

Vol.6 · No.6 · November 1987

BEEBUG

FOR THE
BBC MICRO &
MASTER SERIES



Defining Characters
for an Epson Printer

☐ ROM FILING SYSTEM

☐ FORMATTED LISTINGS

☐ MODE 7 MENU GENERATOR

☐ DRACULA

£1.30

FEATURES

Mode 7 Menu Generator	6
Barry Christie Visuals - Formatted Listings	11
Epson Character Definer	14
The Comms Spot	20
A ROM Filing System Image Generator	22
Auto-Confirm Utility	27
Exploring Assembler (Part 5)	28
The Master Pages - Storing Variables and Procedures in SWR	41
Full Feature Dating for Wordwise Plus	45
Master Hints	46
Workshop - Long Precision Arithmetic (Part 2)	50
High Resolution Screen Dump for Mode 0	52
First Course - Colouring your Beeb (Part 1)	57
Archimedes Competition - The Winning Program	61
Dracula	64

REVIEWS

Pineapple's Diagram II	18
Adventure Games	32
Printer Survey	47
What You See Is What You Get	58
Books for Serious Users	60
Advanced Basic	62

REGULAR ITEMS

Editor's Jottings	4
News	4
Points Arising	13
Supplement	33-40
Master Hints	46
BEEBUG Technical	67
Postbag	68
Hints and Tips	69
Subscriptions & Back Issues	70
Magazine Disc & Cassette	71

HINTS & TIPS

GENERAL

VDU22 Mode Change
6502 Decimal Modes
Space Fire
A New HELP
Dead Centre
Easier Game Play
Protecting Basic Programs

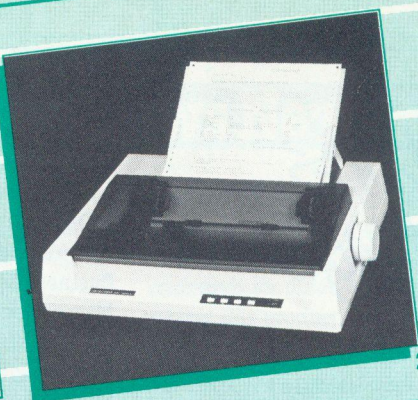
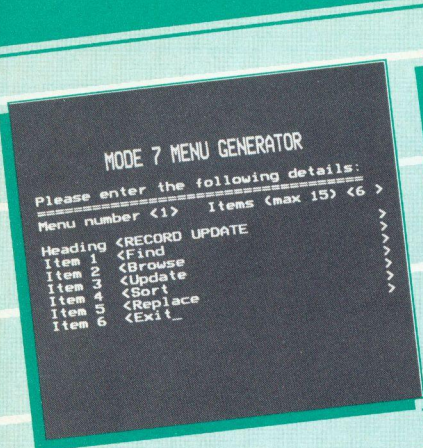
MASTER

Better Blanking
Interword and Spellmaster
Calling the Editor
Automatic Assembler Date Stamp

PROGRAM INFORMATION

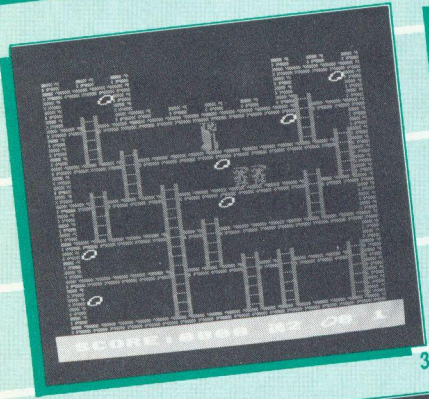
All programs listed in BEEBUG magazine are produced direct from working programs. They are listed in LISTO1 format with a line length of 40. However, you do not need to enter the space after the line number when typing in programs, as this is only included to aid readability. The line length of 40 will help in checking programs listed on a 40 column screen.

Programs are checked against all standard Acorn systems (model B, B+, Master, Compact and Electron; Basic I and Basic II; ADFS, DFS and Cassette filing systems; and the Tube). We hope that the classification symbols for programs, and also reviews, will clarify matters with regard to compatibility. The complete set of icons is given

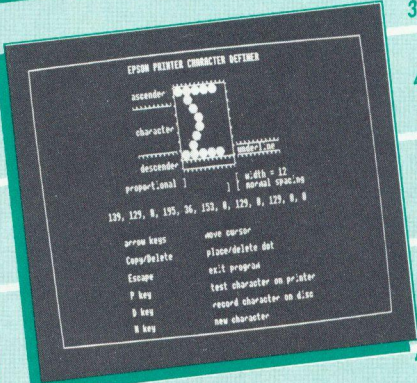


1. A Mode 7 Menu Generator

2. Printer Survey



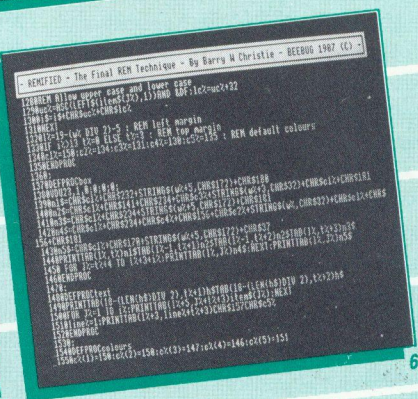
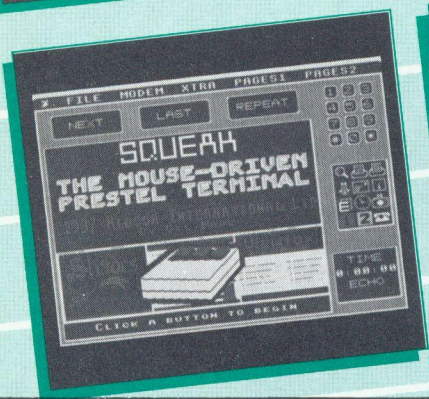
3. Dracula



4. Epson Character Definer

5. The Comms Spot-Squeak

6. Formatted Listings



5.

6.

below. These show clearly the valid combinations of machine (version of Basic) and filing system for each item, and Tube compatibility. A single line through a symbol indicates partial working (normally just a few changes will be needed); a cross shows total incompatibility. Reviews do not distinguish between Basic I and II.

Computer System

Filing System

Master (Basic IV)



ADFS



Compact (Basic VI)



DFS



Model B (Basic II)



Cassette



Model B (Basic I)



Tube Compatibility

Electron



Tube



Editor's Jottings

MICRO USER SHOW AND RISC USER

The Micro User Show in November is always a very popular one. At the time of writing, this show is still two weeks away, but I hope we will have had the opportunity of talking with many BEEBUG members at the show. One exciting development will have been our separate stand and promotion for RISC User, our new magazine for Archimedes users. This has already proved to be very successful, and the second issue will be out very shortly. Remember that there are very advantageous rates for BEEBUG members who also wish to subscribe to RISC User, while it is also possible to convert a BEEBUG subscription to one for RISC User. Full details on the cost of RISC User to BEEBUG members will be found in the supplement pages.

MAGAZINE CASSETTE/DISC

The magazine cassette or disc really is the painless way to benefit from all the programs we publish each month in BEEBUG magazine. No long hours laboriously typing away, and no checking to find the typing mistakes afterwards. Because of increasing volumes, we have now been able to reduce prices for 3.5" discs to the same level as 5.25" discs. Furthermore, we are also offering up to two free discs (or cassettes) on new 3.5" and 5.25" disc or cassette subscriptions (or renewals). This is the cost effective and efficient way to ensure you can make the most of BEEBUG. Full details of this special offer are contained on the insert with this month's issue.

BEEBUG MAGAZINE

We have been working very hard to catch up on previous delays to get this issue of BEEBUG to you a little earlier than of late. We shall be continuing our endeavours with the December issue, which we hope will be with you by mid-December. This is a busy time of year for everyone at BEEBUG, so do make sure that all necessary information is included with any enquiry or order, and that you don't forget to quote your membership number.

Mike Williams

News News News News

MORE FOR ARCHIMEDES

Acorn's Archimedes was selected as Home/Small Business Micro of the Year in the 1987 British Microcomputing Awards during the 1987 PCW Show.

Latest news from Acorn is that the promised free word processor, called Arcwrite, is expected to be released late November, and a copy will be sent out to all registered Archimedes users. Registered users will also be provided with Arthur Release 1 to replace the provisional versions 0.2 and 0.3 supplied with early machines.

Acorn will also be releasing soon a major word processor for Archimedes called First Word Plus, a powerful software package which is being rewritten for Archimedes. Price and availability have not yet been announced. Acorn may be contacted on Cambridge (0223) 214411.

ZIGGY RANSACK DESPATCH RIDER

New games releases from Audiogenic for the Christmas market are Ziggy (a three-dimensional game of strategy), Ransack (a first shoot-em-up game from arcade adventure king Peter Scott), and Despatch Rider (an excellent biker simulation). Ransack costs £9.95 on cassette (BBC/Electron) and £11.95 on disc (Master compatible), while the other two games retail at £8.95 for cassette and £10.95 on disc for the same machines. Contact Audiogenic on (0734) 303663.

BUSY BEEBUG

BEEBUG has not been idle in recent months and follows its well received 'C' compiler with a stand-alone generator for 'C', allowing compiled 'C' programs to run on any machine. A new version of the Command ROM has been announced offering full Hayes modem compatibility, and BEEBUG's first hardware product, a BABT approved internal modem for the Master 128. Last, but not least, the very popular machine code monitor Exmon II is now available in a

fully Master compatible version. Full price details are included in the mail order catalogue, or contact BEEBUG on St Albans (0727) 40303 for further details.

BIG BEN CLUB WELCOMES BEEBUG

The Dutch Acorn users group, known as the Big Ben Club, held their 5th Annual Show on the 10th October 1987. For the first time BEEBUG braved the North Sea to set up its stall alongside five other English companies, and several Dutch dealers. We were very pleased to have the opportunity to talk with so many Dutch Beeb enthusiasts, including a good many BEEBUG members in the Netherlands. The show was very successful, the organisation very efficient and the welcome we received very friendly. We hope to see more Big Ben Club members joining BEEBUG in the future, and we are looking forward to a return visit to Holland next year.

SOLID PERFORMANCE FROM SILICON VISION

Silicon Vision's Real Time Graphics System received an excellent review from BEEBUG when it was released (BEEBUG Vol.5 No.10), and the same

consists of a 32K graphics ROM, six discs and a 150 page manual. This system also includes Solids Realtime Graphics language providing 52 star commands for flicker-free 3D animation. We hope to review this product shortly. Silicon Vision can be contacted on 01-422 2274.

SHADES OF MICRONET

Shades, Micronet's successful multi-user game, which currently clocks up over 3000 hours of playing time every week, is now available to Micronet subscribers in scrolling 80-column mode as well as the original 40-column viewdata mode. This enhancement will provide players with faster game-play and more on-screen information. The new scrolling version will cost subscribers 2p a minute to play while charges for the viewdata version remain unchanged at 1.62p a minute. For more information contact Micronet on 01-278 3143.

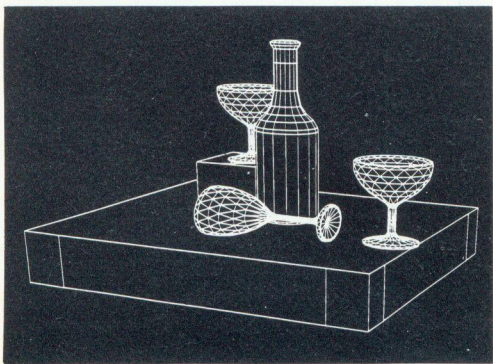
CHOCKS AWAY

Don the old fleece-lined jacket and zoom up into the wide blue yonder. Mirrorsoft's classic Spitfire 40 aircraft simulation is now available for the BBC/Electron at £9.95 on cassette, £12.95 on 5.25" disc and £14.95 on 3.5" disc for the Compact. Mirrorsoft are on 01-377 4837.

PLAY IT AGAIN SAM

This somewhat unoriginal title has been chosen by Superior Software to relaunch four previously successful games for the BBC micro. These include Citadel, a fascinating arcade adventure, Thrust, a game with a space theme, the classic Stryker's Run and another arcade adventure, Ravenskull. All four games are on the same cassette (£9.95 for BBC and Electron versions) or disc (£11.95 on 5.25" disc for the BBC and Electron, £14.95 on 3.5" disc for the Compact and Electron). Superior are on (0532) 459453, and 'Play it again Sam' should be in your favourite computer shop now.

B



company has now released its new Real Time Solids Modeller. At £88.95 this provides 3D solid or wire-frame objects with full colour hidden surface removal, and support for colour hard copy. The package

MODE 7 MENU GENERATOR

Many programs are menu-driven for ease of use, but creating your own professional looking menus is not so easy. Now, however, Graham Crow has done all the hard work for you with this excellent utility.

INTRODUCTION

With so many programs being menu-driven, it is important that menus are attractively presented and easy to use. MENGEN is a disc utility program which enables you to create mode 7 menus for use in your own programs. What's more, the program actually generates the Basic code for you, and saves it as a spooled file ready for appending to your own program. This code is in the form of a function (FNmenu), and includes the lines which call FNmenu. You don't have to write a single line of code!

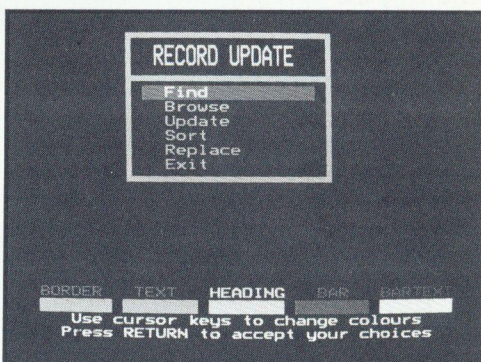
Up to 9 menus may be generated for a single application, each with up to 15 items. Each menu is displayed within a border, with the heading in double-height text. The up and down cursor keys are then used to move a cursor bar between the items. Selection is made either by pressing Return when the cursor bar is on the required item, or simply by pressing its initial letter. Different colours may be selected for the border, the heading, the text, the cursor bar, and the bar text. MENGEN is suitable for all models of the BBC micro which support mode 7.

