the
2320 tya               \ keypress=CTRL @
2330 beq disable       \ disable events.
2340 lda delay         \ Reset counter
2350 sta counter       \ timer after a
2360 lda delay+1       \ keypress.
2370 sta counter+1
2380 lda #1            \ Reset timer
2390 sta timer_flag \ flag.
2400 jmp exit
2410 .disable
2420 lda #13           \ Disable Start of
2430 ldx #4            \ Vertical Sync
2440 jsr osbyte        \ and Character
2450 lda #13           \ Entering Buffer
2460 ldx #2            \ Events.
```

```
2470   jsr osbyte
2480 .exit
2490   pla:tay:pla:tax:pla:plp
2500   rts
2510 :
2520 .delay EQUD(0)
2530 .counter EQUD(0)
2540 .timer_flag EQUB(0)
2550 .himem EQUB(him%)
2560   ]
2570 NEXT pass
2580 ENDPROC
2590 :
2600 DEFFNvdu(N)
2610 LOCALI,B,byte,lob,dat,dat$
2620 FORI=1TON
2630 READdat$
2640 IFRIGHT$(dat$,1)="@" B=2:b$="@"ELS
EB=1:b$=""
2650 IFLEFT$(dat$,1)="d" dat$=vdupars$(
EVALMID$(dat$,2,1))+b$
2660 dat=EVAL(dat$)
2670 IFASC(dat$)>64 [OPTpass:lda dat:js
r oswrch:]:I=N:GOTO2710
2680 IFdat<0 dat=(ABS(dat)EOR&FFFF)+1
2690 byte=dat MOD256:lob=FNvdu_format
2700 IFB=2 byte=dat DIV256:lob=FNvdu_fo
rmat
2710 NEXTI
2720 =""
2730 :
2740 DEFFNvdu_format
2750 IFbyte<>lob [OPTpass:lda #byte:]
2760 [OPTpass:jsr oswrch:]
2770 =byte
2780 :
2790 DEFPROCmono
2800 PRINTTAB(15,10)"BACKGROUND colour
is BLACK, FOREGROUND is WHITE"
2810 bc%=0:vdupars$(1)=STR$bc%
2820 fc%=7:vdupars$(2)=STR$fc%
2830 monoflag=TRUE
2840 ENDPROC
2850 :
2860 DEFPROCcol(txt)
2870 PRINTTAB(1,13)STRING$(78,"-")
2880 PRINTTAB(2,14)"0) black  1) red  2
) green  3) yellow  4) blue  5) magenta
```

```
6) cyan  7) white"
2890 PRINTTAB(1,15)STRING$(78,"-")
2900 RESTORE2910
2910 DATA Black,Red,Green,Yellow,Blue,M
agenta,Cyan,White
2920 FORI=0TO7:READcolour$(I):NEXTI
2930 IFtxt PRINTTAB(15,10)"BACKGROUND c
olour in Text MODE ";STR$mode%;" is ";EL
SE PRINTTAB(15,10)"Select BACKGROUND col
our (0 - 7): ";
2940 REPEAT:*FX15,3
2950 IFtxt THENbc%=48ELSEbc%=GET
2960 UNTIL(bc%>=48ANDbc%<=55)
2970 bc%=bc%-48:vdupars$(1)=STR$bc%
2980 bc$=colour$(bc%):PRINTbc$
2990 PRINTTAB(15,12)"Select FOREGROUND
colour (0 - 7), except ";bc$;": ";
3000 REPEAT:*FX15,3
3010 fc%=GET
3020 UNTIL(fc%>=48ANDfc%<=55ANDfc%<>bc%
+48)
3030 fc%=fc%-48:vdupars$(2)=STR$fc%
3040 fc$=colour$(fc%):PRINTfc$
3050 ENDPROC
3060 :
3070 DEFPROCexit
3080 PRINTTAB(15,16)"Do you wish to SAV
E current screen options? (Y or N): ";
3090 ans$=FNget_ans:PRINTans$
3100 save=FALSE
3110 IFans$="Y" save=TRUE:PRINTTAB(15,1
6)STRING$(64," "):PRINTTAB(15,16)"Type i
n filename (up to 7 letters): ";:fname$=
FNfname(7):PRINTTAB(15,18)"To install ut
ility from disk, type *";fname$
3120 PRINTTAB(15,20)"Press CTRL @ at an
y time to disable screen blanking."
3130 PRINTTAB(13,21)STRING$(55,"*")
3140 PRINTTAB(13,22)"*  Press any key t
o install the screen-save utility.  *"
3150 PRINTTAB(13,23)STRING$(55,"*")
3160 VDU23,1,0;0;0;0;
3170 X=GET
3180 PRINTTAB(14,22)STRING$(18," ");"Pl
ease wait...";STRING$(19," ")
3190 ENDPROC
3200 :
3210 DEFFNget_ans
```

```
3220 REPEAT:*FX15,3
3230 A$=CHR$(GET AND &DF)
3240 UNTILA$="Y" OR A$="N"
3250 =A$
3260 :
3270 DEFFNfname(N%)
3280 LOCALI%,asc,ch$
3290 text$="":I%=0
3300 REPEAT
3310 *FX15,1
3320 VDU31,POS,VPOS
3330 ch$=GET$:asc=ASCch$
3340 IF(asc<>127ANDI%<N%) inc%=1
3350 IF(asc=127ANDI%>0) inc%=-1
3360 IFasc<>127ANDasc<>13ANDI%=N% PROCi
nv_char
3370 IFasc=127ANDI%=0 PROCinv_char
3380 IFasc<47ANDasc<>13 PROCinv_char
3390 IFasc>122ANDasc<>127 PROCinv_char
3400 IFasc=13ANDI%=0 PROCinv_char
3410 I%=I%+inc%:text$=LEFT$(text$,I%):P
RINTch$;
```

```
3420 IFasc<>127ANDasc<>13 text$=text$+c
h$
3430 UNTILasc=13ANDI%>1
3440 =text$
3450 :
3460 DEFPROCinv_char
3470 inc%=0:ch$="":VDU7
3480 ENDPROC
3490 :
3500 DEFPROCerror
3510 CLS:REPORT
3520 PRINT" at line ";ERL
3530 ENDPROC
3540 :
3550 DEFPROCreloc
3560 CLS:PRINTTAB(0,5)"Relocating, plea
se wait..."
3570 FORI%=0TOTOP-PAGE STEP4
3580 I%!&1100=I%!PAGE:NEXT
3590 ?&13=?&13-(PAGE-&1100)DIV256
3600 PAGE=&1100:RUN
3610 ENDPROC                          B
```

Mr Toad's View Envelope Printing ROM (Continued from page 15)

```
1580 .print
1590 STX zpLo
1600 STY zpHi
1610 LDY #&FF
1620 .loop
1630 INY
1640 LDA (zpLo),Y
1650 JSR osasci
1660 CMP #7
1670 BNE loop
1680 RTS
1690 :
1700 .message
1710 EQUW &0D03
1720 EQUS "Get it right, fumblehead!"
1730 EQUW &070D
1740 .startCodes
1750 EQUB ESC:EQUS "x":EQUB 1
1760 EQUB ESC:EQUS "k":EQUB 2
1770 BRK
1780 EQUB ESC:EQUS "w":EQUB 1
1790 EQUB ESC:EQUS "W":EQUB 1
```