HOW TO USE MENGEN

As always, make sure you have saved a copy of the menu generator program before attempting to try it out. Running the program MENGEN will display the title page and show that none of the nine available menus has yet been created. Follow the on-screen prompts, entering the menu number, the number of items per menu, the menu heading, and the description of each item. The first letter of each item should be different if you want to take advantage of initial letter selection.

The screen then clears to display the menu exactly as it will finally appear, but using the default colours of cyan (border and text), yellow (heading), green (cursor bar) and white (bar text). You are asked "OK to continue?". If you want to re-create the text, press "N" to go back to the first screen. Otherwise, press "Y" to display, beneath the menu, a colour palette. Use the right and left cursor keys to highlight the part of the menu you wish to colour, and the up and down cursor keys to cycle through the 7 available colours. As you do so, the menu changes to show the effect. When you are satisfied, press Return to confirm your choices.

You are then asked whether you wish to create another menu. Answering "Y" takes you back to the first screen, and shows which of the 9 menus have been created so far. You can either select a new number (in any order) or select one that has already been created (if you want to change it).



Creating a new menu

If you do not wish to create any more menus, the screen clears and you are prompted for details of the SPOOL file to contain the code for the menus you have designed. Enter the drive number, directory and filename. The default drive is 0 and the default directory is "\$". Pressing Return accepts the defaults, but you can change them if you wish.

The code to generate the menus you have designed is then spooled out to the file specified. This file may subsequently be

*EXECed to append the code to your development program. You are then given the option of installing and running FNmenu forthwith, to check its operation. If you answer "N" the program ends, but if you reply "Y", MENGEn is replaced in memory by the new code. The screen clears to present the first menu in full working order. When you choose any item, the next menu is displayed, and so on. The program in memory is a Basic program and you can of course RUN, LIST, RENUMBER, SAVE, etc. as you wish.

Pressing Escape at any time gives you the option of leaving MENGEn altogether, or returning to the main input screen. You can use MENGEn any number of times to create new menus, or change existing ones. All you need to do is keep a note of the numbers of the menus which have already been created.

HOW TO USE THE SPOOLED MENU FILE

The spooled file is in two parts, the calling lines (one for each menu) starting at line number 20000, and the function FNmenu itself starting at line 30000. To append the code to your own Basic program, type *EXEC filename, where filename is the one you chose for the *SPOOL file. You will then need to renumber the calling line(s) to fit into your program at the appropriate point. They may also be copied if any menu has to be called more than once.

When called, the function FNmenu returns the number of the selected item in the specified menu. So your calling line might be, for example:

```
670 choice=FNmenu(1,"MAIN MENU",9,"Bb
HhFfVvPpLlEeQq",15,7,3,150,134,131,130,135)
680 IF choice=1 THEN PROCbody ELSE IF ch
oice=2 PROChead
690 etc.....
```

Once you have done this, you can delete the lines between 20000 and 20010, and then renumber your program.

If you use MENGEn in future to create additional menus, or alter existing ones, you

will have to renumber not only the calling lines, but also transfer the newly created RESTORE (between 30020 and 30029) and DATA (between 30030 and 30039) lines to your existing copy of FNmenu. You can then delete lines 20000 onwards, and renumber as required.

PROGRAM NOTES - FNmenu

FNmenu takes 12 parameters:

n%	- the menu number;
h\$	- the text for the heading;
i%	- the number of items
i\$	- a string of the initial letters of the items (upper and lower case);
w%	- the width of the menu;
l%	- the left tab position;
t%	- the top tab position;
c1%	- the colours for border, text,
to c5%	heading, cursor bar, and bar text.

The function returns the number of the item selected.

It is possible to use the same code for up to 9 different menus by means of conditional RESTORE and DATA statements. These vary according to the value of n%, the menu number. The RESTORE lines are in intervals of 1, starting at line 30020, and the DATA lines are also in intervals of 1, but starting at line 30030. The DATA statements give the list of items (up to 15 per menu) and the READ is controlled by the parameter i% - the number of items.

The code is fairly straightforward, being concerned with drawing the correct menu in the correct position, and then entering a REPEAT-UNTIL loop which waits for a valid key depression. INSTR is used to check whether a valid initial letter has been pressed. If the up or down cursor keys (139 or 138) are pressed, the position of the cursor bar is adjusted. If Return is pressed or the value of INSTR is greater than 1, then the loop terminates, and the function ends by returning the number of the selected item in p%.


```

10 REM Program Menu7
20 REM Version B1.0
30 REM Author Graham Crow
40 REM BEEBUG November 1987
50 REM Program subject to copyright
60 :
100 MODE 7:ON ERROR PROCerror
110 PROCinit
120 REPEAT:REPEAT:PROCgetinfo:PROCcalculate:CLS:PROCbox:PROCtext
130 UNTIL FYes(10,23,"OK to continue"
)
140 PROCcolours:PROCbuild:PRINTTAB(0,2
3)SPC(79);
150 UNTIL menus=maxmenus OR NOT FYes(
6,24,"Create another menu")
160 PROCspool
170 IF FYes(1,20,"Do you want to inst
all FNmenu")THEN PROCinstall
180 PRINT':END
190 :
1000 DEFPROCinit
1010 maxitems=15:maxcolours=5:maxmenus=
9:maxwidth=27:menus=0
1020 DIM item$(maxitems),c$(maxcolours)
1030 DIM call$(maxmenus),restore$(maxme
nus),data$(maxmenus),created$(maxmenus)
1040 FOR J%=1 TO maxmenus:created$(J%)=
"NO":NEXT
1050 ENDPROC
1060 :
1070 DEFPROCgetinfo
1080 VDU12,28,29,23,38,6:PRINT "MENUS"
"CREATED"
1090 FOR J%=1 TO maxmenus:PRINT;J%;"...
"created$(J%):NEXT:VDU26
1100 FOR J%=1 TO 2:VDU31,7,J%,131,141:P
RINT"MODE 7 MENU GENERATOR":NEXT
1110 PRINTTAB(1,4)"Please enter the fol
lowing details:"TAB(1,5)STRING$(35,"=")
1120 REPEAT:n%=VAL(FNin("Menu number",1
,6,1,"")):UNTIL n%>0 AND n%<=maxmenus
1130 VDU28,29,23,38,6,12,26
1140 REPEAT:i%=VAL(FNin("Items (max "+S
TR$(maxitems+" ),19,6,2,""))
1150 UNTIL i%>0 AND i%<=maxitems
1160 REPEAT:h$=FNin("Heading",1,8,maxwi
dth,""):UNTIL h$<>" "
1170 FOR J%=1 TO i%
1180 REPEAT:item$(J%)=FNin("Item "+STR$(
J%)+",1,J%+8,maxwidth,"")
1190 asc=ASC(LEFT$(item$(J%),1))
1200 UNTIL LEN(item$(J%))>0 AND ((asc>6
4 AND asc<91) OR (asc>96 AND asc<123))
1210 NEXT:ENDPROC
1220 :
1230 DEFPROCcalculate
1240 w%=LEN(h$) : REM initial setting f

```

```

or width
1250 i$="" : REM initial setting initia
l letters
1260 FOR J%=1 TO i%
1270 IF LEN(item$(J%))>w% w%=LEN(item$(
J%))
1280 REM Allow upper case and lower cas
e
1290 uc%=ASC(LEFT$(item$(J%),1))AND &DF
:lc%=uc%+32
1300 i$=i$+CHR$uc%+CHR$lc%
1310 NEXT
1320 l%=19-(w% DIV 2)-5 : REM left marg
in
1330 IF i%>13 t%=0 ELSE t%=3 : REM top
margin
1340 c1%=150:c2%=134:c3%=131:c4%=130:c5
%=135 : REM default colours
1350 ENDPROC
1360 :
1370 DEFPROCbox
1380 VDU23,1,0;0;0;0;
1390 n1$=CHR$c1%+CHR$232+STRING$(w%+5,C
HR$172)+CHR$180
1400 n2$=CHR$c1%+CHR$141+CHR$234+CHR$c3
%+STRING$(w%+3,CHR$32)+CHR$c1%+CHR$181
1410 n3$=CHR$c1%+CHR$234+STRING$(w%+5,C
HR$172)+CHR$181
1420 n4$=CHR$c1%+CHR$234+CHR$c4%+CHR$15
6+CHR$c2%+STRING$(w%,CHR$32)+CHR$c1%+CHR
$156+CHR$181
1430 n5$=CHR$c1%+CHR$170+STRING$(w%+5,C
HR$172)+CHR$37
1440 PRINTTAB(1%,t%)n1$TAB(1%-1,t%+1)n2
$TAB(1%-1,t%+2)n2$TAB(1%,t%+3)n3$
1450 FOR J%=t%+4 TO t%+3+i%:PRINTTAB(1
%,J%)n4$:NEXT:PRINTTAB(1%,J%)n5$
1460 ENDPROC
1470 :
1480 DEFPROCtext
1490 PRINTTAB(18-(LEN(h$)DIV 2),t%+1)h$
TAB(18-(LEN(h$)DIV 2),t%+2)h$
1500 FOR J%=1 TO i%:PRINTTAB(1%+5,J%+t%
+3)item$(J%):NEXT
1510 line%=1:PRINTTAB(1%+3,line%+t%+3)C
HR$157CHR$c5%
1520 ENDPROC
1530 :
1540 DEFPROCcolours
1550 c%(1)=150:c%(2)=150:c%(3)=147:c%(4
)=146:c%(5)=151
1560 REM Default colours for (graphic)
palette
1570 REM border=cyan:text=cyan:heading=
yellow:bar=green:bartext=white
1580 RESTORE 1840:FOR J%=0 TO 35 STEP 8
:READ N$,N%
1590 PRINTTAB(J%,21)CHR$130 TAB(J%+1,21

```



```

)N$
1600 PRINTTAB(J%,22)CHR$N% TAB(J%+1,22)
STRING$(7,CHR$255)
1610 NEXT
1620 PRINTTAB(4,23)"Use cursor keys to
change colours"
1630 PRINTTAB(3,24)"Press RETURN to acc
ept your choices";
1640 VDU23,1,0;0;0;0;:FX4,1
1650 pos%=0:PRINTTAB(pos%,21)CHR$135
1660 REPEAT
1670 REPEAT G%=GET:UNTIL G%=13 OR (G%>=
136 AND G%<=139)
1680 IF G%=13 THEN 1800
1690 IF G%=138 OR G%=139 THEN 1730
1700 IF G%=137 PRINTTAB(pos%,21)CHR$130
:pos%=pos%+8:IF pos%>39 pos%=0
1710 IF G%=136 PRINTTAB(pos%,21)CHR$130
:pos%=pos%-8:IF pos%<0 pos%=32
1720 PRINTTAB(pos%,21)CHR$135:GOTO 1800
1730 p%=(pos%+8)DIV8
1740 IF G%=138 c%(p%)=c%(p%)+1:IF c%(p%
)>151 c%(p%)=145
1750 IF G%=139 c%(p%)=c%(p%)-1:IF c%(p%
)<145 c%(p%)=151
1760 PRINTTAB(pos%,22)CHR$c%(p%)
1770 c1%=c%(1):c2%=c%(2)-16:c3%=c%(3)-1
6:c4%=c%(4)-16:c5%=c%(5)-16
1780 REM Change current colours to alph
a for use in PROCbox
1790 PROCbox:PROCtext
1800 UNTIL G%=13
1810 VDU23,1,1;0;0;0;:FX4
1820 FOR J%=2 TO 5:c%(J%)=c%(J%)-16:NEX
T: REM change to alpha
1830 :
1840 DATA BORDER,150," TEXT",150,HEADIN
G,147," BAR",146,BARTEXT,151
1850 ENDPROC
1860 :
1870 DEFPROCbuild
1880 REM Generate calling lines, RESTOR
E lines, and DATA lines
1890 call$(n%)=""
1900 call$(n%)=STR$n%+", "+"++++h$+"+++++
", "+STR$i%+", "+"++++i$+"+++++", "+STR$w%+
", "+STR$l%+", "+STR$t%
1910 call$(n%)=call$(n%)+", "+STR$c1%+",
"+STR$c2%+", "+STR$c3%+", "+STR$c4%+", "+ST
R$c5%
1920 line%=30030:REM first DATA line
1930 restore$(n%)="" :restore$(n%)="IF n
%="+STR$n%+" RESTORE "+STR$(line%+n%-1)
1940 data$(n%)="" :FOR J%=1 TO i%:data$(
n%)=data$(n%)+item$(J%)+", ":NEXT
1950 data$(n%)=LEFT$(data$(n%), LEN(data
$(n%))-1):REM remove last comma
1960 IF created$(n%)="NO" created$(n%)=

```

```

"YES":menus=menus+1
1970 ENDPROC
1980 :
1990 DEFPROCspool
2000 CLS:IF menus=maxmenus PRINTTAB(1,1
)"All menus created"
2010 PRINTTAB(1,3)"PREPARE TO CREATE A
SPOOL FILE"
2020 PROCgetdrive(1,5,"0"):PROCgetdirec
tory(1,6,"$"):PROCgetfilename(1,7,"")
2030 PROCcheckdisk(1,9):IF NOT ok THEN
2000
2040 CLS:PRINTTAB(1,3)"Please wait...SP
OOling "directory$"."filename$" ";
2050 REM Disable VDU
2060 *FX3,6
2070 REM Disable ESCAPE
2080 *FX229,1
2090 $&700="SPOOL "+fsp$:X%=0:Y%=7:CALL
&FFF7
2100 L%=20000:I%=1
2110 FOR J%=1 TO maxmenus
2120 IF call$(J%)<>"PRINT;L%;" C%=FNme
nu(" "+call$(J%)+")"
2130 L%=L%+I%:NEXT
2140 PRINT"20010 END":PRINT"20020:"
2150 PRINT"30000 DEFFNmenu(n%,h$,i$,i$,
w%,l%,t%,c1%,c2%,c3%,c4%,c5%)"
2160 PRINT"30010 LOCAL N$,G%,J%,I%"
2170 L%=30020:I%=1
2180 FOR J%=1 TO maxmenus
2190 IF restore$(J%)<>"PRINT;L%;" " +re
store$(J%)
2200 L%=L%+I%:NEXT
2210 L%=30030:I%=1
2220 FOR J%=1 TO maxmenus
2230 IF data$(J%)<>"PRINT;L%;" DATA "+
data$(J%)
2240 L%=L%+I%:NEXT
2250 PRINT"30040 VDU12,23,1;0;0;0;0;"
2260 PRINT"30050 PRINTTAB(1%,t%)CHR$c1%
+CHR$232+STRING$(w%+5,CHR$172)+CHR$180"
2270 PRINT"30060 FOR J%=t%+1 TO t%+2:PR
INTTAB(1%-1,J%)CHR$c1%+CHR$141+CHR$234+C
HR$c3%+STRING$(w%+3,CHR$32)+CHR$c1%+CHR$
181:NEXT"
2280 PRINT"30070 PRINTTAB(1%,t%+3)CHR$c
1%+CHR$234+STRING$(w%+5,CHR$172)+CHR$181
"
2290 PRINT"30080 FOR J%=t%+4 TO t%+3+i%
:PRINTTAB(1%,J%)CHR$c1%+CHR$234+CHR$c4%+
CHR$156+CHR$c2%+STRING$(w%,CHR$32)+CHR$c
1%+CHR$156+CHR$181:NEXT"
2300 PRINT"30090 PRINTTAB(1%,J%)CHR$c1%
+CHR$170+STRING$(w%+5,CHR$172)+CHR$37"
2310 PRINT"30100 FOR J%=t%+1 TO t%+2:PR
INTTAB(1%+2+(w%+6-LEN(h$))DIV2,J%)h$:NEX
T"

```



```

2320 PRINT"30110 FOR J%=1 TO i%:READ N$
:PRINTTAB (1%+5,J%+t%+3)N$:NEXT"
2330 PRINT"30120 p%=1:PRINTTAB (1%+3,p%+
t%+3)CHR$157CHR$c5%:*FX4,1"
2340 PRINT"30130 REPEAT:REPEAT:G%=GET:I
%=INSTR (i$,CHR$G%) "
2350 PRINT"30140 UNTIL G%=13 OR G%=138
OR G%=139 OR I%>0"
2360 PRINT"30150 IF G%=13 THEN 30210"
2370 PRINT"30160 PRINTTAB (1%+3,p%+t%+3)
CHR$156CHR$c2%"
2380 PRINT"30170 IF I%>0 p%=(I%+1)DIV2:
GOTO 30200"
2390 PRINT"30180 IF G%=138 p%=p%+1:IF p
%>i% p%=1"
2400 PRINT"30190 IF G%=139 p%=p%-1:IF p
%=0 p%=i%"
2410 PRINT"30200 PRINTTAB (1%+3,p%+t%+3)
CHR$157CHR$c5%"
2420 PRINT"30210 UNTIL G%=13 OR I%>0:VD
U23,1,1;0;0;0;:*FX4"
2430 PRINT"30220 =p%"
2440 *SPOOL
2450 REM Enable VDU
2460 *FX3
2470 REM Enable ESCAPE
2480 *FX229
2490 ENDPROC
2500 :
2510 DEFPROCinstall
2520 REM Load Key 0 with the commands t
o NEW, EXEC fsp$, and RUN
2530 REM and use *FX138 to insert the c
ontents of Key 0 in the keyboard.
2540 REM *FX3 & *FX229 are used to disa
ble/enable the VDU and the ESCAPE key.
2550 $&900=fsp$: REM the NEW command ca
uses loss of fsp$!
2560 *KEY 0 NEW|M $&700="EXEC "+$&900:X
%=0:Y%=7:CALL &FFF7|M *FX3|M *FX229|M RU
N|M
2570 PRINTTAB (1,22) "Please wait...insta
lling FNmenu ";
2580 *FX3,6
2590 *FX229,1
2600 *FX138,0,128
2610 ENDPROC
2620 :
2630 DEFNyes(x$,y$,message$)
2640 VDU23,1,1;0;0;0;
2650 PRINTTAB (x$,y$)message$+"? (Y/N):
";
2660 REPEAT G$=GET$:PRINTG$:I%=INSTR ("Y
yNn",G$)
2670 UNTIL I%>0:IF I%<3 THEN=TRUE ELSE=
FALSE
2680 :
2690 DEFPROCerror:VDU7

```

```

2700 IF ERR<>17 THEN 2720
2710 IF NOT FNyes(1,24,"Do you want to
quit MENGAN") THEN 120
2720 VDU12,23,1,1;0;0;0;:*FX4
2730 *FX229
2740 IF ERR<>17 REPORT:PRINT" at line "
;ERL
2750 END
2760 :
2770 DEFPROCgetdrive(x,y,default$)
2780 A%=0:Y%=0:X%=&70:F%=(USR &FFDA) AN
D &FF0000 DIV &10000
2790 REPEAT:drive$=FNin("Drive.....",x
,y,1,default$)
2800 IF drive$="" drive$=default$
2810 IF F%=8 THEN UNTIL drive$="0" OR d
rive$="1"
2820 IF F%=4 THEN UNTIL drive$="0" OR d
rive$="1" OR drive$="2" OR drive$="3"
2830 IF F%=8 THEN $&700="MOUNT "+drive$
:X%=0:Y%=7:CALL &FFF7
2840 ENDPROC
2850 :
2860 DEFPROCgetdirectory(x,y,default$)
2870 REPEAT:directory$=FNin("Directory.
.",x,y,1,default$)
2880 IF directory$="" directory$=defaul
t$
2890 UNTIL directory$<>" "
2900 ENDPROC
2910 :
2920 DEFPROCgetfilename(x,y,default$)
2930 REPEAT:filename$=FNin("Filename...
",x,y,7,default$)
2940 IF filename$="" filename$=default$
2950 UNTIL filename$<>" "
2960 ENDPROC
2970 :
2980 DEFPROCcheckdisk(x,y)
2990 fsp$=":"+drive$+"."+directory$+"."
+filename$:ok=TRUE
3000 K%=OPENUP (fsp$):IF K%=0 ENDPROC
3010 CLOSE #K%:VDU7:PRINTTAB (x,y)fsp$"
exists. Overwrite? (Y/N): ";
3020 REPEAT G$=GET$:I%=INSTR ("YyNn",G$)
:UNTIL I%>0:PRINT G$:IF I%>2 ok=FALSE
3030 ENDPROC
3040 :
3050 DEFNin(s$,x,y,m,d$)
3060 code=&70:buffer=&C0:P%=code
3070 [OPT0:CLC:LDA#0:TAX:STX&8B:LDX#&C:
STX&8C:LDX#32:STX&8E:LDX#127:STX&8F
3080 TAY:LDX#&8B:JSR&FFF1:RTS:]
3090 PRINTTAB (x,y) s$:x=x+LEN (s$)+1
3100 PRINTTAB (x,y) "<"d$SPC (m-LEND$) ">"T
AB (x+1,y);
3110 ?&8D=m:CALLcode:=$buffer

```

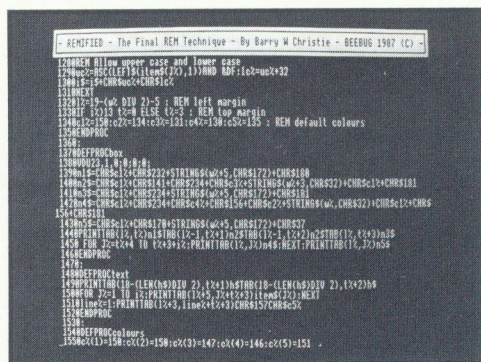
B

With Barry Christie's utility you can append to your program a few special lines of code. These can be used to generate almost any visual effect you choose, whenever your program is listed. Use Barry Christie's technique to customise your own programs.

If you type LIST with a program in memory, the program will be listed out one line at a time under the control of Basic. But by adding a series of VDU codes into the first REM lines of a program you can effectively take control of what happens when the program is listed. As a very simple example, one way to protect programs from prying eyes is to include VDU codes which turn off the VDU drivers when the first line is encountered, so that the result of trying to list the program is a blank screen.

But you can do more imaginative things. The accompanying screen shot shows one example. The first REM lines of the program contain a series of VDU codes which set up mode, windows and colours, and generate the required text heading. You can even create double height or double width titles in this way. But first of all, you need some software to generate the codes, since you can't normally just insert VDU codes of your choice directly into a program line. In case there is some confusion, I should say that the codes which we are using here are incorporated *directly* in the listing. In other words VDU14 would not appear as the five character string **VDU14**, but simply as the single character code 14, etc.

The program provided here, which completely automates the process, works broadly as follows. The user simply replaces the section of the program between the asterisks with a set of Basic commands to set up the effect which he wishes to create - more on this later. When he is happy that the desired effect is produced, he runs the program. This generates a file (called Remprog), containing the required sequence of raw VDU codes set up in a series of REM lines. He then presses function key f0. This loads and renumbers the REM lines. The user then supplies the filename of the target program, and this is automatically loaded in and appended to the REM lines.



Example of customised listing

To get the utility up and running, type in the listing, and save it away. Strictly speaking you do not need lines 190 to 270, though they form a useful testbed. If you run the program, containing all the lines listed, you should see the heading illustrated in the screen shot, and should hear the drive whirr as the REM file is saved away. Now press f0, and supply the name of the program which is to be appended to the REM lines. This should contain no line numbers below 10 in order to avoid overwriting the VDU code REMs. If you now try to LIST your program you should see the same effect as that produced earlier.

CUSTOMISED EFFECTS

Once it is all working, you can turn your attention to creating your own customised effects. In fact you can create any effect which can be broken down into VDU statements of one kind or another - though you do not need to bother with the VDU codes themselves, since they are automatically handled by the program. This means that Basic keywords such as PLOT, MOVE, COLOUR, GCOL, MODE and so on may all be used.

If you take a look at the commands used in lines 190 to 270 you will get some idea of the variety which is possible. The first line contains a mode change. Any mode change must be immediately followed by the HIMEM assignment appearing in line 190. This is used to allocate space for the utility. Lines 210 to 230 create the boxing effect. Then in lines 240 and 250 the title is printed, and so on. Finally a text window is set (line 260) to prevent the heading being scrolled out when the program begins to list, and in line 270 the background and foreground colours are reset to normal.

NOTES

There is only one major limitation to the program as presented. Basic's LIST routine gets confused if it comes across extra VDU code 13s in a listing. It assumes that these all indicate the end of program lines. The utility therefore removes all extra occurrences of VDU13 and replaces them with VDU10. This has two consequences. Firstly, all PRINT statements used *must* be accompanied by a TAB statement - even if it is just TAB(0).

Secondly, PLOT codes, TAB codes and the like will on occasion be adjusted by the program. For example, if the low byte of a plot code is 13 it will be altered to 10. This will have a negligible effect in the majority of cases. The most noticeable effect will probably be that TAB(13,n) will get altered to TAB(10,n). However, you can adjust this yourself by printing 3 spaces before your line of text, and immediately after the TAB.

```
10 REM Program Fancy lister
20 REM Version B 1.0F
30 REM Author Barry Christie
40 REM BEEBUG November 1987
50 REM Program Subject to Copyright
60 :
100 MODE 7
110 PROCgeneratesource
120 ON ERROR CALL resetvector:REPORT:P
PRINT " at line ";ERL:END
130 :
140 REM Replace the lines between the
asterisks
150 REM with your own header generator
160 :
170 REM*****
180 :
190 MODE 0:HIMEM=TOP+512
200 COLOUR 135:COLOUR 0
210 GCOL 0,129:VDU 24, 0;932;1279;1023
;:CLG
220 GCOL 0,128:VDU 24, 8;944;1271;1011
;:CLG
230 GCOL 0,129:VDU 24,12;948;1267;1007
;:CLG
240 PRINT TAB( 1, 1)"- REMIFIED - The
Final REM Technique";
250 PRINT " - By Barry W Christie - BE
EBUG 1987 (C) -"
260 VDU 28,0,31,79,3
270 COLOUR 128:COLOUR 7
280 :
290 REM*****
300 :
1000 PROCgenerateobject
1010 *KEY0 MODE7|MLOAD "Remprog"|M RENU
MBER1,1|MVDU21|MVDU6:INPUT"Target progra
m name ? "file$:A%=TOP-2:OS.("LOAD "+fil
e$+" "+STR$~A%):OS.("FX138,0,129")|M
1020 *KEY1 OLD|M
1030 END
1040 :
1050 DEF PROCgeneratesource
1060 source=TOP+512
1070 FOR pass%=0 TO 2 STEP 2
1080 P%=&900
1090 [ OPT pass%
1100 .redirvector
1110 LDA &20E:STA oswrch+&01:LDA #grabs
A MOD 256:STA &20E
1120 LDA &20F:STA oswrch+&02:LDA #grabs
A DIV 256:STA &20F
1130 RTS
1140 :
1150 .resetvector
1160 LDA oswrch+&01:STA &20E:LDA oswrch
+&02:STA &20F:RTS
```



```

1170 :
1180 .grabsA
1190 CMP #&0D:BNE noreturn
1200 LDA #&0A:JSR storeA
1210 LDA #&0D:JMP oswrch
1220 .noreturn:JSR storeA
1230 .oswrch:JMP &BEEB
1240 :
1250 .storeA:STA source
1260 INC storeA+&01:BNE noinchi
1270 INC storeA+&02:.noinchi:RTS
1280 ]:NEXT pass%
1290 CALL redirvector
1300 ENDPROC
1310 :
1320 DEF PROCgenerateobject
1330 CALL resetvector
1340 length=(storeA+&01+256*storeA+&02)
-source
1350 object=source+length+10:save=objec
t-1:?save=&0D
1360 REPEAT
1370 PROClinestart
1380 REPEAT
1390 byte=offset?source
1400 IF byte<&20 THEN PROCpokeparams EL
SE PROCpokebyte
1410 UNTIL length<=0 OR offset>=&F0
1420 PROClineend
1430 UNTIL length<=0
1440 PROCprogramend
1450 ENDPROC
1460 :
1470 DEF PROClinestart
1480 object!&00=&F4690000:object!&04=&0
622:object=object+6:offset=0