```
1800 EQUB 7
1810 .finishCodes
1820 EQUW &0D0D
1830 EQUB ESC:EQUS "@":EQUB 7
1840 ] NEXT
1850 :
1860 FOR N%=7 TO 4 STEP -1
1870 IF N%?&2A1 NEXT:PRINT'"Sorry, no f
ree SRAM slot.":END
1880 OSCLI "SRWRITE "+STR$~Z%+" "+STR$~
(1+O%)+" 8000 "+STR$ N%
1890 N%?&2A1=&82
1900 PRINT'"OK in slot ";N%'
1910 PRINT"SAVE OBJECT CODE? (Y/N)"'
1920 g$=CHR$(ASC GET$ AND &DF)
1930 IF g$<>"Y" PRINT"See if I care!"':
GOTO 1960
1940 INPUT "Filename? "n$
1950 OSCLI "SRSAVE "+n$+" 8000 "+STR$~(
1+P%)+" "+STR$ N%
1960 N%=4:NEXT:END
1970 Bufo scripsit MCMXCIV             B
```

Machine Code Corner

In which Mr Toad croaks his last.

Avete atque valete: hail and farewell, toad fans and reptile readers all - or should I say, 'both'? As the old song says: 'venit mors velociter, rapit nos atrociter, nemini parcetur'. And that's much too gloomy to translate.

It's been fun all these years, stoking up the old Beeb every month and setting the steam pressure to 'MAX' before blowing a merry toot on the whistle and bashing out the old column, little webbed fingers going like the clappers. Seriously, though (for once), I thought I'd finally relent and tell you how I got involved with BEEBUG and how the whole 'Toad' thing came about.

MR TOAD, THE WILDERNESS YEARS

I had had a Beeb for several years before I dared to submit anything to any of the magazines. I knew about BEEBUG, which was then in Dolphin Place in the centre of my home town of St Albans. Not long before I tried sending them some stuff, they moved to the old 'British Medical Journal' building at 117 Hatfield Road, about 400 yards from my home. I think seeing the name BEEBUG every day on my way to work gave me the impetus to have a go.

Anyhow, early in 1990 I submitted an article on 'Initialising ROM Images', which was about as clear as mud, as funny as Jeremy Beadle and as original as Eastenders. I waited ages for an answer - in those days it took literally months - and I'd almost forgotten about the whole thing when, to my amazement, they accepted it. Reading it again now I cringe, but you have to give me marks for chutzpah.

In the meantime I'd been working on a text encryption ROM called 'Krypton', which wasn't all that bad, looking back on it now. It was the first really big thing I'd done in 6502. The programming was somewhat less than structured, but it did its job and, above all, the main screen looked attractive. So, although encryption programs were not exactly an original idea, I tried it on BEEBUG and 'BBC Acorn User'. Many months later I got an acceptance from Mike Williams at BEEBUG and, about a week after that, a letter from Paul James at 'Acorn User' asking me why I hadn't replied to his acceptance which he had sent me in the summer! I hadn't received it, of course.

At the time I wept buckets, because I thought the other mag far more prestigious - I've since revised that opinion totally, and not because of 'sour grapes'. I've since had stuff accepted by 'Acorn User' and 'The Micro User'. BEEBUG has been something quite unique and I've been amazed at my luck in having been privileged to contribute. That's from the heart; I need no Brownie points now and I'm not going to get an Arc, excellent as they are, so the connection ends here. I'll probably still stroll down to the showroom now and again and annoy them with silly questions - old habits die hard.

MEANWHILE......

But I digress. (Who cares? Ed.) I was getting encouraged by now, and wrote some more ROM software for BEEBUG. BEEBUG were also getting to know me and were giving me quicker decisions: not all acceptances, either. When I wrote my 'Fontz' ROM (again, not a completely original idea, but the programming was not quite so dire by this stage,) I decided on an impulse to send it in as 'Mr Toad's Amazing Magic Fontz ROM'. I was quite certain that Mike would cut that bit out -

I couldn't believe my mince pies when the mag hit the doormat and it had been left in.

Mrs T and I had been using the 'toad' thing for fun as long as we'd been married. We once had the idea of using different false names for the house when paying credit-card bills, etc, so that when junk mail arrived with one of the names on the envelope, we'd know who had sold our address on one of those commercial lists. 'Toad Hall' was one of the names we used so we always knew, when some cheap schmutter merchant got our address, where it came from. It was definitely our favourite, and we had even put a 'TOAD HALL' sign over the front door.

From 'Fontz' on, most of my stuff for BEEBUG went in under that name. Then I had the idea for the column, which has been unique - neither BEEBUG nor anyone else has ever printed such sick and degraded rubbish before or since. I have this theory that Mike Williams hired Marshal Anderson, his freelance assistant, just to avoid having to look at it. I mean, 6502 machine code simply isn't funny, is it? It's like making a silk purse out of a blooming great lump of granite, innit? Marshal, you're a good guy, it's been nice working with you - but do watch out for the word 'honorarium'!

It's been enormous fun, but not easy. My full-time job is pretty demanding - I'm Head of Modern Languages and a Housemaster at the Bury Lawn School in Milton Keynes - and the travelling alone takes up an hour each way. What's more, in 1991 I was a co-founder of the charity PMS Help, which takes up time every single day. I have the honour to be the son-in-law of Dr Katharina Dalton, the PMS expert (she 'invented' the disease) and she and I have co-authored a book on the subject which comes out in September. That took a bit of time, too - but it was all written in View, lang may

its lum reek! My real hobby, historical linguistics, hasn't really had a look-in for years, though in one of the early columns I did parody a famous quote from King Alfred, again not really expecting ever to see it in print!

Talking of PMS Help, another of those weird coincidences happened the other year: a women's magazine printed PMS Help's box number wrongly: instead of PO Box 160, St Albans they put PO Box 50 (why, I'll never know). Happily, the post office people diverted almost all our mail, but we thought we'd better write an apology to the unknown owners of box 50 - it turned out to be BEEBUG.

I mention the time factor as an explanation to the many readers who have so kindly written in and who have had to wait too long for a reply, possibly a scrappy reply at that. It's not that I don't value your letters - I do, very much - but I never get a minute during the school term. In the holidays I tend to be more civilised.

LUVIES, I'D LIKE TO THANK...
Thanks to the following for their input: Fergal McPhurgle, Ossian de Lytelpewbes of Bognor and Ms Scarlatina Nostril. There have been others. Thanks also to all you more regular readers, especially Arthur Adams, Tim Parsons and Silas S Brown (who got good GCSE grades despite being led astray by thoughts of 6502 programming). Silas's last query is the last I'll be answering in these august pages. What it boils down to is this: can you cram a full set of Beeb electronics inside one of the computers at his school, so that the teacher thinks you're working when you aint! I think that was it, more or less; the letter has got lost, as usual. Anway, Silas - no. Nice idea, though, yer a man after me own heart. And if Silas or anyone else got a set of PMS Help literature from me (or if any lady out there suffering from PMS was

puzzled to receive advice on 6502 programming), now you know why. All the letters are piled on the same very small table. I once knew what colour the tabletop is.

If despite this deadly danger you still want to write to Mr T, I'd be delighted to hear from you - write to David Holton at 41 Harlesden Road, St Albans AL1 4LE. Because PMS Help's stuff is now all on PCs, alas, I'm moving on to a PC, too. They're no fun, but one has to interface with other computer users in the real world. Despite this I shall keep the Beeb on until it conks out irreparably or until you can't get the coal.

Now I'm going to be boring and predictable and give you my final bits of advice about 6502 assembler. It's at various levels; in the first MCC I announced my intention of aiming at beginners and old hands in turn, and it's one thing I have stuck to. Here are the ten commandments.

1. Have a go. You have nothing to lose but your brains. I think Lenin said that.

2. Decide on one aspect of the game and specialise in it. Mine was service ROMs, mainly to help Basic programmers. You might be good at graphics, for instance.

3. Do try to use Hex until it becomes second nature and decimal seems clumsy in assembler. I stuck to decimal when I learned Z80, and I never really became good at it. I said that.

4. Try to use all the instruction set, even to the point of playing with test routines for the ones you don't really know. Don't be a Reduced Instruction Set Programmer. I'm not good at this, but I recognise the fault.

5. Rules are there to be broken, never be afraid to have a crack at the unorthodox. I think Pope Flatulentius XXIV said that. BUT, if you ever get to write for a wider audience, do try not to foul up other software - yours will only get discarded. Workspace, in particular, needs selecting with care. I've made some booboos myself in this very mag. I expect all the readers said that.

6. Know the memory-map backwards. Forwards is quite useful, too. Rabbi Lionel Pink said that.

7. Use the same labels for the same functions in many of your programs. No need to make a fetish of this, but it helps. I'm sure Spike Milligan said that.

8. The bug is seldom in the bit of code you're looking at, and seldom the kind of bug you're looking for. A particular piggywig is the hash (#) before numbers. Even in the latest thing I ever wrote, this very week, I spent half an hour wondering why this wouldn't work:

    LDX message MOD &100:LDY message DIV &100:JSR print

    I do it every time. Fergie said that.

9. Always use good-quality coal and have the sweep twice a year.

10. Light a candle to St Postula at Whorplemastide. Look where it's got me.

Here's a really bad joke on which to end: Qu. What's the difference between a duck? Ans. One of its legs is the same length. And that's it, folks. Ite, missa est. B

Public Domain

Alan Blundell looks forward to the latest releases, and back at the development of PD software for the Beeb over the past four years.

I have been looking back over the PD columns in BEEBUG over the last couple of years, and it set me thinking about the development of PD software for the BBC micro. Four years ago, there were PD programs about, but there weren't many, and unless you had a modem, you would have had difficulty in getting hold of them. A strange situation really, for a computer with such a large number of committed users, especially as it's a 10 year old machine. Other computers, such as PCs, Macintoshes, Amigas and STs (if you will, for the moment, allow me to mention them in the same paragraph as the Beeb ...) all had thriving PD software growth and libraries to cater for its distribution to those who weren't 'online'. Perhaps it only needed some PD libraries to make the situation different.

Those were my thoughts when I started the BBC PD library. Other libraries started at around the same time or slightly later, and the amount of PD software available seemed to grow and to keep on growing. The fact that there was a way of distributing their programs seemed to bring home programmers out of the woodwork. Whilst some of the software which appeared has always been of variable quality or of limited interest, right from the start there have been some excellent quality programs and other items (such as clip art discs for PageMaker, Wapping Editor and other DTP programs). Since the BBC magazines (and BEEBUG was in at the early stages) began to report on PD software, many thousands of people must have taken the opportunity to try out some of what's available. Judging from letters I have received, many of them are very happy that they have done so.

There are now many megabytes of PD software available; I once totalled up the file sizes of the available software and got to over 30 Megabytes, but that was quite a while ago now, and it didn't include all of the PC software which has been tested for use on the Master 512 co-processor or the PD software which has become available for the 512. I did think about writing a potted 'best PD software' list, gathered from the programs I have reviewed in writing this column for the last couple of years. But, given that everyone has different tastes and interests, it could only be a personal selection and I'm not sure that I could pick out a list that would fit on two pages of BEEBUG anyway! I must say, though, that the professional quality of some of the software is amazingly good and I have gained a lot of satisfaction from seeing the release of some ex-commercial software to a new and wider audience over the years.

Instead, I'll finish this last column, as I've been able to do all along, with some newly-released PD programs. I have got used to the idea now that the BBC micro will go on for years yet, with people not only actively using it but creating new and high quality programs for it as well. If anything, the quality of the programs has gone up recently, showing that all the 'good' programmers haven't gone on to 'better' computers. These two examples will serve nicely to illustrate the point.

PUPIL ASSESSMENT

I have recently received a disc (BBC PD disc 174) containing an extensive system of pupil assessment recording for use with the National Curriculum. The system is a public domain demonstration only version of a comprehensive computer-based recording system by Birkenhill Computing Services Limited. As it stands, it is limited to the Scottish

system only, but Birkenhill will willingly develop the program for use elsewhere.

To run the system you will need a BBC Master microcomputer with ADFS and a disc drive (for class reorganisation - a double drive) capable of reading 80 track discs. The system assumes that sideways RAM is fitted in socket Nos 4 & 6.

The Scottish curriculum is split into 5 'curricular areas'; Mathematics, Language, Expressive Arts, Religious & Moral Education and Environmental Studies. These areas have been divided further into 15 subjects. Each subject is divided into one or more outcomes and these are then further sub-divided into one of several strands. Within each strand a pupil should be assessed as having attained a level, normally, but not exclusively in the range 'A' to 'E'. This gives something of the order of 183 strands, representing, for a class of, say, 30 pupils, some 5,500 items of assessment information. The prime concern of the system is the organisation of this information. The system provides facilities to maintain a 'Class Register' together with some items of personal information, such as address and home telephone number. The entry, alteration and retrieval of the assessment data is accomplished through the use of a fast and simple series of logical menus.

Also included in the system is a description of all the attainment levels, by strand, for every outcome in every subject. While this is apparently no more, and on occasion less, than the texts supplied in the guidelines, a relatively convenenient, on-line access to this text could be more useful to those teachers who had not yet memorised it, than recourse to the drawer, file or cupboard, where the guidelines were normally to be found.

LIGHT RELIEF
I have also recently heard from Miroslaw Bobrowski, whose name has appeared at the head of more than one article in BEEBUG. I have previously mentioned a collection of his software which he has released as PD, and he has now added to this with two new games.

The first of these is 'Puzzle Master', an enhanced version of an earlier work of his, which is essentially a picture puzzle game. It comes with a selection of graphics screens which are used as the basis for the puzzles, and there is the facility to add your own. The second is a logic puzzle game, 'Octagram', which is based on numbers arranged in an octagram (of course...) and the objective of the game is to move the numbers around to get them into numerical order going clockwise around the octagram. Sounds easy, but of course it isn't (or at least, it wasn't for me).

THE END
I can't let this series of articles come to an end without a few apologies and thanks. The apologies are to the unfortunate few who I've managed to keep waiting for unreasonably long times for a reply to their letters (there have at times been far too many to cope with, most of the time there have just been enough to keep me awake only half the night) and to Mike Williams, whose fingernails must be well bitten at my creative approach to article deadlines on more than one occasion. The thanks rightly go to a few hundred people - too many to list or thank individually. They include writers of PD software, for writing it in the first place and for releasing it as PD, and readers of this column who have written to me with encouragement and appreciation over the years. Thank you, one and all - it really has made a difference!

Now, I think I'm going to start writing a new program for the children. I've been meaning to do it for ages ...... 🅱

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.
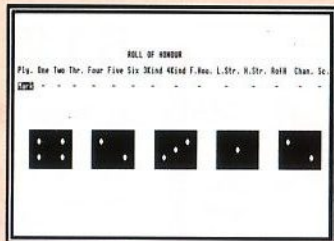
Applications II Disc

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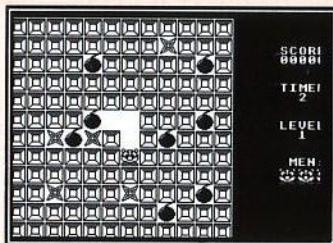
PITFALL PETE - Collect all the diamonds on the screen, but try not to trap yourself when you dislodge the many boulders on your way.

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.

FINAL ✳ OFFERS ✳

Tel. (0727) 840303   Fax. (0727) 860263

512 Forum

by Robin Burton

Here we are, the last ever 512 Forum! I've not been looking forward to it for obvious reasons, but also for the practical reason of finding a memorable subject (or at least interesting) which won't need a continuation article! I hope you think I've done our favourite machine justice.

THE ULTIMATE 512!

Thanks to David MacGraw for supplying the information on his unique 512 upgrade. Space is too limited to do full justice to all his work, so I'll provide photocopies of his summary for a pound to anyone who wants more information.

David and I have communicated for quite a time on general 512 matters, but last year he asked me for my thoughts about the 16Mhz speed upgrade mentioned in the 512 Technical Guide. Basically, did I think the 512 would go even faster and if so, what advice or suggestions could I offer.

There's nothing like being put on the spot, is there? Well, I did write the book and I also produced the 512 memory expansion that quite a few of you including David use, so I could hardly say I had no idea!

Even so, this was totally uncharted territory. I'd never pushed my own 512 beyond 12Mhz and certainly never tried going beyond 16Mhz, because I knew very well that it would be a time consuming job even if it worked, and of course there was no guarantee it would.

After a bit of thought I told David that I couldn't see any obvious reasons why it wouldn't work in theory, provided that the job was tackled methodically and that he was prepared to replace (probably most of) the chips. I also said I

thought it likely that if he reached it the ultimate performance barrier would probably be the tube, if not the ULA itself then because many XIOS (tube I-O) functions are time critical.

A few more letters passed and finally David asked me if I could supply a spare 512. He'd decided to have a go!

EVOLUTION

The first upgrade was a processor change, to a 16MHz CMOS version. This also involved a crystal (the hardware timer) change and a couple of other very minor mods, but no other changes. (More on this later, but if you're tempted this is the only upgrade I suggest you try and you might still need to upgrade the RAM.)

The result of this fairly simple change is an increase in general processing speed of around 60%, as you'd expect from the speeds of the two crystals, 20 and 32MHz Of course there's little benefit for tube/BBC micro dependent operations such as disc access or screen output, but that's a function of the machine's overall design so you can't do anything about it.

Successful so far, David's next step was a 20Mhz processor with a 40Mhz crystal, but he found that this time a RAM change to 80ns (nano-seconds) or better couldn't be avoided. It did then produce the expected performance improvement, but at the cost of re-wiring the 512's RAM for different chips.

In addition this change also threw up an 'ESC opcode error' from a number of programs, although others still worked normally. It didn't take long to spot that the troublesome programs were all EXE files, although not all EXE files were affected. When David mentioned this to me I dimly remembered the same problem in another machine a few years back.

Unfortunately the processor had failed, so the user had replaced it with a CMOS version, although in this case he certainly wasn't running it beyond 16MHz He asked me if I had any ideas why a number of EXE programs would no longer run, all failing with an ESC opcode error.

At the time I hadn't, so I was only able to say that the new processor appeared to be responsible since it was unlikely to be the speed, my own standard chip being quite happy at 12Mhz (also, why some programs and not others?). In any case, although it was true some 512s wouldn't run reliably even at 12Mhz this wasn't the symptom they displayed, the whole machine crashed randomly.

The best guess was a minor incompatibility in the CMOS processor, in probability a bug which was never fixed because the 80186 was over-taken by the '286 and so was never used in PCs. How David reached 20Mhz before he encountered the problem I don't know, but possibly not all CMOS '186 chips are the same, I can certainly confirm from fitting memory expansions that not all 512s are the same.

David persisted and found that the cause was a maths co-processor instruction. Many EXE files compiled and linked in 'C' include floating point instructions in the program in case an '87 maths chip is available. So they can run in machines without the maths co-pro. these programs check to see if the maths chip is present at run time. Clearly this check fails in the CMOS '186 though not in the standard processor. Equally clearly it can be viewed as a bug, because maths co-processors weren't introduced until the 80286 appeared.

Having established that the fault wasn't speed related David pressed on, this time to a 25MHz processor, which called for another RAM swap, down to 70ns, giving roughly 250% of the speed of the standard 512.

I've already mentioned that I actually run my (nominal 10MHz) 80186 at 12MHz without problems and I know quite a few other 512 users who do so too. Indeed a good many PCs employ a processor which is driven beyond its nominal rated speed to produce a faster machine without the costs of a faster chip.

Applying the same logic David pushed his 25MHz processor to 27.75MHz but to do so demanded yet another RAM change, now to 60ns. However, David feels that the cost and effort was worthwhile because he uses his 512 for a number of processor intensive design applications. For these his 512 now runs at 275% times its original speed (and around 75% of the speed of the '386 he uses at work for the same job).

David has tried to push the 25MHz chip to 30MHz but without success, so he says he's probably going to stop at that. From his notes it looks like the 25MHz processor might now be the limiting factor, but while I don't know the speed of the fastest C186 chip I do know it can't be much more than 25MHz, so perhaps there's no choice anyway.

What does surprise me is that the tube hasn't been a problem so far, unless that's what's actually now preventing a further speed increase. Bear in mind that the main difficulty of this sort of experimental development is that you can't identify a failure until, by a trial and error process of elimination, you've cured it. Each RAM and/or processor upgrade is quite expensive, so it obviously can't go on for ever.

I think David's wise to call it a day, he's got further than I expected and without doubt has the fastest 512 ever built. To give you a better idea I've summarised some of his performance tests in the table so you can see for yourself.

Operation - Pc:-	Integer count	Real count	Table lookup	String manip.	Empty loop	Av.
4Mhz.8088	1.0	1.0	1.0	1.0	1.0	1.0
8Mhz.AT	4.0	3.6	4.0	4.1	4.1	4.0
25Mhz. '386	12.0	14.5	14.0	14.5	14.5	13.9
80C186 512:-						
10Mhz.	3.0	3.2	3.5	3.6	3.6	3.4
12MHz.	3.4	4.1	4.0	4.8	4.1	4.1
16Mhz.	4.8	5.8	5.6	5.8	5.8	5.6
20Mhz.	6.0	7.3	7.0	7.3	7.3	6.9
27.75Mhz.	8.0	9.7	14.0	9.7	9.7	9.9

Tests using SSE.COM v.2 (Dabs shareware vol.2 disc 5)

You can clearly see that for some operations the improvement is better than others, but as I said, for I/O the results remain almost the same regardless of processor speed. For example, comparing the 10MHz machine to the 27.75 version using *512BBCBASIC 'NEWBENCH'*, screen text output improves only 0.3 from 17.1 to 16.8, while disc access gains a little more (because there's more data to move!) from 5.0 to 4.3.

BACK TO REALITY
Before you get carried away with the idea of upgrading your own 512 a few words of warning are justified. I'll supply the information I have to anyone who wants it and David may be able to offer additional notes and guidance for would-be upgraders. However, if you do feel tempted be aware that you'll absolutely definitely have to do the job yourself.

Don't be under any illusions, the full upgrade is a considerable amount of work. It's fiddly, it's time consuming and you must know your way round the 512 circuit and the pins of both 256x1 and 256x4 DRAMs, since you will need to rewire the 512 to take four of the faster CMOS chips in place of the original sixteen 150ns NMOS devices.

I've only sketched the main steps in the development of this upgrade. Don't imagine it was straightforward, quite the opposite. Apart from the first change to

16MHz (which is a potential proposition for most people) David had numerous false starts and various problems to resolve before each speed increase finally worked properly (and he runs an electronics company!).

Note too that the EXE file problem remains and the only solution is to manually patch each affected program. Add to this the fact that some 512s are 'better' than others, even though the upgrade works in David's system on both the internal and external tube, there's no guarantee it would in yours.

Don't even contemplate this upgrade if you're not absolutely confident of your ability to modify the hardware and some programs. I repeat, the work CANNOT be done for you. (Even if it could, to put it in perspective, it would probably be almost as cheap to buy a PC.)

ODDS AND ENDS
Sorry to those who've been waiting for things from me, my Beeb has problems (the final two Forums were written on a PC!!) which I hope will soon be fixed. When they are, I shall, thanks to Beebug, be able to release a (DOS) disc containing all the Forums (about 55 issues) still on my hard disc. See Editor's jottings in issue 8 for possible sources, or I can supply it on the usual terms.

THE FINAL FAREWELL
The moment has arrived. It only remains to thank you all for your encouragement, kind words and enthusiasm over the years. I hope you continue to enjoy your 512 for many years to come and that I might hear from one or two of you from time to time.

BBC MICRO/MASTER 128 & COMPACT

A Simple Disc Menu

John Wescott's utility makes program selection easy.

This menu allows you to select and load a program with one keypress from a one page menu of all 29 programs on a disc. The DFS allows a maximum of 31 files on a disc and after allowing for *!BOOT* and *!MENU* you have room for another 29. In addition you have plenty of space for a description of these programs.

This very simple program selects pairs of data to construct the menu. The first data item is the disc program filename, the second data item is the description of that program. The filename is the name of one of your programs already saved to that disc. The description can be up to 67 characters long.

!menu in action

As printed the data statements hold two dummy programs and 27 other dummy data pairs, all of which will eventually be replaced by the user as they add further programs to the disc. The dummy data pairs will give the user a quick guide to how many more programs can be added to that menu.

The menu is booted in the usual way by holding down Shift and pressing and releasing Break then releasing Shift. To select any program from the menu simply press the key indicated on the left hand side of the screen (i.e 'A' to 'Z', '[', '\' or ']'). If the program selected is not on the disc then a 'Not Found' message is shown and after pressing spacebar the menu program is run again.

A typical menu in practice

To use this program type it in and save it to your disc as *!MENU*. Now save it again under another name, e.g. *NewMenu*, this version will be the program to save on all your other discs as *!MENU* because the !MENU version on this disc will gradually fill up with your programs. Add a !BOOT file (which includes CHAIN "!MENU") to your disc and type in *OPT 4,3. Now save one or more of your own programs to that disc and add your own DATA statements starting at line 2000, in place of the dummy DATA statements.

Your menu program will now be ready to use. To boot the menu, hold down Shift whilst pressing and releasing Break.

This will chain the menu. Choose a program from the menu by selecting one of the characters at the left-hand side of the screen. To add another program to the menu at a later stage just load the program as normal and alter the next dummy data pair. Save the amended program again as *!MENU*.

```
 10 REM Program !Menu
 20 REM Version B1.0
 30 REM Author  John Wescott
 40 REM BEEBUG  April 1994
 50 REM Program subject to copyright
 60 :
100 ON ERROR GOTO 240
110 MODE 0
120 VDU 19,1,2,0,0,0:VDU 19,2,4,0,0,0
130 DIM progName$(29)
140 PRINTTAB(34,2)"PROGRAM MENU"'
150 FOR menu = 1 TO 29
160 READ progName$(menu),describe$
170 PRINT CHR$(menu+64);"..";progName$
(menu);TAB(12);describe$
180 NEXT
190 PRINT"SELECT BY TYPING CHARACTER O
N LEFT (ie A-Z, or [, or \, or ])";
200 REPEAT G=GET:G=G AND &DF:UNTIL G>6
4 AND G<94
210 MODE7
220 CHAIN progName$(G-64)
230 REM * If program not found then re
run menu program by pressing SPACEBAR *
240 IF ERR=214 THEN PRINT;"'";progName
$(G-64);"' NOT FOUND - PRESS SPACEBAR":R
EPEAT UNTIL GET=32:RUN ELSE REPORT
```

```
250 REM * Your data follows. The first
data item on each line is the   *
260 REM * disc program name, the secon
d data item is a description of *
270 REM * that program.  There is room
for 29 programs.                *
280 REM ================================
==================================
2000 DATA MyProg,Dummy entry - replace
this with your program name and descript
ion.
2010 DATA ANOTHER,Dummy entry-the descr
iption should be no longer than 67 chara
cters.
2020 DATA P27,P27=Room for 27 more prog
rams.
2030 DATA P26,P26
2040 DATA P25,P25
2050 DATA P24,P24
2060 DATA P23,P23
2070 DATA P22,P22
2080 DATA P21,P21
2090 DATA P20,P20
2100 DATA P19,P19
2110 DATA P18,P18
2120 DATA P17,P17
2130 DATA P16,P16
2140 DATA P15,P15
2150 DATA P14,P14
2160 DATA P13,P13
2170 DATA P12,P12
2180 DATA P11,P11
2190 DATA P10,P10
2200 DATA P9,P9
2210 DATA P8,P8
2220 DATA P7,P7
2230 DATA P6,P6
2240 DATA P5,P5
2250 DATA P4,P4
2260 DATA P3,P3
2270 DATA P2,P2
2280 DATA P1,P1
```

B

MeReloc

Miroslaw Bobrowski shifts some bytes.

The program presented here will give your Beeb two new star commands. These will enable you to move data and machine code programs around in memory very easily. The Basic program *MeRelocS* is a source program generating the ROM image called *MeReloc*. To generate the ROM, type in the MeRelocS program and save it to disc. Once you have checked your code for any typing mistakes run the program. The ROM image will be saved to disc and you can then load it into a free sideways RAM slot using the *SRLOAD command. Pressing Ctrl-Break will initialize the ROM and produce two new star commands:

        *MSHIFT <start> <end> <dest>

and

        *RELOC <start> <end> <dest>

These two commands do basically the same job; they move a block of memory from one part of the RAM to another. The syntax is like most star commands, data is separated by spaces rather than commas. Both commands use the same data: <start> and <end> are the first and the last addresses of a memory block being moved, and <dest> is the starting address of the destination area. Both commands transfer a block of memory from an old location to a new one, without or with adjustment of internal addresses, respectively. In fact, the *MSHIFT command does the same job as the ADT's command *MOVE.

There are many uses for this type of operation within Basic programs. It could, for example, be used to put an image on screen very quickly, having

assembled it elsewhere, or you could dump data directly into various buffers. The important thing to remember is that, if you are moving data that you want preserved, you should use *MSHIFT; if you are moving machine code then you will need *RELOC.

As it stands, the MeRelocS program will only work on a Master 128 or Compact. However, making it run on a BBC B is no problem. Just replace the following lines.