```

```

1490 ENDPROC
1500 :
1510 DEF PROClineend
1520 object!offset=&00000D15:?(object-&
04)=offset+8
1530 object=object+offset+2 :source=sou
rce+offset
1540 ENDPROC
1550 :
1560 DEF PROCpokeparams
1570 parameters=VAL(MID$("0100000000000
0000125001985004402",byte+1,1))
1580 FOR I%=0 TO parameters:PROCpokebyt
e:NEXT
1590 ENDPROC
1600 :
1610 DEF PROCpokebyte
1620 object?offset=source?offset:offset
=offset+1:length=length-1
1630 ENDPROC
1640 :
1650 DEF PROCprogramend
1660 REM Change object!&04=&00FF0D06 to
object!&04=&00FF0D15 to prevent any of
the user's program from being listed.
1670 object!&00=&F4060000:object!&04=&0
0FF0D06:object=object+7
1680 OSCLI("SAVE Remprog "+STR$~save+"
"+STR$~object)
1690 PRINT"REM header now generated on
disc"
1700 PRINT"Press f0 to load this from
disc"
1710 PRINT"and append target program"
1720 ENDPROC

```

B

POINTS ARISING...POINTS ARISING...POINTS ARISING

CATALOGUING DISCS WITH FILER (BEEBUG Vol.6 No.5)

This program was incorrectly shown as working on an ADFS system. This is not possible as the program is designed, and we apologise for any inconvenience that may have occurred. We shall, however, be considering the possibility of an ADFS version of this program.

PERSONAL ADDRESS BOOK (BEEBUG Vol.6 No.3)

Despite our best intentions to ensure that this program works with both Basic I and Basic II, some incompatibility remained. For Basic I (only), make the following changes to the published program:

Delete lines 1530 and 1540 completely (magazine and cassette/disc)

Replace OPENUP with OPENIN at line 1400 (magazine only)

Note: OPENUP in Basic II has the same token as OPENIN in Basic I. OPENIN in Basic II generates an unrecognised token if run subsequently under Basic I.

B

EPSON CHARACTER DEFINER

Customise your printer's character set with this utility by Cliff Blake, which will do much to help you make the most of your printer, by providing an easy way of defining all those extra characters (like the pound sign), which are so essential.

Several good programs have been published for defining screen characters using the VDU23 statement. Few have been written to define characters on a printer. Special fonts are usually obtained by bit-image printing which gives a more detailed design, but there are occasions when a simple user-defined non-standard character is better for faster printing. One particular application of this is to provide hard copy output of the extended character set of the Master and Compact.

The DEFPRIN program listed here, simulates the printer matrix pattern exactly, and makes character definition as automatic as possible. As written, it provides for final designs to be recorded on disc, but for tape use it is probably quicker to ignore this feature and take the definition values from the screen or sample printer output.

USING THE PROGRAM

Type the program in carefully, paying particular attention to spaces in instructions, and save away to disc or tape before you start to use it. When run, the program displays an outline character grid on the screen with all the additional information you will need, including a continually updated character definition.

The grid for printer characters has 12 columns. A dot occupies 2 of them, and its position is specified in the column for its left half. The last column therefore is never specified, leaving 11 values for positioning the dots. Use the cursor keys to move around the grid, with Copy

inserting a dot, and Delete removing a dot in any position.

When an attempt is made to overlap dots, the first one is automatically deleted. It is necessary to press the left or right cursor key twice in order to place a separate adjacent dot. Horizontally, it is possible to fit in a maximum of 6 dots. All can be used for a graphical symbol, but for text characters the practice is to use the first 5 only, with the last 2 columns being left as the space between letters.

Having arrived at a provisional design for your character, pressing key P will send this to the printer in a variety of styles (see illustration) for checking. If this is unsatisfactory, further changes to the design may be made.

IMPORTANT NOTE

Your printer MUST be configured for user-defined characters or this program will not work. This usually involves setting a DIP switch on your printer, and you should consult your printer's manual for more information. On an Epson FX80 set switch SW1-4 to OFF, likewise switch SW2-3 on a Kaga KP810 should also be set to OFF. Switch your printer off before changing any switch settings.

Pressing key N at any time simply clears the current character definition from the screen, ready to start on a new character.

MORE ON CHARACTER DEFINITIONS

The program defaults to the ascender condition, with the upper 8 pins of 9 in the printer head being used. Placing a dot in the bottom 9th row, switches to the descender condition in which pins 2 to 9 are active instead, and any dots already in the top row are deleted. Similarly placing a dot in the top row deletes any in the 9th row. The top 2 rows are used for an ascender, the next 5 for the main body of a character, and the 8th row is for underlining. A descender occupies rows 8 and 9.

The 12 columns are numbered 0 to 11 for defining the proportional width. Normally the START value will be the first column used, and the END value will be the last column used + 2

for spacing giving:

$$\text{WIDTH} = \text{END} - \text{START} + 1.$$

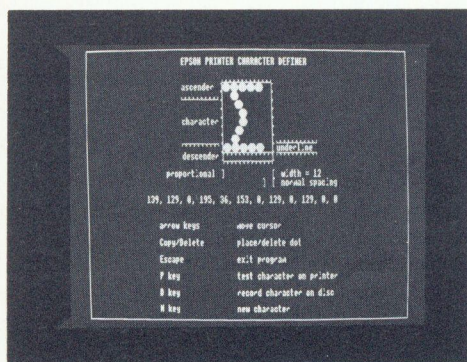
The rule breaks down when a single vertical line of dots is defined, or when the character does not leave the last 2 columns vacant. The program takes care of this automatically, and calculates the attribute value, including the ascender information. A character should be designed in the first 10 columns, having regard for its centring in the non-proportional fonts.

USING THE CHARACTER DEFINITIONS

When writing a program to print user-defined characters to an Epson (or compatible) printer, you will need to include some code in your program to correctly initialise the printer for this purpose. This code is listed below and should be included in your program before the character definitions themselves. Each line has a REM statement giving its function.

```
VDU2:REM Printer ON
VDU1,27,1,64:REM Initialise printer
VDU1,27,1,54:REM Printable code area expanded
VDU1,27,1,73,1,1:REM Control codes printable
VDU1,27,1,58,1,0,1,0,1,0:REM Copy ROM character set into RAM
VDU1,27,1,37,1,1,1,0:REM Select Download RAM
VDU1,27,1,38,1,0:REM Select define function
```

The last two lines should precede each set of sequential character definitions with *first* and *last* being replaced with the first and last ASCII codes of each block.



Defining a new printer character

Key Controls

P print character to printer
D output character design to disc.
N new character

Double Emphasised	£	££££	ABCDEFGHIJ
Single strike	...	££££	ABCDEFGHIJ
Proportional	£	££££	ABCDEFGHIJ

139, 18, 0, 126, 128, 18, 128, 2, 128, 66, 0, 0

PROGRAM NOTES

The DEFPRIN program uses mode 0 and is very tight on space. Model B users should set PAGE to &1400 for the program to work correctly (type PAGE=&1400 <Return> before loading the program).

Pressing key D will send the final character definition to disc or tape. The file name incorporates the ASCII code for the character assigned by the user, for example "char145".

The resulting text file can then be read into an editor or word processor to form a corresponding Basic program line. If neither of these facilities are available, then:

*TYPE char145

(or similar) will put the line on the screen, from where it

may be copied. A REM statement gives the ASCII code allocated.

```
10 REM Program DEFPRIN
20 REM Version B1.4
30 REM Author Cliff Blake
40 REM BEEBUG November 1987
50 REM Program subject to copyright
60 :
100 MODE 0:ON ERROR GOTO 3100
110 DIM grid%(13,9)
120 VDU23,0,10,68,0,0,0,0,0
130 VDU23,0,11,5,0,0,0,0,0
140 *FX144,0,1
150 *FX4,1
160 PROCdefine
170 REPEAT
180 PROCscreen
190 PROCinitial
200 REPEAT
210 PRINT TAB(x%+33,y%+3);:A%=GET
220 IF A%=136 PROCmove(-1,0):GOTO 300
230 IF A%=137 PROCmove(1,0):GOTO 300
240 IF A%=138 PROCmove(0,1):GOTO 300
250 IF A%=139 PROCmove(0,-1):GOTO 300
260 IF A%=135 PROCdot:GOTO 300
```



```

270 IF A%=127 PROCdelete:GOTO 300
280 IF A%=80 PROCprinter:GOTO 300
290 IF A%=68 PROCdisc
300 UNTIL A%=78
310 UNTIL FALSE
320 END
330 :
1000 DEF PROCmove(dx%,dy%)
1010 x%=x%+dx%:y%=y%+dy%
1020 IF x%<1 x%=11
1030 IF x%>11 x%=1
1040 IF y%<1 y%=9
1050 IF y%>9 y%=1
1060 ENDPROC
1070 :
1080 DEF PROCdot
1090 IF y%=1 AND ascend%=0 THEN PROCasc
ender
1100 IF y%=9 AND ascend%=128 THEN PROCd
escender
1110 IF grid%(x%,y%)=2 THEN grid%(x%-1,
y%)=0:grid%(x%,y%)=0:PRINTTAB(x%+32,y%+3
)delete$
1120 IF grid%(x%+1,y%)=1 THEN grid%(x%+
1,y%)=0:grid%(x%+2,y%)=0:PRINTTAB(x%+34,
y%+3)delete$
1130 grid%(x%,y%)=1:grid%(x%+1,y%)=2:PR
INTTAB(x%+33,y%+3)dot$
1140 PROCcalc
1150 ENDPROC
1160 :
1170 DEF PROCdelete
1180 IF grid%(x%,y%)=2 THEN grid%(x%-1,
y%)=0:grid%(x%,y%)=0:PRINTTAB(x%+32,y%+3
)delete$
1190 IF grid%(x%,y%)=1 THEN grid%(x%,y
%)=0:grid%(x%+1,y%)=0:PRINTTAB(x%+33,y%+3
)delete$
1200 IF ascend%=128 GOTO 1260
1210 sum%=ascend%
1220 FOR xt%=1 TO 11
1230 sum%=sum%+grid%(xt%,9)
1240 NEXT xt%
1250 IF sum%=0 PROCascender
1260 PROCcalc
1270 ENDPROC
1280 :
1290 DEF PROCcalc
1300 IF ascend%=0 THEN ref%=9 ELSE ref%
=-8
1310 FOR xt%=1 TO 11
1320 grid%(xt%,0)=0
1330 FOR n%=0 TO 7
1340 yt%=ref%-n%
1350 IF grid%(xt%,yt%)=1 THEN grid%(xt%
,0)=grid%(xt%,0)+2^n%
1360 NEXT n%,xt%
1370 PROCstart:PROCend:PROCwidth
1380 attrib%=end%+16*start%+ascend%
1390 PROCupdate
1400 ENDPROC
1410 :
1420 DEF PROCupdate
1430 PRINTTAB(33,14)SPC14
1440 PRINTTAB(start%+33,14)L$

```

```

1450 PRINTTAB(end%+35,14)R$
1460 PRINTTAB(53,14)SPC5
1470 PRINTTAB(53,14)" = ",width%
1480 PRINTTAB(8,17)SPC69
1490 PRINTTAB(8,17)attrib%;
1500 FOR xt%=1 TO 11
1510 PRINT", ";grid%(xt%,0);
1520 NEXT xt%
1530 ENDPROC
1540 :
1550 DEF PROCascender
1560 ascend%=128
1570 FOR xt%=0 TO 13
1580 grid%(xt%,9)=0
1590 NEXT xt%
1600 PRINTTAB(34,4)b$
1610 PRINTTAB(34,12)n$
1620 ENDPROC
1630 :
1640 DEF PROCdescender
1650 ascend%=0
1660 FOR xt%=0 TO 13
1670 grid%(xt%,1)=0
1680 NEXT xt%
1690 PRINTTAB(34,4)u$
1700 PRINTTAB(34,12)b$
1710 ENDPROC
1720 :
1730 DEF PROCstart
1740 n%=0
1750 REPEAT
1760 n%=n%+1
1770 UNTIL grid%(n%,0) OR n%=12
1780 IF n%=12 THEN start%=0 ELSE start%
=n%-1
1790 ENDPROC
1800 :
1810 DEF PROCend
1820 n%=12
1830 REPEAT
1840 n%=n%-1
1850 UNTIL grid%(n%,0) OR n%=0
1860 IF n%=0 THEN end%=11 ELSE end%=n%+
2
1870 IF end%>11 THEN end%=11
1880 ENDPROC
1890 :
1900 DEF PROCwidth
1910 IF end%-start%>3 THEN 2000
1920 switch%=1
1930 REPEAT
1940 ON switch% GOTO 1950,1970
1950 IF end%<11 THEN end%=end%+1
1960 switch%=2:GOTO 1990
1970 IF start%>0 THEN start%=start%-1
1980 switch%=1
1990 UNTIL end%-start%>3
2000 width%=end%-start%+1
2010 ENDPROC
2020 :
2030 DEF PROCprinter
2040 *FX3,10
2050 VDUI,27,1,64
2060 VDUI,27,1,58,1,0,1,0,1,0
2070 VDUI,27,1,37,1,1,1,0

```



```

2080 VDU1,27,1,38,1,0
2090 VDU1,64,1,64:VDU1,attrib%
2100 FOR xt%=1 TO 11
2110 VDU1,grid%(xt%,0)
2120 NEXT xt%
2130 VDU1,27,1,69:VDU1,27,1,71
2140 PRINTTAB(10);"Double Emphasised @
@@@ ABCDE@FGHIJ"
2150 VDU1,27,1,70:VDU1,27,1,72
2160 PRINTTAB(10);"Single strike ... @
@@@ ABCDE@FGHIJ"
2170 VDU1,27,1,112,1,1
2180 PRINTTAB(10);"Proportional .....
... @ @@@@ ABCDE@FGHIJ"
2190 VDU1,27,1,112,1,0
2200 PRINTTAB(10);attrib%;
2210 FOR xt%=1 TO 11
2220 PRINT", ";grid%(xt%,0);
2230 NEXT xt%
2240 PRINT"'"':*FX3,0
2250 ENDPROC
2260 :
2270 DEF PROCdefine
2280 VDU23,224,0,0,0,0,0,&81,&81,&FF
2290 VDU23,225,&FF,&81,&81,0,0,0,0
2300 VDU23,226,7,1,1,1,1,1,7
2310 VDU23,227,&E0,&80,&80,&80,&80,&80,
&80,&E0
2320 VDU23,228,&F,&3F,&7F,&FF,&FF,&7F,&
3F,&F
2330 VDU23,229,&F0,&FC,&FE,&FF,&FF,&FE,
&FC,&F0
2340 dot$=CHR$(228)+CHR$(229)
2350 delete$=" "
2360 L$=CHR$(226):R$=CHR$(227)
2370 u$=STRING$(12,CHR$(224))
2380 n$=STRING$(12,CHR$(225))
2390 b$=STRING$(12," ")
2400 sp$=STRING$(80,"*"):sp$=""
2410 ENDPROC
2420 :
2430 DEF PROCframe
2440 GCOL 0,1
2450 MOVE 0,0:DRAW 0,1023:DRAW 1279,102
3:DRAW 1279,0:DRAW 0,0
2460 ENDPROC
2470 :
2480 DEF PROCscreen
2490 PROCframe
2500 PROCscrnl:PROCscrn2
2510 ENDPROC
2520 :
2530 DEF PROCscrnl
2540 PRINTTAB(24,1)"EPSON PRINTER CHARA
CTER DEFINER"
2550 PRINTTAB(34,3)u$
2560 PRINTTAB(24,4)"ascender ";L$b$;R$
2570 PRINTTAB(24,5)u$,TAB(33,5)L$b$;R$
2580 PRINTTAB(33,6)L$b$;R$
2590 PRINTTAB(33,7)L$b$;R$
2600 PRINTTAB(24,8)"character";L$b$;R$
2610 PRINTTAB(33,9)L$b$;R$
2620 PRINTTAB(45,10)u$,TAB(33,10)L$b$;
R$

```

```

2630 PRINTTAB(24,11)n$,TAB(33,11)L$b$;
R$;"underline"
2640 PRINTTAB(45,12)n$,TAB(24,12)"desce
nder";L$b$;R$
2650 PRINTTAB(34,13)n$
2660 PRINTTAB(43,15)L$;" ";R$;" normal
spacing"
2670 PRINTTAB(20,14)"proportional ";L$b$;
R$;" width"
2680 ENDPROC
2690 :
2700 DEF PROCscrn2
2710 PRINTTAB(18,20)"arrow keys";SPC9;"
move cursor"
2720 PRINTTAB(18,22)"Copy/Delete";SPC8;
"place/delete dot"
2730 PRINTTAB(18,24)"Escape";SPC13;"exi
t program"
2740 PRINTTAB(18,26)"P key";SPC14;"test
character on printer"
2750 PRINTTAB(18,28)"D key";SPC14;"reco
rd character on disc"
2760 PRINTTAB(18,30)"N key";SPC14;"new
character"
2770 ENDPROC
2780 :
2790 DEF PROCinitial
2800 FOR yt%=0 TO 9:FOR xt%=0 TO 13
2810 grid%(xt%,yt%)=0
2820 NEXT xt%,yt%
2830 x%=1:y%=1:ref%=8:ascend%=128
2840 start%=0:end%=11:width%=11
2850 PROCcalc
2860 ENDPROC
2870 :
2880 DEF PROCsp
2890 FOR n%=1 TO LEN(sp$)
2900 BPUT# file%,ASC(MID$(sp$,n%,1))
2910 NEXT n$:BPUT# file%,13
2920 ENDPROC
2930 :
2940 DEF PROCdisc
2950 VDU28,1,30,78,20,12
2960 PRINTTAB(27,2)"SAVE CHARACTER DEFI
NITION"
2970 PRINT TAB(19,4)"ASCII Number for C
haracter to be Spooled?"
2980 INPUT TAB(27,7)"File: char"asc%
2990 ch$=STR$(asc%):file$="char"+ch$
3000 sp$="VDU1,"+STR$(attrib%)
3010 FOR xt%=1 TO 11
3020 sp$=sp$+"",1,"+STR$(grid%(xt%,0))
3030 NEXT xt%
3040 sp$=sp$+"":REM "+ch$
3050 file%=OPENOUT file$
3060 PROCsp:CLOSE# file$:VDU12,26
3070 PROCscrn2
3080 ENDPROC
3090 :
3100 ON ERROR OFF:*FX4,0
3110 IF ERR=17 MODE7:END
3120 REPORT:PRINT" at line ";ERL
3130 END

```


PINEAPPLE'S DIAGRAM II

Pineapple Software's highly successful drawing package, Diagram, now has even more features. Geoff Bains reports on the latest version.

Product	Diagram II
Supplier	Pineapple Software 39 Brownlea Road, Seven Kings, Ilford, Essex IG3 9NL. Tel. 01-599 1476
Price	£63.25 inc. VAT (discounts available for Diagram I users).

Readers of BEEBUG back in 1985 may remember my ravings over a drawing package called Diagram in Vol.4 No.6. Now Diagram has been rewritten with many additional and improved features. I am happy to say my respect for the package has only been increased.

Diagram II is not like any other drawing package for the BBC micro - and there are certainly enough others! Most drawing packages work in one of two ways: either the start and end co-ordinates of each line are stored, or alternatively the bit map of the screen picture is stored. Both methods have their disadvantages.

Storing the bit map (the screen memory) is wasteful of space on disc. Storing the co-ordinates is a more economical and versatile method, but simple editing is slow and it prohibits the most dramatic feature of Diagram II - smooth scrolling across a picture of up to 30 mode 0 screens in size.

Diagram II stores the code number of each character which makes up the diagram along with the definitions of the character. Of course, such a method is only sensible if the picture contains a majority of repeated characters.

However, Diagram II is (naturally) meant for drawing diagrams which are usually made up of a decidedly limited number of different subsections.

This basic difference to other 'normal' drawing packages is not immediately apparent when you start using Diagram II, because the software effectively hides the mechanics of it from the user. Drawing lines is as simple as with any other package. You just put the program into 'line drawing' mode and move the cursor around the screen with the cursor keys. The program automatically lays down the correct selection of pre-defined characters along the path you trace to create the line on the screen. Turn a corner, and a right angle turn character is printed, cross another line and a cross character is printed at the intersection.

For drawing circuit diagrams, a 'blob' can be left at all line intersections by pressing Ctrl as you reach the join. Now that just isn't possible with other types of drawing package. This character system also means that diagrams are naturally and easily kept aligned and 'square'. If the characters are defined correctly, it is genuinely difficult not to draw neat and impressive diagrams using this software.

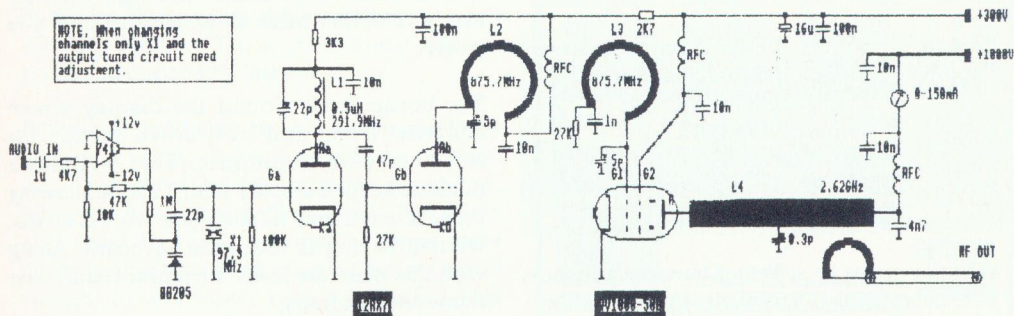
A major improvement over the old Diagram program is the additional ability to use a rubber banding system for angled lines or for tracing an irregular shape. Amazingly, this still uses predefined line section characters, all selected and printed automatically.

Diagram pictures are drawn a screen at a time. Once the whole picture has been set up (if it's a new diagram) on disc, the screen sized section to work on is selected and displayed. The view window can then be scrolled across the whole diagram until the required screenful is visible.

Only characters to produce lines are supplied predefined. The characters to make up other shapes on the diagram must be defined by the user. These are made up of 1 to 12 normal Beeb 8x8 pixel characters. 880 Beeb characters can be defined (only 335 without shadow memory),


character definer, and these can then be automatically used as text is typed in.

Each character can have a 'direction' defined with it. When the character is typed, the cursor moves on ready for the next character in this direction. So text defined as rotated through 90 degrees, or upside down, can be typed with the correct movement inbetween letters.



Other utilities allow up to 8x8 reduced screens to be displayed simultaneously, borders to be added to a diagram, and existing diagrams to be changed in size for additions, or shifted across an enlarged diagram.

Once the diagram has been completed on screen, it is printed using Diagram II's superb printing section. This will accept any printer, but defaults to Epson¹ compatible codes. Diagrams can be printed in excellent quality at any size from three screens to just one character across an A4 page. All the parameters entered to control the size, position and codes used for the printout can be saved to disc for later repeat runs.

Diagram II should not be confused with normal drawing packages. It is only suitable for drawing electronic circuits, architects' plans and other schematic diagrams. However, if these are the kind of picture you want to produce, there is no better package available for flexibility, ease of use, and performance. 



THE COMMS SPOT

In this month's Comms Spot, Peter Rochford takes a look at Squeak, the Mouse-Driven Prestel Terminal Software.

Product Supplier	Squeak Prestel Terminal Software Aldoda International Ltd 27 Elizabeth Mews, London NW3 4UH. Tel. 01-586 5686
Price	£24.50 inc. VAT (disc + ROM) £22.00 inc. VAT (disc only)

When it comes to selecting terminal software, BBC micro owners are really spoilt for choice. There are just so many packages around - of varying price, quality and sophistication. To stand a chance of being successful in an already crowded market, any newly-released terminal package must offer something really different in terms of facilities or ease-of-use. Aldoda International appears to have done just that with 'Squeak', their mouse-driven Prestel software package.

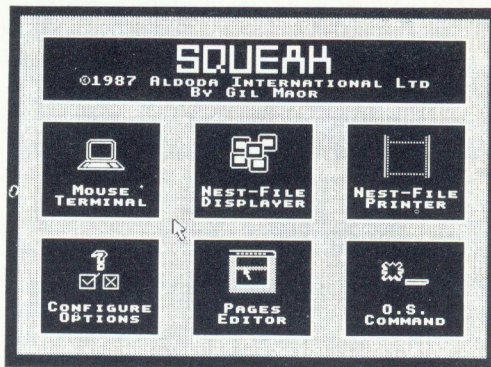
Squeak will work with the BBC Model B and Master, and relies on the AMX Super ROM being present in the machine. Although it is designed to work with a mouse, it can be used via cursor keys or joystick, or alternatively, can be controlled with the usual function keys. It is supplied either on ROM with a utilities disc, or on disc containing a ROM image (to load into sideways RAM) plus the utilities. Along with this you get a rather thin but adequate 16 page user manual.

FACILITIES

Squeak can be used to access any Prestel-type viewdata service, and screens are displayed either in the normal Beeb mode 7, or else in

mode 1, taking up only three quarters of the screen. In mode 7, not all the features of the terminal are available, and the usual arrow mouse pointer is replaced by a block. There is some distortion of the character set too, but this does not alter the legibility of text to any great degree. When using Squeak with its mode 1 screen display, the window, icons, mouse and pointer features of the AMX ROM come into play. However, because it is in mode 1, only four colours are available, although you may easily re-define them to whichever four you prefer.

The border area around the display screen contains icons and pull-down menus for selection with the mouse. This includes a numeric keypad display with * and # allowing input of any frame number you wish to access, without having to touch the keyboard. Along with this there are icons for repeat frame, next frame and last frame.