```
100MODE7
150:
4120EQUD &81810F81:EQUD &81178314
4140EQUD &AAA999B9:EQUD &55655555
4150EQUD &66AA5565:EQUD &EEEDEEEE
4160EQUD &77F77775:EQUD &FFF7FFFF
```

You will need some sort of sideways RAM available to make use of the image.

```
  10 REM Program MemRelocS
  20 REM Version B1.0
  30 REM Author  Miroslaw Bobrowski
  40 REM BEEBUG  April 1994
  50 REM Program subject to copyright
  60 :
 100 MODE 131
 110 ON ERROR PRINT:REPORT:PRINT" at li
ne ";ERL:END
 120 PROCassemble
 130 PRINT"Code: &8000-&";~P%;" stored
at &5000-&";~O%
 140 PRINT"To save code enter:"'" *SAVE
 MeReloc 5000 ";STR$~O%;" FFFF8000 FFFF8
000"
 150 PRINT"To transfer code into sidewa
ys RAM enter:"'" *SRWRITE 5000 ";STR$~O%
;" 8000 4"
 160 END
 170 :
1000 DEFPROCassemble
```

```
1010 osbyte=&FFF4:osnewl=&FFE7
1020 oswrch=&FFEE:osasci=&FFE3
1030 num=&2A:tempa=&4F:tempx=&50:tempy=
&51
1040 addr=&52:vect=&54:lim=&56
1050 srce=&70:dest=&72:len=&74
1060 oldloc=&76:oldend=&78
1070 newloc=&7A:newend=&7C
1080 diff=&76:ptr=&78
1090 :
1100 FOR pass=4 TO 7 STEP 3
1110 P%=&8000:O%=&5000
1120 [OPTpass
1130 .header:EQUB 0:EQUW 0:JMP service
1140 EQUB &82:EQUB (copyright-header):E
QUB 7
1150 .title:EQUS "Move & Relocate Utils
":EQUB 0
1160 .vers:EQUS "1.07"
1170 .copyright:EQUB 0:EQUS "(C) 1994":
EQUB 0
1180 .helpword:EQUS "MRUT":EQUB 13
1190 .helpmess:EQUS "MSHIFT <start> <en
d> <dest>":EQUB 13
1200 EQUS "RELOC <start> <end> <dest>":
EQUB 13:EQUW &FF0D
1210 :
1220 .service
1230 PHP:CMP #4:BNE notcom:JMP command
1240 .notcom
1250 CMP #6:BNE notbrk:JMP break
1260 .notbrk
1270 CMP #9:BEQ help:PLP:RTS
1280 :
1290 .help
1300 PHA:TYA:PHA:TXA:PHA
1310 LDA (&F2),Y
1320 CMP #&D:BEQ nameonly
1330 LDX #0
1340 .help1
1350 INY:LDA (&F2),Y
1360 CMP helpword+1,X:BNE helpend
1370 CMP #&D:BEQ help2
1380 INX:JMP help1
1390 .help2
1400 JSR showname:JSR osnewl
1410 LDX #0
1420 .help3
1430 LDA helpmess,X
1440 CMP #&FF:BEQ help4
1450 JSR osasci
1460 INX:JMP help3
1470 .help4
1480 JMP helpend
1490 .nameonly
1500 JSR showname:JSR space:JSR space
1510 LDX #255
1520 .showword
1530 INX:LDA helpword,X
1540 JSR osasci
1550 CMP #13:BNE showword
1560 JMP helpend
1570 :
1580 .showname
1590 JSR osnewl
1600 LDX #0
1610 .name1
1620 LDA title,X:BEQ name2
1630 JSR osasci
1640 INX:BNE name1
1650 .name2
1660 JSR space:LDX #0
1670 .name3
1680 LDA vers,X:BEQ name4
1690 JSR osasci
1700 INX:BNE name3
1710 .name4
1720 JMP osnewl
1730 :
1740 .helpend
1750 PLA:TAX:PLA:TAY:PLA:PLP:RTS
1760 :
1770 .break
1780 PHA:TYA:PHA:TXA:PHA
1790 LDA #&BA:LDX #0:LDY #&FF:JSR osbyt
e
1800 CPX &F4:BNE ignore:JSR copyerr
1810 .ignore:JMP helpend
1820 .copyerr
1830 LDY #0
1840 .errloop
1850 LDA (&FD),Y:STA &100,Y:BEQ errend
1860 INY:BNE errloop
1870 .errend
1880 STA &FD:LDA #1:STA &FE
1890 RTS
1900 :
1910 .command
```

```
1920 STA tempa:STY tempy:STX tempx
1930 JSR find:BCS notthisrom
1940 LDA vect:STA &F2:LDA vect+1:STA &F
3
1950 DEY:JMP (num)
1960 :
1970 .notthisrom
1980 LDX tempx:LDY tempy:LDA tempa:PLP:
RTS
1990 :
2000 .find
2010 LDA (&F2),Y:AND #&DF
2020 TYA:CLC:ADC &F2:STA vect
2030 LDA &F3:ADC #0:STA vect+1
2040 LDA #comtable MOD256:STA addr
2050 LDA #comtable DIV256:STA addr+1
2060 .check
2070 LDY #0:LDA (addr),Y:BEQ nomatch
2080 .nextword
2090 LDA (addr),Y:BMI getadr
2100 LDA (vect),Y:AND #&DF
2110 CMP (addr),Y:BNE different
2120 INY:BNE nextword
2130 .different
2140 INY:LDA (addr),Y:BPL different
2150 INY:INY:TYA:CLC:ADC addr:STA addr
2160 LDA addr+1:ADC #0:STA addr+1
2170 JMP check
2180 :
2190 .getadr
2200 STA num+1:INY:LDA (addr),Y:STA num
2210 CLC:RTS
2220 :
2230 .nomatch
2240 SEC:RTS
2250 :
2260 .readhex
2270 LDA #0:STA num:STA num+1:JSR skip
2280 .hexloop
2290 LDA (&F2),Y:CMP #&D:BEQ rhexit
2300 CMP #32:BEQ rhskip
2310 CMP #ASC",":BEQ comma
2320 JSR mult16:JSR checkhex
2330 CLC:ADC num:STA num
2340 LDA num+1:ADC #0:STA num+1
2350 INY:CPY #5:BNE hexloop
2360 .badnum
2370 BRK:EQUB 28:EQUS "Bad hex":BRK
2380 :
2390 .rhskip
2400 JSR skip:CMP #ASC",":BNE rhexit
2410 .comma
2420 INY
2430 .rhexit
2440 LDA lim:CMP num:LDA lim+1:SBC num+
1
2450 BCS rhend
2460 .toobig
2470 BRK:EQUB 20:EQUS "Too big":BRK
2480 .rhend
2490 RTS
2500 :
2510 .checkhex
2520 CMP #ASC"0":BMI badnum
2530 CMP #ASC"G":BPL badnum
2540 CMP #ASC"A":BMI decimal
2550 SEC:SBC #7
2560 .decimal
2570 SEC:SBC #ASC"0"
2580 RTS
2590 :
2600 .mult16
2610 LDX #3
2620 .multloop
2630 ASL num:ROL num+1
2640 DEX:BPL multloop
2650 RTS
2660 :
2670 .skip
2680 LDA (&F2),Y:CMP #32:BNE noskip
2690 INY:BNE skip
2700 .noskip
2710 RTS
2720 :
2730 .space
2740 PHP:LDA #32:JSR oswrch
2750 PLP:RTS
2760 :
2770 .comtable
2780 EQUS "MSHIFT":EQUB (mshift DIV256)
:EQUB (mshift MOD256)
2790 EQUS "RELOC":EQUB (relocate DIV256
):EQUB (relocate MOD256)
2800 EQUB 0
2810 :
2820 .mshift
2830 LDA (&F2),Y:CMP #&0D:BNE mshift2
2840 .mshsyntax
```