Where Squeak really comes into its own, is when you wish to select items from a Prestel frame. All you have to do is to move the pointer on the screen to the item you want and click the mouse. It does not really matter much where on the line you move the pointer, as the software searches the line for a number starting left of the pointer. If there is no number to the left it will then search to the right. This is all done at great speed, with only the occasional hiccup when there is more than one possible selection on a single line.

Selected frames may be saved to disc, either singly, or by putting them into a 'nest' file. This nest file may contain as many frames as the available disc space will allow. These can then be re-displayed or printed off line as required, either in carousel fashion or by individual selection. There is also an option to strip the text from the saved frames and send it to a SPOOL file for loading into a word processor. Screens may also be printed out while on line. There is the option of using either a graphics dump for Epson-compatible printers, or a fast text-only printout. The graphics dump is rather good, and reasonably fast too.

Pre-written mail may be sent automatically by Squeak if prepared using Wordwise or View etc. This must be formatted beforehand, however, to suit the particular mailbox frame and must be in 40 columns. Telesoftware downloading is catered for, but files are saved straight to disc rather than to a buffer. This makes for rather slower downloading than I care for.

One nice feature of this package is the on-screen clock that ticks away whilst you are on line. ALL terminal packages should feature clocks I think, as it is so easy to forget just how long you have been logged on.

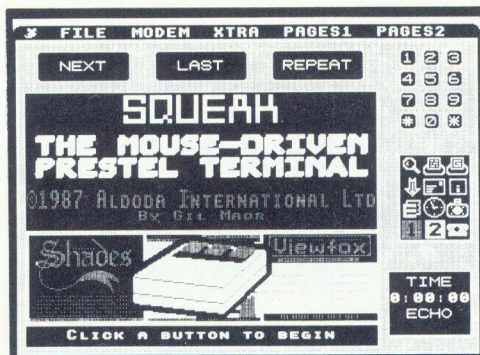
CONFIGURATION

When you first boot-up the Squeak disc, the software asks you to configure the system to suit your own requirements. You may select the input device such as mouse, keyboard or joystick, the baud rates, the function key definitions, and enter your Prestel ID and password for auto log-ons.

You can also elect to load the images of the Squeak ROM and the AMX ROM into sideways RAM, and these will be automatically initialised. Once the configuration has been completed, your choices are saved to disc so that next time the Squeak disc is booted the system will be configured as per your instructions.

VERDICT

This package is indeed very different from any other viewdata terminal for the Beeb. It emulates rather well some of the features found on terminal software for machines like the Amiga. It is strange at first accessing Prestel without the usual character set, colours and full screen, but you quickly get used to it and all seems quite natural after a while.



I would not recommend the package for use with joystick or cursor keys, as suggested by Aldoda, as I have always found the AMX ROM which controls Squeak awkward to use with a joystick and downright tedious with cursor keys.

Would I buy Squeak myself? The answer is yes, but I have reservations about it. For me it lacks certain facilities I am used to and need, which are found on packages like Commsoft and Command. Obviously it is viewdata only, and there is no scrolling terminal facility. I miss very much the auto-dial of my usual software. Having to pick up the phone and dial seems positively archaic now! Modem drivers and auto-dial would be a welcome addition to Squeak.

But Squeak is fun, good value for money and will certainly find favour with those new to comms who require an easy-to-use viewdata terminal. Personally I like it, and will no doubt be using it in the future for a quite a lot of my Prestel log-ons. B

A ROM Filing System Image Generator

With this program by Mark Lock you can save your favourite programs and text files in ROM image format. Then load them into SWR or put them on EPROM for instant retrieval.

As you may be aware, the BBC micro operating system has a fully implemented ROM filing system (RFS). To enter this filing system, just type *ROM. Now, all LOAD, CHAIN, *LOAD, BGET# and *TYPE operations will access the ROM filing system rather than disc. Unfortunately, the ROM filing system, as implemented, does not allow for the SAVING of programs or data, so that it is not a great deal of use.

This is where the accompanying program comes in. It provides a way of generating ROM filing system compatible files. Essentially the idea is that when the accompanying program is run, it asks for the names of a series of files on disc (or cassette). These are then built up into a ROM image in ROM filing system format. At the end of the operation, you will have on disc one or more ROM images, each of which contains a set of programs or text files. The ROM image may then either be blown into EPROM, for permanent installation in your machine, or you may load the image into sideways RAM whenever you require it.

To activate the system, you just need to type *ROM, and then use *CAT to catalogue the contents of your ROM image. The filing system may be used from Basic, or from any piece of

software which loads files from disc or cassette. For example, it works easily from within Wordwise or View using the normal load operations of the word processor. You might wish to store set-up files for setting the function keys, or for entering standard letter heads, configuring printers etc. Essentially, the RFS is ideal for any programs or data which you use repeatedly, and which do not change too often. You might for example store the Filer database suite in this way. The RFS can be particularly useful for utilities because once installed in RAM or EPROM, they will always be present, regardless of which disc you have in your drive. And of course, cassette users will find it a particular boon because it is relatively fast and much more convenient to use than tapes. You may even wish to store the image generating program on ROM.

GETTING STARTED

To get the system up and running, just type in the listing, and save it away to disc or tape. When you are ready to use it, you should collect together the various programs and data which you wish to store in RFS format. Now run the program. It first asks if you wish the ROM header to be date stamped. On a Master it

will automatically take the date from the Master's clock, but on other machine variants you will need to enter the date: any format is acceptable.

```

RFS ROM Generator by M.Lock
Date 10/09/87 10:10:00
Do you require date stamping ? Yes ?
Date 10/10/87
Rom header
HeadName

Image name Discfile1
Input file TestProgl
RFS file Progl UnLocked
Confirm generation ? Yes
Progl 04 48D FFFF0f

Input file MenuVer4
RFS file Menu
Confirm generation ? Yes
Menu 0A A9C FFFF0f

> Input file
  
```

Next you will be asked for a header name. This is just the name that will be incorporated into the ROM header itself: it is not a filename, and any string of up to 10 characters may be

entered. Next you need to give the filename under which the ROM image will be stored on disc. DFS users are allowed a maximum of 7 characters for this, CFS and ADFS users may enter up to 10 characters.

At this stage you start to supply the filenames of the files which you want converted to RFS format. After each name is supplied you will be

asked for a new name for the RFS file for the given item, and here you are allowed up to 10 characters. If there is not enough room on the 16K ROM image for a particular file, the program will inform you of this, and offer to split the file over two ROM images. This option will of course only be useful to users with more than one bank of RAM or EPROM available.

After you have supplied source and destination filenames for each file to be created, you will be asked whether you want the RFS file to be locked. This restricts the way in which the file can be accessed to a *RUN operation. It is therefore an excellent way to lock machine code programs, but should not be used on other file types. Once you have finished converting files, just press Escape, and the disc file containing the ROM image will be closed. To use it, you must load the image into sideways RAM, or blow it directly into EPROM. If you have an ATPL ROM board with sideways RAM on a model B, use:

```
*LOAD filename 8000
```

Users of other boards should consult the instructions supplied with the device. Master and Compact users should use:

```
*SRLOAD filename 8000 n
```

where n is the number of the destination RAM bank (normally 4-7).

Whichever machine system you are using, the ROM image must be initialised before use. Model B users can press Break. Compact users can use the "I" parameter with the SRLOAD command, and Master users must press Ctrl-Break, unless they use the *INIT command on the Master ROM. Now just type:

```
*ROM
```

```
and *CAT
```

to check the contents of your RFS. You can now use any of the files as described earlier in the article.

```
10 REM Program ROM Generator
20 REM Version B 4.9G
30 REM Author Mark Lock
40 REM BEEBUG November 1987
50 REM Program subject to copyright
60 :
100 exit%=FALSE:allin%=TRUE:t%=Fnt:ope
n=FALSE:optfl$=STRING$(22,"#"):ifl$=STR
ING$(11,"#"):ifl$="":optfl$="":lo%=FALSE:
b$=CHR$(148)+STRING$(39,CHR$(172))
```

```
110 vers$="Version 4.9G"
120 ON ERROR ON ERROR OFF:VDU26,31,0,2
1:CLOSE #0:IF ERR<>17 AND ERR<45 REPORT:
PRINT" at line ";ERL':END:ELSE IF ERR=17
PROCe("Escape",TRUE):ELSE REPORT:PRINT:
END
130 d$="":spfl%=FALSE
140 MODE 7:CLOSE#0:*OPT
150 FORZ%=0 TO 1
160 PRINTTAB(0,Z%);CHR$129;CHR$157;CHR
$141;CHR$131;" RFS ROM Generator by M.
Lock"
170 NEXT
180 PRINTTAB(0,Z%);CHR$131;CHR$157;CHR
$129;" (C) 1987 BEEBUG ";vers$'
190 PRINTTAB(0,Z%+1);
200 IF NOT FNfok THEN PROCe("Current f
iling system not acceptable",TRUE)
210 IF t% AND PAGE>=E00 PROCe("With CF
S PAGE must = &E00.",TRUE)
220 DIM prb% &20,flnm &20,hd% &150,c%
&150,mc% &40
230 osasci=&FFE3:osfl=&FFDD
240 d$=FNdate
250 PROCrh(TRUE):IF t% DIM fb% &3FFF
260 o%=0:re%=&BFFF:r%=0:PROCop
270 PROCaCRC:PROCaROM:flst%=d%
280 ON ERROR ON ERROR OFF:VDU26,31,0,2
1:IF ERR<45 REPORT:PROCCls:END:ELSE IF E
RR<> 17 AND ERR <45 REPORT:PRINT" at lin
e ";ERL':END:ELSE PROCCls:IF ERR=17 PRINT
b$:END
290 REPEAT:PROCF1
300 PRINTb$STRING$(10,CHR$10);STRING$(
10,CHR$11)
310 UNTIL(ifl$=" " AND NOT t%) OR exit%
320 PROCCls:END
330 :
1000 DEFPROCaROM:r%=r%+1:p1=4:p2=6
1010 FOR opt=p1 TO p2 STEP p2-p1
1020 O%=c%:P%=&8000:[OPT opt
1030 RTS:RTS:RTS
1040 JMP service:EQUB &82:EQUB copy-&80
00
1050 EQUB r%=EQUB FNromname:BRK
1060 EQUB vers$
1070 .copy BRK:EQUB"(C) 1987 BEEBUG"
1080 BRK:EQUW 0:.service:CMP#&9
1090 BEQ help:CMP#&D:BEQ init:CMP#&E
1100 BEQ getbyt:RTS
1110 .init PHA:JSRadjust:CMP &F4
1120 BCC notus:LDA#d% MOD 256:STA &F6
1130 LDA#d% DIV 256:STA &F7:LDA &F4
1140 JSR adjust2:STA &F5:.done:PLA
1150 LDA#0:RTS:.notus PLA:RTS
1160 .getbyt:PHA:TYA:JSR adjust:CMP &F4
1170 BNE notus:LDY#0:LDA (&F6),Y:TAY
1180 INC &F6:BNE done:INC &F7:JMP done
```



```

1190 .adjust:LDA &F5:.adjust2 EOR #&FF
1200 AND #&F:RTS
1210 .help:PHA:TXA:PHA:TYA:PHA
1220 LDA(&F2),Y:CMPI#&D:BEQ simple
1230 .hlpeX:PLA:TAY:PLA:TAX:PLA:RTS
1240 .simple:LDA title+1:CMPI#&D
1250 BEQ hlpeX:]:PROCChelpOn:[OPT opt
1260 .lp:LDA title,Y:JSRasci:INY
1270 CMPI#0:BNE lp:JMP hlpeX:.title
1280 EQU B &D:EQU S rt$&D:" "+d$
1290 EQU B &D:.title: EQU S RFS number"
1300 EQU S " "+STR$rt$&CHR$&D:BRK:.d$
1310 ]:NEXT:PROCsave(c$,d$-&8000)
1320 flst$=d$:ENDPROC
1330 :
1340 DEFPROCld(I$,L$)
1350 LOCAL P$
1360 FOR P$=I$ TO I$+L$-1
1370 IF NOT t$ ?P$=BGET#fi$ ELSE?P$=FNI
dtb
1380 NEXT:ENDPROC
1390 :
1400 DEFFNcrc(st$,A$)
1410 LOCAL X$,Y$
1420 X$=st$ MOD 256:Y$=st$ DIV 256
1430 =(USR crc)AND&FFFF
1440 :
1450 DEFPROCaCRC:PROCinitCRC
1460 FORpa$=pal$ToPa2$STEPpa2$-pal$
1470 P$=mc$: [OPTpa$
1480 .crc:STY bufpnt+1:STX bufpnt+0
1490 STA ln:LDY#0:STY M:STY L:next
1500 LDA M:EOR (bufpnt),Y:STA M:LDX #8
1510 .lp:LDA M:ROL A:BCC cc:LDA M
1520 EOR #8:STA M:LDA L:EOR #&10:STA L
1530 .cc:ROL L:ROL M:DEX:BNE lp:INY
1540 CPY ln:BNE next:LDA M:LDX L:RTS
1550 ]:NEXT:ENDPROC
1560 :
1570 DEFPROCinitCRC
1580 pal$=0:pa2$=2:m$=&70:ln=FNrmb(1)
1590 bufpnt=FNrmb(2):M=FNrmb(1)
1600 L=FNrmb(1):ENDPROC
1610 :
1620 DEFFNrmb(c):m$=m$+c:=m$-c
1630 :
1640 DEFPROCcls
1650 ?&70=ASC("+"):PROCsave(&70,1)
1660 CLOSE#fo$:open=FALSE:ENDPROC
1670 :
1680 DEFPROCgfn:fi$=0:REPEAT
1690 IF t$ m$=11:ELSEm$=21
1700 PROCifl("")
1710 IF ifl$<>" AND NOT t$ fi$=OPENIN
ifl$
1720 IF ifl$<>" AND fi$=0 AND NOT t$ P
ROCe("Can't find"+CHR$133+ifl$,FALSE)
1730 UNTILfi$<>0 OR ifl$="" OR t$

```

```

1740 IF NOT t$ AND ifl$="" THEN exit$=T
RUE:ENDPROC
1750 PROCOptfl("")
1760 PROClock(TRUE)
1770 ENDPROC
1780 :
1790 DEFPROCfullhedr:$hd$="*"+optfl$
1800 hp$=LEN(optfl$)+2+hd$
1810 ?(hd$+LEN(optfl$)+1)=0
1820 dp$=hp$+19:hp$!0=la$:hp$!4=ea$
1830 hp$!8=blkn$:hp$!10=1$
1840 hp$!12=(&80 AND (FNp+1$>=FNe))OR(&
40 AND 1$=0)+lo$
1850 hp$!13=flst$+FNfln
1860 IF spfl$ AND blkn$=0 THEN hp$!13=
FNeor
1870 IF spfl$ AND (blkn$<>0) hp$!13=o$
+&8000+FNeofflp
1880 IF t$ AND spfl$ AND blkn$=0 THEN
hp$!13=flst$+FNfln
1890 IF allin$=FALSE THEN hp$!12=lo$
1900 hp$!17=FNcrc(hd$+1, hp$-hd$+17-1)
1910 ENDPROC
1920 :
1930 DEFPROCchashhedr:$hd$=ASC("#")
1940 dp$=hd$+1:ENDPROC
1950 :
1960 DEFFNfln:LOCAL blks$,l$
1970 blks$=(FNe+255)DIV 256
1980 l$=LEN(optfl$)+21
1990 IF blks$>1 l$=l$+1$
2000 IF blks$>2 l$=l$+(blks$-2)
2010 l$=l$+(blks$*2):l$=l$+FNe:=l$
2020 :
2030 DEFPROCfl:PROCgfn
2040 IF exit$ ENDPROC
2050 blkn$=0:lbn$=-1
2060 IF t$ PROCldfl
2070 IF ifl$="" AND NOT t$ ENDPROC
2080 PROCnrIfull
2090 open=TRUE
2100 IF exit$ ENDPROC
2110 PROCcistr
2120 IF exit$ ENDPROC
2130 IF FNyn(CHR$130+"Confirm generatio
n")=FALSE ENDPROC
2140 PROCgfa:aplt$=FALSE:Y$=VPOS
2150 REPEAT
2160 PRINTTAB(0,Y$);CHR$135;optfl$;TAB(
12,Y$);STRING$(2-LEN(STR$-blkn$),"0");~
blkn$;
2170 PROCcrblk:aplt$=FALSE
2180 IF FNnmr THEN aplt$=TRUE:PROCnmr:I
F t$ THEN PROCldfl
2190 UNTIL aplt$=FALSE AND FNe-FNp=0 AN
D allin$=TRUE
2200 flst$=&8000+o$
2210 PRINT " ;~FNe;" " ;~la%;" " ;~ea$

```



```

2220 IF NOT t% CLOSE#fi%
2230 ENDPROC
2240 :
2250 DEFFNfullblk:LOCAL blks%,l%
2260 blks%=(FNe+255)DIV 256
2270 l%=LEN(optfl$)+21
2280 IF blks%=1 =l%+FNe:ELSE =l%+256
2290 :
2300 DEFFPROCgfa:$flnm=ifl$
2310 IF NOT t% !prb%=flnm:A%=5:X%=prb%:
Y%=X% DIV 256:CALLosfl
2320 la%=prb%!2:IF t% la%!=&3BE
2330 ea%=prb%!6:IF t% ea%!=&3C2
2340 l%=FNe:ENDPROC
2350 :
2360 DEFFPROCcrblk:l%=FNe-FNp
2370 IF l%>256 l%=256
2380 IF FNsplittedone OR blknm%=0 OR (FNe
-FNp<=256) PROCfullhedr ELSE PROChashed
r
2390 PROCld(dp%,l%)
2400 dp%!l%=FNcrcl(dp%,l%)
2410 PROCsave(hd%,dp%+l%+2-hd%)
2420 blknm%=blknm%+1:ENDPROC
2430 :
2440 DEFFNblkadd
2450 IF blknm%=0 OR (FNe-FNp)<=256 THEN
=FNfullblk:ELSE =259
2460 :
2470 DEFFNi(s$,l%,t%)
2480 LOCALX%,Y%,x%,y%:x%=POS:y%=VPOS
2490 IF s$<>" " PRINT TAB(0,y%);CHR$135;
s$;
2500 IF s$<>" " x%=POS
2510 REPEAT:PRINTTAB(x%,y%);
2520 !prb%=flnm:prb%?2=t%-1:prb%?3=32:p
rb%?4=126:X%=prb% MOD 256:Y%=prb% DIV 25
6:A%=0:CALL &FFF1
2530 UNTIL LEN($flnm)<t% AND LEN($flnm
)>=l%
2540 IF s$<>" " PRINTCHR$11;CHR$11;TAB(0
);CHR$134:ELSEPRINTTAB(x%-1,y%);CHR$131
2550 =$flnm
2560 :
2570 DEFFNyn(string$):LOCAL k%,X%,Y%
2580 PRINTstring$:X%=POS:Y%=VPOS
2590 PRINTCHR$130;" (Y-N)";CHR$135;"?";
2600 REPEAT:k%=(INSTR("YNyn",GET$))
2610 UNTILk%>0:k%=k%MOD2
2620 IF k%=1PRINTTAB(X%,Y%);" ? Yes " :
ELSEPRINTTAB(X%,Y%);" ? No "
2630 PRINTTAB(0,Y%):IF k%=1 PRINTCHR$1
30:~k%:ELSEPRINTCHR$130
2640 =FALSE
2650 :
2660 DEFFROCsave(st%,l%):LOCAL point%
2670 FOR point%=st% TO st%+l%-1
2680 BPUT#fo%,?point%:NEXT

```

```

2690 o%=o%+l%:ENDPROC
2700 :
2710 DEFFPROCop:fo%=0:REPEAT
2720 PROCofl(TRUE):open=TRUE
2730 fo%=OPENOUT ofl$
2740 IF fo%=0 open=FALSE:PROCe("Can't o
pen "tofl$,FALSE)
2750 UNTILfo%<>0:ENDPROC
2760 :
2770 DEFFPROCclsop:PROCcls:PROCop
2780 ENDPROC
2790 :
2800 DEFFNpt(s$):IF s$<>" " THEN =s$
2810 =CHR$134+"(Null)"
2820 :
2830 DEFFNnumberofblks=((re%-flst%)-(2
1+256+LEN(optfl$)))DIV 259
2840 :
2850 DEFFROCe(mess$,f):*FX 15,1
2860 IFmess$="" ENDPROC
2870 IF NOT f PRINTCHR$130;mess$:ELSE P
RINTCHR$133;mess$
2880 IF NOT f ENDPROC
2890 IF f CLOSE#0:END
2900 IF NOT open END
2910 PROCcls:END
2920 :
2930 DEFFROChelpon
2940 IF r%=1 Q%=0:ELSEQ%=titlen-title
2950 [OPT opt:LDY#Q%]:ENDPROC
2960 :
2970 DEFFNf
2980 A%=0:X%=0:Y%=0:USR(&FFDA)AND&FF
2990 DEFFNt:IF Fnf=1 OR Fnf=2 THEN=TRUE
ELSE=FALSE
3000 DEFFNFok:IF Fnf=3 OR Fnf=6 OR Fnf=
0 THEN =FALSE ELSE =TRUE
3010 DEFFNe:IF t%=FALSE =EXT#fi%
3020 =e%
3030 :
3040 DEFFNp
3050 IF t% THEN =p% ELSE =PTR#fi%
3060 :
3070 DEFFROCc(s$):PRINTCHR$133;"Insert"
;CHR$134;s$:CHR$133;"tape, press"CHR$131
"any key.":IF GET ENDPROC
3080 :
3090 DEFFROCifl(a$)
3100 IF t% PROCc("LOAD")
3110 PRINTCHR$135;"Input file ";:IF a
$<>" " PRINTCHR$8;CHR$131;FNpt(a$) ELSEif
l$=FNI("","0,m%)
3120 ENDPROC
3130 :
3140 DEFFROCptfl(a$)
3150 PRINTCHR$135;"RFS file ";:IFa$
<>" "PRINTCHR$8;CHR$135;FNpt(a$) ELSEptf
l$=FNI("","1,11)

```



```

3160 ENDPROC
3170 :
3180 DEFPROCofl(a%)
3190 PRINTCHR$134;"Image name";SPC3;
3200 IF a% THEN ofl$=FNI(" ",1,21):ELSE
PRINTCHR$8;CHR$131;ofl$
3210 ENDPROC
3220 :
3230 DEFPROCrh(a%)
3240 PRINTCHR$134;"Rom header";SPC3
3250 PRINTCHR$134;SPC3;
3260 IF a% THEN rt$=FNI(" ",0,35) ELSE P
RINTCHR$8;CHR$131;rt$
3270 PRINTb$:VDU 28,0,24,39,VPOS
3280 ENDPROC
3290 :
3300 DEFPROCldfl:fi%=OPENINifl$
3310 LOCAL m%,Y%:m%=FNin:Y%=VPOS:e%=0
3320 REPEAT:IF (e% AND &FF)=0 lblknm%=1
blknm%+1:PRINTTAB(0,Y%);CHR$134;ifl$:TAB
(12,Y%);STRING$(2-LEN(STR$~lblknm%),"0")
;~lblknm%;
3330 e%?fb%=BGET#fi%:e%=e%+1
3340 UNTILe%>m% OR EOF#fi%
3350 allin%=EOF#fi%
3360 IF EOF#fi% CLOSE#fi%
3370 p%=0:PRINT " ";~e%:PROCc("SAVE")
3380 ENDPROC
3390 :
3400 DEFFNin:LOCAL z%:z%=re%-flst%-1
3410 =((z%-21-LENofl$)+3)/259*256-1
3420 DEFFNldtb:p%=p%+1:fb%?(p%-1)
3430 :
3440 DEFPROCnr:LOCAL y%:y%=VPOS
3450 PRINTCHR$134;"A new ROM about to b
e generated .."
3460 PROCclsop:o%=0
3470 PROCaROM:flst%=d%
3480 PRINTTAB(0,y%);SPC(80);TAB(y%)
3490 ENDPROC
3500 :
3510 DEFPROCnrIfull
3520 IF (re%-flst%)>292 THEN ENDPROC
3530 exit%=TRUE:r%=0
3540 PRINT" This ROM is completely full
."
3550 IF FNyn(CHR$134+"Use another") THE
N PROCnr : exit%=FALSE
3560 ENDPROC
3570 :
3580 DEFPROCcistr:spfl%=FALSE
3590 IF t%=FALSE AND (re%-flst%)>FNfln
THEN ENDPROC
3600 IF t% AND allin% ENDPROC
3610 exit%=TRUE
3620 PRINT" This file is too long for t
he remaining"" ROM space."
3630 IF FNyn(CHR$134+"Split it across t

```

```

wo ROMS") THEN exit%=FALSE:spfl%=TRUE:EN
DPROC
3640 IF FNyn(CHR$134+"Start a fresh ROM
, then") THEN PROCnr:exit%=FALSE:r%=1
3650 ENDPROC
3660 :
3670 DEFFNnmr:IF NOT spfl% THEN =FALSE
3680 IF o%+&8000+FNblkadd > re%-1 THEN
=TRUE
3690 IF t% AND NOT allin% AND FNp>=FNe
=TRUE
3700 =FALSE
3710 :
3720 DEFFNeor
3730 =flst%+(21+256+2+LEN(optfl$))+FNnu
mberofblks *259
3740 :
3750 DEFFNeofflpart:LOCAL blks%,l%
3760 blks%=(FNe-FNp)+255)DIV 256
3770 l%=LEN(optfl$)+21
3780 IF blks%>1 l%=l%+1%
3790 IF blks%>2 l%=l%+(blks%-2)
3800 l%=l%+(blks%*2):l%=l%+(FNe-FNp)
3810 =l%
3820 :
3830 DEFFNsplitdone
3840 IF aplt%=FALSE =FALSE
3850 IF d%=flst% =TRUE
3860 =FALSE
3870 :
3880 DEFFNdate
3890 IF FNyn(CHR$(131)+"Do you require
date stamping")=FALSE THEN =""
3900 IFFNmaster THEN =LEFT$(TIME$,15)
3910 =FNI("Date ",0,16)
3920 :
3930 DEFFNmaster
3940 IF INKEY(-256)=253 =TRUE
3950 =FALSE
3960 :
3970 DEFPROCnmr:PRINT:PROCnr:aplt%=TRUE
:ENDPROC
3980 :
3990 DEFPROCclock(a%)
4000 PRINTTAB(29);CHR$11;
4010 IF a%=FALSE AND lo%=0 PRINTCHR$130
;"Un"; ELSE PRINT " ";CHR$130;
4020 IF a%=FALSE PRINT"Locked":ENDPROC
4030 PRINT"U/L";:REPEAT:a$=GET$:UNTILIN
STR("UuLl",a$)<>0
4040 lo%=0:IF a$="L"OR a$="l" lo%=1
4050 PROClock(FALSE):ENDPROC
4060 :
4070 DEFFNromname
4080 IF FNmaster THEN =RFS : "+d$+" nu
mber":ELSE="RFS : "+d$+" number "+STR$r%

```

B

AUTO- CONFIRM UTILITY

The accidental overwriting of a valuable file can be a nightmare for disc users. Bernard Hill's short utility will end all that by automatically checking any file save operation, regardless of the application. Now it's sweet dreams all the way.