```
2850 BRK:EQUB 16:EQUS "Syntax: MSHIFT <
start> <end> <dest>":BRK
2860 .mshift2
2870 LDA #0:STA lim:LDA #&7C:STA lim+1
2880 JSR readhex:LDA (&F2),Y:CMP #13:BE
Q mshsyntax
2890 LDA num:STA oldloc:STA srce
2900 LDA num+1:STA oldloc+1:STA srce+1
2910 JSR readhex:LDA (&F2),Y:CMP #13:BE
Q mshsyntax
2920 JSR mshift3
2930 LDA #0:PLP
2940 RTS
2950 :
2960 .mshift3
2970 LDA num:STA oldend:LDA num+1:STA o
ldend+1
2980 JSR readhex
2990 LDA num:STA newloc:STA dest
3000 LDA num+1:STA newloc+1:STA dest+1
3010 SEC:LDA oldend:SBC oldloc:STA len
3020 LDA oldend+1:SBC oldloc+1:STA len+
1
3030 :
3040 LDA oldloc:CMP newloc
3050 LDA oldloc+1:SBC newloc+1
3060 BCS shiftup
3070 LDA newloc:CMP oldend:BNE ntequal:
CLC
3080 .ntequal LDA newloc+1:SBC oldend+1
3090 BCC shiftdown
3100 :
3110 .shiftup
3120 LDX len+1:BEQ loloop1
3130 LDY #0
3140 .loop1
3150 LDA (oldloc),Y:STA (newloc),Y
3160 INY:BNE loop1
3170 INC oldloc+1:INC newloc+1
3180 DEX:BNE loop1
3190 .loloop1
3200 LDX len:BEQ finish1
3210 LDY #0
3220 .loop2
3230 LDA (oldloc),Y:STA (newloc),Y
3240 INY:DEX:BNE loop2
3250 .finish1
3260 RTS
3270 :
3280 .shiftdown
3290 DEC oldend+1
3300 CLC:LDA newloc:ADC len:STA newend
3310 LDA newloc+1:ADC len+1:STA newend+
1
3320 DEC newend+1
3330 LDY #&FF:LDX len+1:BEQ loloop2
3340 .loop3
3350 LDA (oldend),Y:STA (newend),Y
3360 DEY:CPY #&FF:BNE loop3
3370 DEC oldend+1:DEC newend+1
3380 DEX:BNE loop3
3390 .loloop2
3400 LDX len:BEQ finish2
3410 .loop4
3420 LDA (oldend),Y:STA (newend),Y
3430 DEY:DEX:BNE loop4
3440 .finish2
3450 RTS
3460 :
3470 .relocate
3480 LDA (&F2),Y:CMP #&0D:BNE relocate2
3490 .relsyntax
3500 BRK:EQUB 16:EQUS "Syntax: RELOC <s
tart> <end> <dest>":BRK
3510 .relocate2
3520 LDA #0:STA lim:LDA #&7C:STA lim+1
3530 JSR readhex:LDA (&F2),Y:CMP #13:BE
Q relsyntax
3540 LDA num:STA oldloc:STA srce
3550 LDA num+1:STA oldloc+1:STA srce+1
3560 JSR readhex:LDA (&F2),Y:CMP #13:BE
Q relsyntax
3570 JSR mshift3
3580 :
3590 LDA dest:STA newloc
3600 CLC:ADC len:STA newend
3610 LDA dest+1:STA newloc+1
3620 ADC len+1:STA newend+1
3630 SEC:LDA #0:SBC len:STA ptr
3640 LDA #0:SBC len+1:STA ptr+1
3650 SEC:LDA newloc:SBC srce:STA diff
3660 LDA newloc+1:SBC srce+1:STA diff+1
3670 :
3680 .reloc
3690 LDY #0:LDA (dest),Y
3700 TAY:AND #&0F
3710 TAX:LDA rdata,X
```

Date Handling Functions and Procedures (Part 2)

by Paul Cuthbertson

This second and concluding part of the Workshop on date handling routines deals with the formatting of dates for output. The code for this is listed opposite. Lines 120 to 140 set up arrays containing names of days etc, and these should be included as part of your main program.

The date formatter requires a packed date (explained in part 1), plus a format string, as parameters, and returns a date as a string developed using the format string as a template. A format string is built up out of the following elements in any order:

YYYY causes the year to be inserted into the string in the form 1994. It always has four digits.

YY causes the year to be inserted in the form 94 or 04. It always has two digits including a leading zero where necessary. It is assumed that the user knows which century is being referred to!

M inserts the month as a number, e.g. 12 for December, 5 for May.

MM does the same but in the case of January to September it inserts a leading space as well.

MMM does as above but adds a leading zero if required, rather than a leading space.

MMMM inserts the first three letters of the month, e.g. December would be represented as 'Dec', and January as 'Jan'; remember to set up the string arrays to accommodate this feature. If capital letters are used in the array then, of course, you will get 'DEC' or 'JAN'.

MMMMM inserts the entire name of the month. Caution: if you want the dates to come out in neat columns you may need to include trailing spaces in the string array elements. September is the longest name, with nine letters; a suggestion is that you make all the string array elements nine or ten characters long.

D to **DDDDD** does the same thing for days as M to MMMMM does for months. Again if you want neat columns then include trailing spaces in the day names to make everything at least as long as Wednesday (which I always maintain is the longest day of the week anyway!).

Any other character, be it space, letter, digit or punctuation will be inserted as it is encountered in the format string.

Some examples:

FORMAT STRING	POSSIBLE STRING
DDD MMM YY	12 02 86
DDD.NM.YY	06. 2.09
DDDDD DD (YYYY)	Wednesday 5 (2025)
YY MM DD	86 3 19

To summarise the routines and describe their use:

FNdatesp(date%,format$) is the main function and returns a string containing the date in printable form according to the contents of *format$* .

FNd0to6(date%) returns a number between 0 and 6, representing a day of the week, to reference *day$()* .

FNdayif(firstdate%,secondate%) returns an integer equal to the number of days between the two dates. If the first date is earlier then the result is positive. If the first date is the greater then *FNdayif* calls itself recursively just once with its own formal parameters reversed, then multiplying by minus one. Note that using the day strings can be slow if years far from 1986 (the base date used by the program) are entered, as *FNdayif* has a lot of work to do.

PROCdmyout simply acts as a collection point for **FNyearip** , **FNmonip** , *FNdayip* and *PROCfeb* to prevent the need to call them all separately on various occasions.

PROCyok performs a check to ensure that year boundaries have not been violated, and sets the date to 31 December 9999 or 1 January 1752 if they have, sounding a warning as well. The variables *d%* , *m%* , and *y%* are global to *PROCdmyout* and *PROCyok* ; so exercise caution if you want to use these particular routines on their own (I don't think you will).

FNdayip(date%) strips the days part of a date from a packed date and returns it as an integer.

FNmonip(date%) and **FNyearip(date%)** do likewise for the month and year. The usual postfix convention applies to the function names, 'ip' referring to 'packed integer'.

All the procedures and functions referred to must be included (the last four were also listed with part 1), for date output.

The magazine disc contains a demonstration program which includes a number of useful combinations of different functions and procedures, as well as a complete trial of all the formatting possibilities. Overall, the date handlers provide a means for you to use dates easily within your own programs without the extra work imposed by writing such handlers yourself.

Note: this Workshop was first published in BEEBUG Vol.5 No.6.

We thus come to the end of our last Workshop, one of the longest running series to have been published in BEEBUG. With a few breaks, the Workshop series has appeared regularly since Vol.3 No.1 in May 1984. And to reveal a small secret which may be of interest to long-standing readers, the 'name' Surac under which many of the Workshops appeared was intended to give a sense of continuity, though different authors wrote Workshops at different times. And the name Surac - it was chosen by David Fell, the originator of the Workshop concept on the grounds that he could think of nothing at all for which the letters might stand as an acronym. Well, now you know.

```
 120 DIM mon%(11),day$(6),mon$(11):FORa
%=0TO11:READmon%(a%),mon$(a%):NEXT:FORa%
=0TO6:READday$(a%):NEXT
 130 DATA 31,JANUARY  ,28,FEBRUARY ,31,
MARCH   ,30,APRIL   ,31,MAY    ,30,J
UNE   ,31,JULY   ,31,AUGUST  ,30,SE
PTEMBER,31,OCTOBER  ,30,NOVEMBER ,31,DEC
EMBER
 140 DATA SUNDAY   ,MONDAY   ,TUESDAY
,WEDNESDAY,THURSDAY ,FRIDAY   ,SATURDAY
 1450 :
 1460 DEFFNstr(u%,sel%):LOCALu$:@%=@%AND
&FFFFFF:u$=STR$(u%):IFLEN(u$)>1=u$ELSEIF
sel%-1THEN="0"+u$ELSE=" "+u$
 1470 :
 1480 DEFFNcseq(j$,q%):IFq%>LEN(j$)=""
 1490 LOCALv$,x$,y$:x$=MID$(j$,q%,1):REP
EAT:v$=v$+x$:q%=q%+1:y$=x$:x$=MID$(j$,q%
,1):UNTILx$<>y$:=v$
 1500 :
 1510 DEFFNd0to6(d%):LOCALl%:l%=(FNdaydi
f(324403735,d%))MOD7:IFl%<0=l%+7ELSE=l%
 1520 :
 1530 DEFFNdatesp(d%,f$):LOCALw$,p%,f1$,
```

```
1%:p%=1:REPEAT:f1$=FNcseq(f$,p%):l%=LEN(
f1$):p%=p%+l%:ONINSTR("*DMY",LEFT$(f1$,1
))GOSUB1550,1560,1630,1700ELSEGOSUB1740
 1540 UNTILf1$="":=w$
 1550 RETURN
 1560 ON1%GOSUB1580,1590,1600,1610,1620E
LSEGOSUB1620
 1570 RETURN
 1580 w$=w$+STR$(FNdayip(d%)):RETURN
 1590 w$=w$+FNstr(FNdayip(d%),1):RETURN
 1600 w$=w$+FNstr(FNdayip(d%),0):RETURN
 1610 w$=w$+LEFT$(day$(FNd0to6(d%)),3):R
ETURN
 1620 w$=w$+day$(FNd0to6(d%)):RETURN
 1630 ON1%GOSUB1650,1660,1670,1680,1690E
LSEGOSUB1690
 1640 RETURN
 1650 w$=w$+STR$(FNmonip(d%)+1):RETURN
 1660 w$=w$+FNstr(FNmonip(d%)+1,1):RETUR
N
 1670 w$=w$+FNstr(FNmonip(d%)+1,0):RETUR
N
 1680 w$=w$+LEFT$(mon$(FNmonip(d%)),3):R
ETURN
 1690 w$=w$+mon$(FNmonip(d%)):RETURN
 1700 ON1%GOSUB1720,1720ELSEGOSUB1730
 1710 RETURN
 1720 w$=w$+FNstr(FNyearip(d%)MOD100,0):
RETURN
 1730 w$=w$+STR$(FNyearip(d%)):RETURN
 1740 w$=w$+f1$:RETURN
 1750 :
 1760 DEFFNdaydif(d1%,d2%):IFd1%>d2%=FNd
aydif(d2%,d1%)*-1
 1770 LOCALy1%,y2%,m%,r%:y2%=FNyearip(d2
%):y1%=FNyearip(d1%):r%=(y2%-y1%)*365:mo
n%(1)=28
 1780 FORm%=y1%TOy2%:r%=r%-FNlpyr(m%):NE
XT
 1790 m%=FNmonip(d1%):IFFNlpyr(y1%)ANDm%
>1r%=r%-1
 1800 IFm%>0FORy1%=0TOm%-1:r%=r%-mon%(y1
%):NEXT
 1810 m%=FNmonip(d2%):IFFNlpyr(y2%)ANDm%
<2r%=r%-1
 1820 IFm%>0FORy2%=0TOm%-1:r%=r%+mon%(y2
%):NEXT
 1830 =r%+FNdayip(d2%)-FNdayip(d1%)
32700 DEFPROCa(A$,B$):VDU22,7:G%=(40-(LE
NA$))/2:H%=(40-(LENB$))/2:FORT%=8TO9:PRI
NTTAB(G%-1,T%);CHR$141;CHR$130;A$:NEXT
32701 PRINTTAB(18,12);CHR$134;"by"TAB(H%
-1,14);CHR$134;B$TAB(7,16)CHR$130;"Press
any key to continue":G=GET:CLS:ENDPROC  B
```

MeReloc (Continued from page 39)

```
3720 BMI update
3730 TYA:ASL A
3740 PHP
3750 SEC:SBC rdata,X
3760 PLP:ROL A:ROL A
3770 TAY:AND #&1F
3780 TAX:TYA
3790 ROL A:ROL A
3800 AND #3:TAY:LDA rdata2,X
3810 .loop5
3820 DEY
3830 BMI update
3840 LSR A:LSR A:BNE loop5
3850 :
3860 .update
3870 AND #3:CMP #3:BNE update2
3880 PHA
3890 LDY #1:LDA (dest),Y
3900 CLC:ADC diff:STA srce
3910 TAX:INY:LDA (dest),Y
3920 ADC diff+1:STA srce+1
3930 CPX newloc:SBC newloc+1:BCC update
1
3940 LDA newend:CMP srce
3950 LDA newend+1:SBC srce+1:BCC update
```

```
1
3960 LDA srce+1:STA (dest),Y
3970 DEY:TXA:STA (dest),Y
3980 .update1
3990 PLA
4000 .update2
4010 CLC:ADC dest:STA dest
4020 BCC adjustpointer
4030 INC dest+1
4040 .adjustpointer
4050 INC ptr:BNE reloc
4060 INC ptr+1:BNE reloc
4070 LDA #0:PLP
4080 RTS
4090 :
4100 .rdata
4110 EQUD &81038200:EQUD &81828206
4120 EQUD &81810F81:EQUD &81838314
4130 .rdata2
4140 EQUD &AAAA99B9:EQUD &99A99999
4150 EQUD &66AAA5AA:EQUD &EEEEEEEE
4160 EQUD &77FFF7FF
4170 :
4180 ]:NEXT
4190 ENDPROC                            B
```

Acorn COMPUTING

will soon be the only magazine that caters for YOU and all BBC Micro machine owners

Are you one of many people looking for a magazine that caters for owners of BBC Micro machines? Let us tell you about Acorn Computing...

Every month we provide comprehensive coverage of education and public domain. Acorn Computing is exactly what you are looking for.

For only £2.95 Acorn Computing is the best buy for owners of any Acorn machines.

Plus...

It gets even better when you subscribe because, in addition to having your reserved copy delivered postage free to your home, subscribers get a special disk containing ingenious and varied new programs written specially for BBC B or Master machines. Every month we have complete programs written specially for you and your computer.

Special subscription offer

To top all of this we have a special subscription offer we have put together for BeeBug readers only. Take out a subscription to Acorn Computing today and we'll send you an exclusive pack of the last six months' subscriber disks for 8-bit machine owners. Not only will you secure interesting reading, useful information and future disks, but you'll also have the last six months' subscriber disks to start off your collection.

This is a great opportunity to save some money, as an ongoing quarterly direct debit subscription costs just £6.73 each quarter, and you get an extra issue each year at no cost.

To SUBSCRIBE and claim your free gift call our 24 hour HOTLINE
051-357 1275

Last Bit

1st course

Marshal Anderson finally throws in the towel.

Last month I was ranting on about how BBC Basic was such a Jolly Good Idea because it was a structured language (almost). Let's start this last journey Beebwards with a look at the other bits of the language that make life so exciting. From the point of view of structure, all Basics give us FOR-TO-EXT to go loopy with, but BBC Basic went one better than that.

AGAIN AND AGAIN

REPEAT-UNTIL were a welcome addition to the language. It's important, though, to realise that these do nothing that the language couldn't do before. FOR-TO-NEXT is a loop that must be executed a specific number of times once it has been entered, nothing that happens whist the program is in the loop can affect the length of it (actually, that's not strictly true, you can change the value of the counting variable to drop out early or extend your stay - *but you shouldn't*). If you want to REPEAT something UNTIL a certain set of circumstances applies, with other Basics you had to set up a loop with GOTO, so consider the following:

```
10 REM Use GOTO to repeat something
until the N key is pressed
20 A$=GET$
30 IF A$"N" THEN GOTO 20
```

```
10 REM Now with REPEAT UNTIL
20 REPEAT
30 A$=GET$
40 UNTIL A$="N"
```

Those of you that are still awake will have noticed that the REPEAT-UNTIL method actually takes more lines, but that's not really important. You might

also argue, probably correctly, that the machine will be doing the same job in both cases; i.e. it will go back to line 20 if 'N' isn't pressed.

BUT THAT'S NOT THE POINT!

The second example is better because it makes more sense in terms of normal language and the way we do things in real life. For that reason it's got to be easier to understand. As an aside, you might have noticed that BBC Basic gets quite sniffy if you try to re-start a FOR-TO-NEXT loop that hasn't been previously finished. It will let you do it a few times (10, I think) until it comes up with 'Too many FORs'. The reason for this is that Basic keeps a 'stack' of line numbers for the beginnings of these loops - if you keep piling on the loop starts without taking off the loop ends things get sticky. I remember that this really irritated me when I first started on the Beeb, I was using an adventure game generator I had written on the TRS 80 that positively relied on re-starting these loops. I eventually replaced them with REPEAT-UNTIL loops on the Beeb and, in retrospect, often wondered how Tandy Level II Basic was able to cope with this rather bizarre bit of programming.

So BBC Basic provides two useful, ready-made loop structures and it's always worth looking at these before you try to 'grow your own' with GOTOs.

POKING ABOUT

As I pointed out last month, a lot of the machines of the time provided the commands PEEK and POKE. The main purpose of these commands was to provide adolescent male programmers of any age with boundless opportunities to snigger a lot.

Secondary to this they provided the Basic programmer with direct access to the memory, with potentially disastrous results. The idea was that the language designers didn't have to do so much work, they just let the programmer get at everything in the hope that they would get it right in the end. If you are a believer in conspiracy theory you might also think they did this so that they could make a fortune writing books like, '100 Secret POKES For Boys' as the manuals were always incredibly reticent about what you could actually do with these commands.

For all that, PEEK and POKE were incredibly powerful commands - in the right hands. They let you access system variables directly so they were very fast, and they could be used for writing things directly onto the screen, so they were helpful in games writing. They were also used for creating machine code programs.

WE DON'T WANT TO KNOW ABOUT MACHINE CODE

But you should because it's important in the development of BBC Basic. If you wanted to write machine code on any of these other eight bit machines you either poked it in a byte at a time as a series of numbers - one error could crash the machine and you had no way of knowing where it was - or, you could buy a piece of software called an assembler. This made machine code writing a lot easier but also caused a lot of bother if you wanted to write something that combined Basic and machine code.

BBC Basic cracked this problem by building the assembler into Basic. This means it's really easy to write little bits of machine code via Basic and, I think, encouraged a lot more people to mess around with machine code than would have done in other circumstances.

That solved the machine code bit but how did BBC Basic cope with access to systems variables and other tricks that PEEK and POKE did on other machines. From the point of view of the graphics we have the VDU commands. These are incredibly flexible and allow you to do all sorts of really clever things with the screen without having to resort to machine code of any sort. Page 378 in the original User Guide gives a nice little summary; Appendix G in the Master Welcome Guide shows how things developed, but neither of them takes you into the real depths of VDU23. This is a really good one and well worth finding out about as VDU23 gives you direct control of the video chip. You don't have to know how it works, and this is the sort of command that only the bravest actually experiment with, but if you go through your back-copies of BEEBUG and the other Beeb based magazines you're bound to find plenty of hints and tips on what you can do with it.

The other 'machine level' commands available in Basic are the *FX commands. These represent a bit of a rag-bag of functions - it seems to be all those things that didn't want to fit in anywhere else. However, these really do beg for investigation; they're all those things you can't imagine any use for until you really need them, but when you do ...

All this, of course, is not to say that BBC Basic doesn't actually have its own PEEK and POKE commands. Not only does it have them, it has them with knobs on. You probably won't find much use for BBC Basic's 'indirection operators' but, rather than just loading and reading one byte of memory they can get at up to 256 at a time. The '?' reads or writes one byte, the '!' does four and '$' can read and write whole strings to memory. If you can think of any use for these, considering the wide range of machine access BBC Basic gives

you in other forms, then you'll find details on page 409 of the BBC User Guide.

BUT

You won't find any information about indirection operators in the Master's Welcome Guide. 'And a good thing too', I hear you cry. Well, not really. While Acorn has been developing BBC Basic into a better and better language it seems to have been putting at least as much R & D into manual writing. When the first User Guide came out with the BBC A/B in 1982, Acorn realised they had made a grave mistake. Written by John Coll and edited by David Allen (hang your heads in shame) the guide was unacceptably concise. It listed all the Basic keywords in alphabetical order which meant users could actually find them. The tutorial section actually taught people things that were useful, all features were documented and the technical stuff was there but neatly tucked away in appendices where only those who needed it would go.

Not only did this put a whole swathe of computer-book hacks on the dole (the ones who write books like 'How to Program Your Popsicle 9000' and 'The Sinclear Manual Explained') it also seemed to make Acorn feel they somehow weren't being taken seriously. After all, if everything you need to know about a computer is in the manual then where's the fun?

The Master Manual or (un)Welcome Guide hit upon the trick of telling you just enough about everything to let you know that you didn't know anything useful about anything. Anyone who wanted to know something (about anything) had to spend about £30 on the Reference Manual (parts 1 & 2) and buy guides to View and ViewSheet at £10 a throw. The most tantalising bit of the Welcome Guide for us programmers is

the single page devoted to BAS128, the new version of Basic provided with the Master. This came on the Welcome disc or tape and, once loaded, gave you access to all 128K of RAM instead of the standard 32K + shadow screens. Hands up all the poor saps who actually believed that page's closing sentence, 'Technical information relating to BAS128 can be found in the Reference Manual.'

If you decide to go on to program on the Arc (or whatever they call them now) you will find that Acorn have continued in this vein. You'll need a new set of book shelves and an extra £100 or so on your budget if you want the relevant reference works and Basic manual.

If you are still a beginner, or consider yourself one, then don't stop just because we have. People program for lots of reasons and the old Beeb will continue to provide what many of you want. For sheer mind-stretching exercise, why not give machine code a go, have a look through the back-copies of BEEBUG, especially at Mr Toad's disreputable column, and teach yourself. Keep any eye out for books on programming in general, local libraries keep stuff for years, and try to learn some of the skills the professionals use in development.

If you want to move on to a new machine the Arc has to be the one. You will find Basic V (5) friendly and familiar and you don't have to spend a fortune. A second hand A3000 with a little judicious upgrading will likely cost you less than a new Master did. Add to that a copy of Basic V: A Dabhand Guide to tell you just about the bits of Basic V that you need to know and you'll be well into the 32 bit age.

Which ever way you decide to go, I'm sure you'll enjoy it and may The Force be with you all the way. B

Extended Keyboard (Part 2)

Andrew Roland concludes his package to expand your keyboard.

In part one I presented a program which turns the Shift-Lock key into an Alt key ¨ a special shift key for entering foreign letters and symbols. This month we will look more closely at the ROM and design new keyboard layouts.

SETTING UP THE SYSTEM

Initially, *AltRom* needs no setting up other than the boot sequence described in the last issue. If changes to the defaults are required, *KeyCaps* provides a friendly means of doing so. However, you can automate the process by using star commands in your boot file.

Firstly, the character set should be selected with *IBM (which can be reversed with *MASTER) and your printer also set to the IBM character set. Next, the country (i.e. keyboard layout) may be altered if the default (UK) doesn't suit, e.g. *KEYB France. If you type *HELP KEYB a full list of countries supported is given, and *KEYB on its own informs you which country is the current one.

CHARACTER SETS AND YOUR PRINTER

Unfortunately, the Master's built-in characters do not correspond to those of any printer. My solution is to use the IBM character set, which most printers incorporate these days, and change the computer's character set to match. Ensure your printer is set to what is referred to in printer manuals variously as 'graphics', IBM set #2 or Code Page #437. You may need to alter some DIP switches in the printer. I assume the US set, not the UK one.

If your printer has different characters between 128 and 255 (or only italics),

you can use a character designer such as *CharDes* on the Master's Welcome disc to make the computer match your printer. Alternatively, you could use J.R.Barker's program (BEEBUG Vol.9 No.7 disc only) which makes the printer match the Master. You will also need to modify the keyboard layouts in *KeyMapG* as explained below.

DEFINING KEYBOARD LAYOUTS

As most people will only want to add characters to, or change, the 'shadow keyboard' accessed by the Alt key, we will start there. You will probably find it easier to follow this explanation if you have *KeyMapG*, lines 1560 onwards, in front of you while reading it.

The layout is defined in the form of DATA statements <key>=<code>, where <key> is the legend on the key and <code> the character you want it to produce. So if we want Alt-Q to produce, say, ñ enter q=164 (the ASCII code for ñ) somewhere *after* the 'DATA >>' in line 1580 ¨ on the end of line 1650 is best. Also, add Q=165 so that Alt Shift-Q gives Ñ ¨ Caps lock also forces the capital. If a DATA statement must include a comma, enclose it all in speech marks (e.g. ",=249"), and if it includes a speech mark both double it and enclose it in speech marks thus: """"=130".

If you run out of Alt key combinations, you can also use Ctrl, though *not* with Alt, so it must come before 'DATA >>'. Line 1570 shows one example of this:

```
DATA [E]=138
```

which sets Ctrl-E to è. You can use capital letters E-Z plus [\ | ^ and _.

NEW COUNTRIES

New keyboard layouts are called *countries*. To keep the work to a minimum, KeyMapG sets up a default

layout for each country and you only supply the changes. It is important to remember that the changes are always in terms of the key legend, not in terms of any subsequent change. So if you accidentally define a change twice, the second will overwrite the first. There are two exceptions: Shift-@ becomes 0 and 0 becomes ^ (Shift-0 is ~), so to make 0 into H, you need ^=H, not 0=H. This is not just to confuse you; it frees the ^/~ key to be a special accent key. You can see it being used like this in the German layout, for instance. Of course, it can be programmed to behave like an ordinary key.

The default layout is the same as the ordinary keyboard (the two keys mentioned above excepted), and the default for Alt-key combinations is to have no effect. Lines 1560 to 1650 then define a standard layout which is applied to *all* countries. As you can see, it consists mostly of Alt key definitions: these are the ones common to each layout. I feel it would be too confusing for them to alter every time you change to another country ¨ but you are free to do what you want.

There then follow the country definitions themselves. Each one starts with ***, a name (up to 12 characters) and a type byte, e.g.

        DATA ***,UK,0

The type byte is a code that controls which keys should be treated as letters and capitalised when Caps Lock is on (see table 1).

There then follow any changes which need to be made to ordinary keypresses. For the UK this is only:

        DATA 0=@,^=48,~=48

which returns the 0 and @ keys to their accustomed functions.

Following this there may be a section starting >> which defines Alt key combinations. For example,

        DATA >>,a=132,e=130,A=142

makes Alt-a into ä, Alt-e into é and Alt-A into Ä.

0	-	only the usual keys are letters
1	-	M is NOT a letter
2	-	:/* key is a letter
4	-	;/+ key is a letter
8	-	@ key is a letter
16	-	{/[key is a letter

Table 1: The type byte can be calculated by adding together all the numbers which apply.

Finally comes the only obligatory section, the accent definitions. It starts with /// and two codes which define the actions of the special accent key, ~/^ (in that order). A number of 1 to 4 leaves the key as an accent (see table 2), any other number generates that ASCII code, so the accent keys revert to being normal ones. There is no '=' in these two statements, e.g:

        DATA ///,~,^

restores the ^/~ key to its accustomed state, and:

        DATA ///,2,1

makes them into accent keys. They are optionally followed by further statements defining other keys as accents. The last line of data should be:

        DATA ***,*END*,0.

A note on data statements: Q=A and Q=65 are equivalent; [Q]=A makes Ctrl-A produce A. The second element should consist either of a single character or of a decimal number with at least two digits, so if you choose to make @ into the Tab key, use @=09 (@=9 would make @ produce "9", i.e. ASCII 57). 255 turns the key off, ASCII 255 therefore cannot be generated from the keyboard, but it is a non-printing character on the printer anyway. When altering letters, remember to do both capitals and lower case.

All the sections bar the accent keys are optional: you don't *have* to change anything. The program gives you complete freedom to alter any key except the function and cursor keys. Don't forget to use *KeyCaps* to check your work. It is not necessary to run AltRomB every time you want to check your altered layouts, as KeyCaps uses the data files directly.

The standard IBM character set does not include ¯ which is needed for Danish and Norwegian, so if you wish to use the Denmark or Norway layouts, you must ensure your printer is set to the correct country either using its DIP switches or sending ESC "R" 13 before printing. AltRom alters the screen characters automatically. For this reason, ensure that the sixth and seventh countries in the file are Norway and Denmark if you alter it in any way (if your printer doesn't support the Danish/Norwegian character set, see *ReadMe*).

Incidentally, if you always use a foreign keyboard layout you might like to change the key caps round by *carefully* lifting them off the keyboard with the aid of a thin screwdriver. Take care not to twist the key but lift it straight up.

ACCENTS
It is important to distinguish between the default accent key (~/^) and others you may set up. The former must always be defined immediately following / / / as explained above. Any other key can also be set up as an accent key, but it must first be given a unique character code. For instance, to make '/' into an accent key ¨ an acute, say ¨ and make '*' into '/' instead. In the first section of data statements in my country definition I therefore enter *=/. If I then make / into an accent, I

would have *two* acute accents (/ and *) and no ordinary /, so I choose a little used letter, '+' perhaps, and enter /=+. Then in the accent data section I enter +=1, where 1 is the number for the acute accent (see table 2).

This means I have sacrificed +, so if I'm not using the default accent key I can reinstate + there: / / /,+ ... If I want

1	- /	(acute)
2	- \	(grave)
3	- ^	(circumflex)
4	- ¨	(diuresis)

Table 2

more than two accent keys then I must sacrifice some little-used character ¨ which *cannot* be one with an ASCII code above 127 ¨ for this purpose.

ON THE DISC
AltRomB on last month's disc had been compacted to save memory as it was too large to work unaltered on a Master ¨ I had to use an Archimedes and the Tube emulator to develop it. This month's magazine disc carries the full length version, so if you are interested in the internal workings of the ROM or want to modify it, see *ReadMe*. You also get a full version of KeyCaps.

CONCLUSION
All the above may give the impression of complexity, but defining your own countries aside, you should now have a disc which boots into View (if you use this word processor) and from then on remains in the background until you want to print a screen, write a word like 'Æsop' or the school language assistant (remember them?) wants to type something and needs a foreign keyboard. I have tried to provide facilities which ought to have been part of the machine from the beginning. I hope you find them useful.
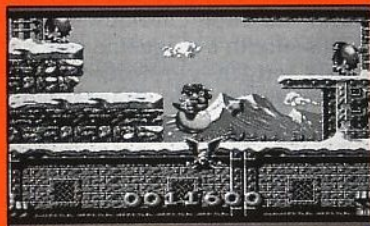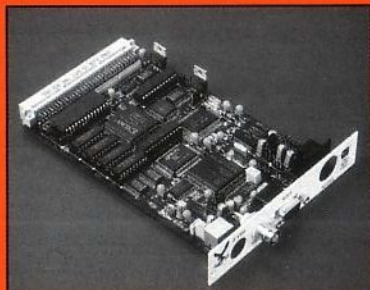
Ⓑ

RISC
user

RISC User, probably the most popular subscription magazine for the Archimedes, offers all the information you need as an Archimedes user. In every issue of RISC User you will find a wealth of articles and programs with professionally written reviews, lively news, help and advice for beginners and experienced users, and items of home entertainment.

The B5 size of RISC User allows a sophisticated design, big colour illustrations and pages full of information, and yet is still a convenient size to assemble into an easy-to-use reference library. Altogether, in its six years of existence, RISC User has established a reputation for a professional magazine with accurate, objective and informed articles of real practical use to all users of Acorn's range of RISC computers.

Contents of the latest Vol.7 Issue 5 of RISC User:

WINDOW ON THE WORLD
A look at weather satellites and the pictures which can be received using an Archimedes plus appropriate hardware and software.

SPREADSHEET SURVEY
A major survey of the spreadsheet packages available for the Arc, with timing comparisons and feature table.

NEW MODES FOR THE FUTURE
An investigation of new screen formats being designed by Acorn for 16-bit and 24-bit colour.

SHRINK
A substantial application for improving life on the Desktop for the benefit of all users.

GAMES ROUNDUP
A look at all the major games releases of recent months, and some still to appear.

EAGLE MK II
A review of Wild Vision's multimedia card with video digitising, 14-bit sound and MIDI capabilities.

IRLAM 24-bit COLOUR SCANNER
A review of a new high quality hand-held scanning device.

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

INTO THE ARC
A regular series for beginners.

SUBSCRIPTION DETAILS
As a member of BEEBUG you may subscribe to RISC User for the reduced rate of £18.40 (a saving of £1.50 on the normal subscription rate).
Overseas subscription rates are as follows:
£27.50 Europe and Eire, £33.50 Middle East, £36.50 Americas & Africa, £39.50 Elsewhere

PRESERVING TUBE TEST

Owners of a 6502 second processor and Wordwise Plus may be interested to know that any text in Wordwise Plus can be recovered after the machine has been switched off, so long as the tube has not been switched off. This is because all text is stored in the second processor memory, rather than the BBC memory.

CLEARING A REPEAT LOOP

The most obvious way to reset all arrays within a running program is to use CLEAR, and then dimension each array again. But you should not do this within a REPEAT loop (or a FOR-NEXT loop) because all loop pointers (and of course FOR-NEXT loop variables) are also cleared. If it is essential to clear arrays within a loop, make sure it is to a GOTO loop (perish the thought!), or clear the arrays one element at a time using a separate FOR-NEXT loop, rather than clear.

MORE MEMORY FOR DISC USERS

The 81 locations from &380 to &3D0 are reserved for workspace for the cassette filing system, and appear to be unused unless this filing system is called. This area makes an excellent additional user RAM scratch pad for the hard pressed disc user. It is ideal for passing parameter blocks in OSWORD calls and the like, and is even large enough to hold modest machine code frequently leaving the used &A00 buffer area free for other purposes.

VERTICAL PRINTING

Printing a string with each letter the correct way up but beneath the previous one is useful for labelling the Y-axis of a graph, amongst other applications. However, it it is tedious (and slow) to produce the desired effect with a PRINT TAB command for each letter even if a loop is used. Instead define a

text window only one character wide, but several lines deep, and the PRINT the string as normal. The Beeb does all the hard work by filling the window with the string of characters.

FILE DATA STORAGE

In a disc or cassette file, integers are stored in five bytes and real numbers are stored in six bytes regardless of their contents. So:

```
x=10.5:y%=8:PRINT#C,X,Y%
```

stores the 8 as a five byte integer and the 10.5 as a six byte real number.

```
X=10.5:PRINT #C,X,8
```

will store both numbers as a six byte real numbers. Failure to realize this can lead to the error message 'Type mismatch' on re-reading this data. Basic can, however, read an integer value and assign it to a real variable.

ROM REMINDER

In these days of ROMs on disc for loading into sideways RAM and sideways ZIF sockets for readily removable ROMs, it is difficult to know whether a particular ROM is going to be there when a command is issued. With the 1770DFS you can use *BUILD to create an ASCII file on disc with the same name as the ROM command. Include REM statement (to avoid causing an error) to the effect that the ROM concerned is not present. This will be printed if the ROM is not able to accept the command. For example :

```
*BUILD WORD
001 REM View ROM not plugged in
```

BUG IN SCROLLING

For an interesting (but useless) MOS bug in the Beeb screen scrolling run the following short program.

```
10 MODE3:PRINTSTRING$(45,CHR$10)
20 PRINTTAB(45,3)"0123456789"
30 VDU 28,0,24,79,0
40 VDU 30,11,11,11
```

B

Personal Ads

Games for BBC B or Electron, 9 tapes, 5.25" 40/80T send for full list and prices, Graphic Adventure Creator, Machine code language monitor, Science: Electromagnetic Waves, Electromagnetic Spectrum £10, Panasonic KX-P 1081 printer £40. Tel. Middlesex 081-977 3647.

A5000 RISC OS 3.1, 4Mb RAM, multi-sync monitor, 2x40Mb IDE drives, JP150 printer, Turbo-Driver, PC Emulator 1.81, Ovation, Thesaurus, Investigator III, Almanac plus 22 top games, originals in boxes, i.e. Dungeon, Gods, Birds of War etc. £1350 o.n.o. Tel. Leeds 0532 736943.

WANTED: Aries B-32 (preferably), B-20 or Watford 32k shadow RAM card. Tel. Derbyshire 0773 822477.

WANTED: Toolkit and Spellmaster. Tel. Herts 0462 685693.

BBC B issue 7 with DFS and Watford 5.25" single D/S 40/80T disc drive with PSU, fitted with Solidisk 2Mb RAM/ROM expansion, Shadow RAM, View 3, Viewstore, Viewsheet, MFNLQ, Toolkit, Discdoctor all in good condition £110 o.n.o. Tel. Lancs 0772 865087 4.30pm onwards.

WANTED: Masterfile II, geniune package with manual for BBC B on 5.25" disc, prefer version 2.44 or later. Tel. Suffolk 0842 860580 6-9pm.

BBC Acorn DFS OS1.20, 2 40/80T Cumana DS drives, Microvitec Cub colour monitor, 6502 2nd processor, Solidisk 4Mb 32SWR, lots of software and manuals, also

Smith Corona printer (Epson compatible) thrown in, separately or altogether. Offers? Discount for collecting. Tel. Suffolk 0787 280243 or 0532 306276.

11 Master 512's, fitted with various ROMs, 11 Microvitec 653 montiors, 1 AMX mouse, 8 Watford mouse, 1 Epson FX80, 1 Epson FX85, 1 Epson FX800, 2 Epson JX80, 1 Epson EX800, 2 Integrex ColourJet printers, 11 Cumana disc drives, 3 Acorn Electrons. All equipment in good working order offers welcome. Tel. Cordwainers College - London 081-985 0273 extn 222 (Mr Peter Curry).

A5000 L/C, OS 3.1, 4Mb RAM 40Mb HD, 5.25" board and DD, CC Faxpack and Compression, FFile, PCm V1.8, RU Hard Disc Companion, ArcFS 2.2, Arc DFS, AC and RU mags and discs, Arcscan 3, RU Thesaurus, PD discs £950. Tel. London 071-267 9076.

Master 128, fitted with BB Master ROM and Wordwise Plus, Philips colour monitor CM11342 /05G, Cumana double disc drive, mains powered, Epson FX80, complete set £250 o.n.o. BB Printwise disc and guide £5, BB Studio8 disc and guide £5, Advanced User Guide for BBC Micro £5, 30Hr Basic 250 page booklet £5, all small items include postage. Tel. Farnham 0252 710219.

Archimedes RGB 14" colour monitor AKF12 £85. Tel. North London 081-882 2592.

WANTED: Interbase, Interchart, Intersheet, ROM board or cartridge for Master, any programming language ROMs with manuals, PD

software for 512 DOS+. Tel. Bristol 0272 273366.

WANTED: Viglen PC style case for M128. Tel. 0634 374130.

WANTED: Faulty Panasonic KX-P1124 printer for printhead parts, or printhead only, printhead need not be working perfectly. Tel. 031-449 3956 eves.

A3000, Acorn stereo monitor, 4Mb RAM, RISC OS 3.1, 40Mb SCSI hard disc (external) + interface, Panasonic 24pin printer, manuals some software, all in excellent condition £625 o.n.o. may split. Tel. Essex 081-514 5995.

A3000, RISC OS 3.1, 4Mb (no monitor), discs and manuals £350, Pipedream 3 £30, Pipedream 4 £80, Easiwriter (2.07) £60, Superior Golf £10, or all for £500. Tel. Herts 0462 682961.

Lots of various hardware, software, books, manuals to be sold, please call for list. Tel. St Albans 0727 843600 extn 236 (ask for Paul).

A5000, 4Mb, 250Mb HD, software including Artworks, Impression II, Multistore and numerous games etc. still under warranty, under 12 months old, offers? Tel. East Sussex 0323 487700.

BBC B issue 7, plus disc drive, 2nd processor, Philips monitor and Juki printer £150 o.n.o. Tel. 0273 603825 eves.

BBC/ Master to Arc entry upgrade system, A310M, Star LC24-200 colour printer, Med

Res Philips monitor, IFEL 4Mb RAM/MEMC1a upgrade, interfaced 5.25" twin 40/80 disc drive in monitor plinth, serial link, PC Emulator V1.7, Dr DOS 5.0, considerable amount of Utility/Games software £540. Tel. Oakham 0572 821313.

A3000, 4Mb, Userport/midi interface fitted, offers? Tel. 0323 487700.

WANTED: BEEBUG Volumes 1, 5 and 6. Tel. Wolverhampton 0902 783299.

WANTED: Genlock for BBC B. Tel. Coventry 0203 610445 anytime.

WANTED URGENTLY: BBC Teletext adaptor V2.50 ROM for my Master 128. Tel. North Yorks 0751 473342.

Daisywheel printer, Triumph Adler wide carriage, spare wheel and ribbons £45, buyer collects, books for BBC B. Ring for list. Tel. Sheffield 0226 762450.

WANTED: Morley battery back upped 32k RAM cartridge for BBC Master. Tel. Edinburgh 031-650 2365.

A5000 L/C, 4Mb 40Mb hard disc, M/scan monitor, PipeDream 3, Educational software and games etc. Genuine reason for reluctant sale £1095 o.n.o. Tel. Bath 0225 864526.

Juki 6100 daisywheel printer £10 but buyer collects. Tel. Herts 0462 682961.

Archimedes RGB 14" colour monitor AKF12 £85. Tel. North London 081-882 2592.

THANKS FROM THE BEEBUG FRATERNITY

I have just read the *Editor's Jottings* and feel so sorry to think that there are only two more copies of BEEBUG to go. They have always given me the impetus to move on in something I am doing, and how to get out of a fix I have got into and could not resolve, although it is still a lot of black magic to me.

I have been a member of the BEEBUG fraternity since it started, and have a Beeb issue 1 dated 1981. I fitted a Watford RAM board, and I have filled all the space with ROMs. I still use tape and have an old daisy wheel printer.

A year ago I acquired a Master 512 with lots of discs, but as yet I have NOT been able to load down a single file.

I am well passed my "sell-by-date", and have found programming very puzzling, but with the help of BEEBUG I have managed to understand the basics of my issue 1 Beeb. I write all my letters on it, and use it for all our accounts, but it is nothing like the Master 512.

Because of the coming demise of BEEBUG I am interested in the reference to BBC PD in the feature by Robin Burton on page 30/31 of the Jan/Feb 1994 issue. What is it, and where do I find it?

Also, is there a club in the Brighton area, or a local Master 512 enthusiast who would help me, if you could put me in touch with them?

Many thanks for a most stimulating publication over the years. And to everyone involved in BEEBUG all the very best in the future.

Fred Hodgkins

Thanks for your appreciative comments. For reference, BBC PD is a Public Domain Software Library run by Alan Blundell. His address is 18 Carlton Close, Blackrod, Bolton BL6 5DL. And if anyone lives near Mr.Hodgkins and would like to get in touch, please write to him c/o the BEEBUG address and we will pass your letter on.

AND MORE BOUQUETS

I have now been a member for a number of years, and I am very sorry that time, and the advancement of present day Acorn computers, make it uneconomic to continue support for the 8-bit BBC micro and Master machines.

Over the years I have acquired some favourite programs from BEEBUG magazine, programs which I still use regularly today: an early but simple accounts program, Filer, Texbase and a Diary program are still as useful to me as they have ever been.

My reason for writing is to express my thanks for your magazine. I have had no formal training whatsoever for using a computer. All my knowledge has been gained by reading magazines and typing in programs to use.

I have watched with great interest my last three issues, and have been impressed with the very high standard. In particular, I would like to say "Thank you" to you and Ian Palmer for M-Base. What a great program for the Master 128. I look forward to my remaining issues of BEEBUG with eagerness. What further gems will there be in your excellent magazine?

T.G.Westwood

It has always been satisfying when readers have responded positively to programs we have published. I hope that the last two issues (for March and April) do live up to Mr.Westwood's expectations. B

ANNOUNCEMENT

With the termination of publication of BEEBUG, neither RISC Developments Ltd. nor Beebug Ltd. will be able to provide any sales or support for users of the 8-bit BBC micro, Master series etc. after 30th April 1994. If you do want to buy anything from us for your BBC micro then please make sure you send in your order by this date.

Square Dance

by Martin Richards

This program displays a series of expanding and contracting rectangles which change colours during the course of execution. The actual 'form' of the patterns is random, and therefore a number of runs will display different patterns. Some patterns will repeat after a while, whilst others will appear to keep on changing, giving an attractive display of coloured graphics.

This program was first published in BEEBUG Vol.1 No.10.

```
10 REM Program Square Dance
20 REM Version B2.0
30 REM Author  Martin Richards
40 REM BEEBUG  April 1994
50 REM Program subject to copyright
60 :
100 ON ERROR PROCerror
```

```
110 MODE 1
120 VDU 23;8202;0;0;0;
130 VDU 29,640;510;
140 X=0:Y=0
150 DX=RND(50):DY=RND(50)
160 C=RND(3)
170 REPEAT
180 GCOL 3,C
190 MOVE X,Y:DRAW -X,Y:DRAW -X,-Y
200 DRAW X,-Y:DRAW X,Y
210 X=X+DX:Y=Y+DY
220 IF ABS(X)>640 THEN DX=-DX:C=RND(3)
230 IF ABS(Y)>510 THEN DY=-DY:VDU 19,R
ND(3),RND(7);0;
240 A=INKEY(1):IF A=32 THEN RUN
250 UNTIL FALSE
260 :
1000 DEF PROCerror
1010 CLS
1020 REPORT:PRINT" at line ";ERL
1030 END
```

BBC Micro User Groups

We give below details of user groups catering for BBC micro users. Since we last published this information in the March issue, one further user group has come to our attention, *The Beeb Supporters Group* run by Mark Brocklehurst. Mark intends to produce an A4 newsletter covering the model B, Master 128, Master Compact and even the Electron. Subscriptions start at £7 for those in the UK, £7.60 for Europe and £8.50 for the rest of the world.

If you write to any of these user groups for information then do please enclose an SAE (stamped addressed envelope) for your reply - most of these groups are run in the organiser's 'free' time and often at their own expense. I am sure any offers of help, articles, programs etc. will be most welcome.

Beeb Developments User Group, 73 Spital Crescent, Newbiggin-by-Sea, Northumberland NE64 6SQ. Tel. 0670 521055.

The Beeb Supporters Group, 1 Park Avenue, Markfield, Leicester LE67 9WA.

8 Bit Software, 17 Lambert Park Road, Hedon, Hull HU12 8HF. Tel. 0482 896868.

ByteBack, 33 King Henry Mews, Enfield Lock, Middlesex EN3 6JS.

Destroyed Realities Disc Based Magazine, 82 Main Street, Pembroke, Dyfed, Wales SA71 4HH.

Solinet, 41 Wentworth Drive, Rainworth, Mansfield, Nottingham NG21 0FB.

Magazine Disc

April 1994

EXTENDED KEYBOARD (PART 2) - This month we have included the expanded versions of AltRomB and KeyCaps for those interested in understanding these programs

SCREEN SAVER - Leaving screens switched on with a static display can cause damage in the long run. Use this utility for automatic screen blanking.

CUBERT - This classic game, repeated from the early issues of BEEBUG, challenges cuddly Cubert to negotiate the treacherous cubes.

MERELOCS - This is a utility which will appeal to the more technically minded, as it enables data and programs to be relocated in memory.

SQUARE DANCE - This very short program provides a visually fascinating feast of moving patterns.

BEEBUG WORKSHOP - This month we have a second collection of date handling routines, this time for formatting dates, together with a complete demo program.

DISC MENU - This handy program provides a simple but effective way of constructing an informative menu system for each of your discs.

ENVELOPE PRINTING - For View enthusiasts, this utility provides a convenient means of transferring an address from a letter head onto an envelope.

MAGSCAN DATA - Bibliography for this issue of BEEBUG (Vol.12 No.10).

BONUS ITEMS

MASON - From the past - see this month's editorial - one of the classics from the BEEBUG archives.
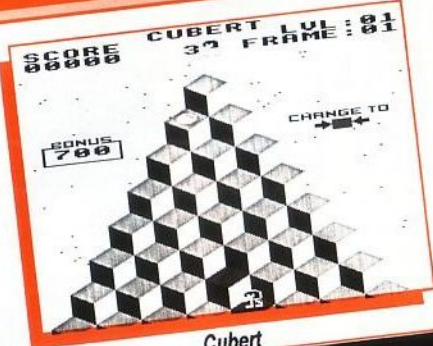
Cubert
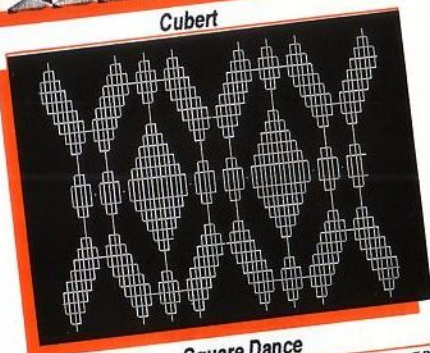
Square Dance
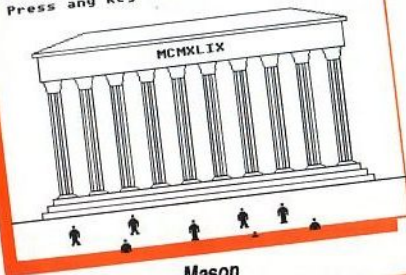
Mason

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.6 No.1) available at the same prices.
Prices are inclusive of VAT and postage as applicable. Sterling only please.

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