This short machine code utility is designed to sit unnoticed in your micro until you try to save a file to disc. Whether you are running your own Basic program (using OPENOUT etc), working with a commercial ROM such as Interword or BEEBUG's own Command ROM, or just saving to disc using SAVE or *SAVE, this routine will spring to life as soon as it detects an attempt to save to disc. If a file of the same name already exists, then the routine will seek confirmation from the user before allowing the save to proceed. Of course, if the routine is used with a package like Wordwise Plus which already carries out such a check, it will now occur twice.

No other action is required than to install the assembled routine in your machine. You may find in some cases that your screen display is somewhat corrupted, but that is a small price to pay for the assurance that no file will be accidentally overwritten.

INSTALLING THE ROUTINE

Type the program into your micro and save to disc. Then run the program to assemble the machine code which is automatically saved to disc under the name 'Confirm'. From then on, typing *Confirm (or including this line in a Basic program) is all that is needed to install the routine ready for work. If you press Break at any time, you will need to reinitialise the routine by issuing another *Confirm.

```
10 REM Program Confirm
20 REM Version B1.1
30 REM Author Bernard Hill
40 REM BEEBUG October 1987
50 REM Program subject to copyright
60 :
100 FOR opt=0 TO 2 STEP 2
110 P%=&900:Q%=P%
120 [ OPT opt
130 SEI:LDA &212:STA oldfilev
140 LDA &213:STA oldfilev+1
150 LDA &21C:STA oldfindv
160 LDA &21D:STA oldfindv+1
170 LDA #filev MOD 256:STA &212
180 LDA #filev DIV 256:STA &213
190 LDA #findv MOD 256:STA &21C
200 LDA #findv DIV 256:STA &21D
210 CLI:RTS
220 .filev
230 CMP #0:BNE okfilev
240 LDA &A8:PHA:LDA &A9:PHA
250 STX &A8:STY &A9:LDY #0
260 LDA (&A8),Y:TAX:INY
270 LDA (&A8),Y:TAY:JSR check
280 LDX &A8:LDY &A9:PLA:STA &A9
290 PLA:STA &A8:LDA #0
300 .okfilev
310 JMP (oldfilev)
320 .findv
330 CMP #&80:BNE okfindv:JSR check
340 .okfindv
350 JMP (oldfindv)
360 .err
370 BRK:EQU 255:EQU "Aborted"
380 EQU 0
390 .mess
400 EQU "File exists - please confirm:
"+CHR$0
410 .check
420 PHA:LDA #&40:JSR okfindv
430 CMP #0:BNE check1:PLA:RTS
440 .check1
450 STA &100:TXA:PHA:TYA:PHA
460 LDY &100:LDA #0:JSR okfindv
470 LDX #0:.loop LDA mess,X
480 BEQ answer:JSR &FFE3:INX
490 BNE loop
500 .answer
510 JSR &FFE0:JSR &FFE3:AND #&DF
520 CMP #ASC"Y":PHP:JSR &FFE7
530 PLP:BNE err
540 PLA:TAY:PLA:TAX:PLA:RTS
550 .oldfilev EQUW 0
560 .oldfindv EQUW 0
570 ]
580 NEXT
590 OSCLI("SAVE Confirm "+STR$~Q%+" "+
STR$~P%)
```




Part 5

ASSEMBLER

A series for complete beginners to machine code by Lee Calcraft

This month: Addition and Subtraction

THE 6502's ARITHMETIC INSTRUCTION SET

Apart from the increment and decrement instructions which we have already come across, the 6502 has just two arithmetic instructions. These are ADC - Add with Carry - and SBC - Subtract with Carry. As you might expect, these are only single byte instructions, so if you wish to perform multi-byte addition or subtraction, or the more complex operations of multiplication and division, you must write the code yourself using these and other instructions as a basis. Let us begin by taking a look at the addition instruction.

ADD WITH CARRY

The effect of the ADC instruction is best conveyed through example. Take the following sequence:

```
CLC
LDA #4
ADC #3
```

Ignoring for a moment the CLC instruction, the two which follow it load the accumulator with the value 4 then add 3 to it, leaving the result in the accumulator. Here we have used so-called *immediate mode* addressing, which means that the operand supplied with ADC (i.e. the 3) is not a memory location but a single-byte number. The ADC instruction can in fact take any of the addressing modes introduced earlier in the series. For example:

```
ADC address
ADC address,X
ADC (address),Y
```

The first of these uses absolute addressing to add to the accumulator the byte held at location *address*; while the second uses indexed addressing to add to the accumulator the number found at memory location *address + X*.

The third uses the more complex indirect indexed addressing discussed in the last issue.

So far we have ignored the "carry" part of the operation. When the 6502 processes an ADC it always takes account of the state of the carry flag. If the flag is set, then it adds one to the result of its addition. If it is unset, then it does not. Thus the following two cases will give different results:

```
CLC          Clear carry flag
LDA #4
ADC #3
```

In this case the carry flag is cleared before the addition, so the result is 7 as we would expect.

```
SEC          Set carry flag
LDA #4
ADC #3
```

Because the carry flag is now *set* before the addition takes place, the result is not 7 but 8. Obviously if you are using ADC for single byte addition then you must remember to clear the carry flag before each addition.

There are in fact four flags affected by the ADC instruction. Of these, the carry flag is in many respects the most important. It is set if the result of an addition exceeds 255 decimal (or &FF hex). Otherwise it is reset (to zero). The zero and negative flags are also affected, as they are with any operation which changes the contents of the accumulator. Finally, the overflow flag is used by the processor to indicate certain conditions of interest when the so-called signed integer mode is in use, and we can ignore this here.

So far we have only considered the use of ADC with immediate addressing. In real life the operands are more likely to be held in memory. The following example adds the contents of two memory locations *Int1* and *Int2*, and stores the result in the location *Answer*.

```
CLC
LDA Int1
ADC Int2
STA Answer
```

As it stands, this will give the correct result unless the sum exceeds 255 decimal. We could detect this situation by checking to see if the carry flag is set by using BCC (Branch if Carry Clear) or BCS (Branch if Carry Set), but in some respects it is more useful to respond to a carry by using multi-byte addition techniques.

MULTI-BYTE ADDITION

The way in which the 6502 ADC instruction handles the carry flag makes the implementation of multi-byte addition extremely easy. Suppose we wish to add two two-byte integers together. Assume that they are held in memory at Int1 (with Int1+1 holding the high byte) and Int2 (with Int2+1 holding the high byte), and that the result is to be stored at Ans and Ans+1; as ever with the 6502, this is held low byte first. The following code is (almost) all that is needed:

```
CLC
LDA Int1      Low byte of 1st no
ADC Int2      Low byte of 2nd no
STA Ans       Low byte of answer
LDA Int1+1    Hi byte of 1st no
ADC Int2+1    Hi byte of 2nd no
STA Ans+1     Hi byte of answer
```

The first four instructions carry out a simple addition on the low bytes of the two numbers, and store the result as the low byte of the answer. The operation is then repeated for the high bytes, except that the carry flag is not cleared. This will now automatically take account in the high byte addition of any carry from the low byte addition. If the result exceeds 255 decimal, the carry flag will cause an extra one to be added to the high byte sum. The only thing that we have not yet covered is the possibility of a high byte carry. This is catered for by adding the following three instructions:

```
LDA #0
ADC #0
STA Ans+2
```

This simply performs a null addition so that any carry from the previous high byte addition will appear as the result, giving one if a carry occurred, and zero if it did not. The complete result may now be read as a three byte integer (at Ans, Ans+1 and Ans+2). As you can probably see, it is an easy matter to construct an algorithm for addition to any level of precision required. The same is true of subtraction.

SUBTRACTION

The 6502's SBC instruction works in a very similar way to ADC except that it uses the carry flag to signify a *borrow* rather than a carry, and the flag's role is reversed. When set it indicates that no borrow has taken place and vice versa. Thus before you start a single-byte subtraction, you must *set* the carry flag rather than clear it.

Again, an example should help:

```
SEC
LDA #5
SBC #3
```

This short sequence will subtract 3 from 5, leaving the result in the accumulator. Since the second operand (i.e. the 3) is smaller than the first, no borrow takes place, and the carry flag remains set after the calculation. If the number subtracted had been the larger of the two, the carry flag would have been reset to zero, to indicate that a borrow had occurred.

Carry flag (C)	Set if the sum of a binary addition exceeds decimal 255 (hex &FF); otherwise it is reset.
Zero flag (Z)	Set if the sum is zero; otherwise it is reset.
Negative flag (N)	Set if bit 7 of the result is logic 1; otherwise it is reset.
Overflow flag (V)	May be ignored unless dealing with signed integers.

Flags altered by the ADC instruction

MULTI-BYTE SUBTRACTION

If we now move straight on to multi-byte subtraction, you will see more clearly the role played by the carry flag. The listing below performs a simple two-byte subtraction, using the same memory locations as the earlier addition example:

```
SEC
LDA Int1      Low byte of 1st no
SBC Int2      Low byte of 2nd no
STA Ans       Low byte of answer
LDA Int1+1    Hi byte of 1st no
SBC Int2+1    Hi byte of 2nd no
STA Ans+1     Hi byte of answer
```

The first four instructions execute an ordinary single-byte subtraction, storing the result in Ans. The high byte subtraction then proceeds without a new instruction to zero the carry flag. This causes any borrow operation from the first subtraction to be automatically taken into account in the second. The result is stored in Ans+1, so that the two bytes at Ans and Ans+1 contain the full result, low byte first.

SIGNED INTEGERS

This example will give correct results providing that the second operand is not greater than the first. To see what happens when a larger number is subtracted from a smaller one, we will take a closer look at the case of single byte subtraction. Suppose that we execute:

```
SEC
LDA #5
SBC #7
```

At the end of this sequence, the carry flag will be *reset*, and the accumulator will contain 254 (&FE hex). This is exactly the same value that it would have held if we had loaded it with 5 and then decremented it 7 times. The reason why the accumulator holds 254 is that it just cycles around when it is incremented above 255 or decremented below zero. If you add 1 to 255 the result is zero (with a carry), and by the same token, if you subtract 1 from zero the result is 255 (with a borrow). The sum 5-7 thus gives the result 254; or in binary:

```
0000 0101
-0000 0111
1111 1110
```

Now this is actually quite useful because it suggests a way of treating signed integers. By this convention the number 2 would be represented, as we would expect, by the binary number 0000 0010, while -2 would be represented by 1111 1110. A further important feature of the convention is that adding -2 to +2 gives a meaningful result:

```
0000 0010   or   &02
+1111 1110   +&FE
10000 0000   &100
```

The result ought of course to be zero. It is in fact zero plus a carry.

Using this signed integer convention the top bit of any number is taken as the sign bit. If you are dealing in 4-byte integers, then it is bit 31 which indicates the sign of the number. If you are dealing in single byte integers, it is bit 7 (10000000 binary), and so on. Of course, setting aside the top bit of a number to indicate its sign reduces by a half the maximum positive number that can be represented by a given number of bytes. For example, a single byte can be used to represent the 256 integer values from 0 to 255 when using unsigned integers, but if the signed integer convention is in use, the same byte represents numbers in the range -128 (1000 0000) to +127 (0111 1111 binary).

Dec	Binary	Dec	Binary
127	0111 1111	-1	1111 1111
126	0111 1110	-2	1111 1110
125	0111 1101	-3	1111 1101
3	0000 0011	-125	1000 0011
2	0000 0010	-126	1000 0010
1	0000 0001	-127	1000 0001
0	0000 0000	-128	1000 0000

Examples of the signed integer convention

Another aspect of this convention is that negative numbers are said to be held in two's complement form. What this means is that to obtain a negative number from a positive one, you just take the binary representation of the positive number, reverse all the zeros and ones, and add one. For example to find the binary representation of -2, take the binary representation of 2:

```
0000 0010
```

invert it:

```
1111 1101
```

and add one:

```
1111 1110
```

As you can see, this is indeed the same number that we were using before for -2, &FE hex, or 254 decimal. As a matter of interest this fact also provides the basis for the algorithm by which most processors perform subtraction. To subtract one number from another, they generate the two's complement of the second operand, and add it to the first.

It should be stressed however that the signed integer convention is no more or less than a convention. It is up to the programmer to make use of it at any point which is convenient, and to use unsigned positive integers at other times. The processor does not need to know which convention the programmer is using, though to assist in the use of this convention, the designers of the 6502 have incorporated an overflow flag which the 6502 sets to one whenever the result of an addition would give ambiguous results if the programmer happened to be using the signed integer convention. BBC Basic uses this convention for the storage of integers. Thus if you type:

```
A%=&FFFFFFFF
PRINT A%
```

the result will be -1, and not the decimal equivalent of &FFFFFFFF as you might have

expected. It may be worth experimenting with this a little to get the feel of using signed integers.

```
First number ? 125
Subtract ? 42
Result (decimal)....83
Result (hex).....&53
```

FOUR-BYTE SIGNED INTEGER SUBTRACTION

As a tailpiece to this month's article, I have included a program which performs a four-byte subtraction on numbers input at the keyboard. When you run it, it requests two integer operands, and prints out the result in both decimal and hex. The input uses Basic's EVAL function, so that operands may be supplied in decimal or hex, providing that the latter are preceded by an ampersand (&).

Because the input and output of the routine are handled by Basic, the signed integer convention is assumed. As a result you may even enter negative numbers as operands without tripping up the program. The maximum positive integer allowed is &7FFFFFFF, since on the sign convention the number which is one digit higher, &80000000, is taken to be a negative number because its top bit is set. The program also warns if a so-called overflow condition has taken place, though we do not have the space to pursue this here.

PROGRAM NOTES

The heart of the routine lies in lines 240 to 260. This subtraction sequence, which uses X-indexed addressing, is called four times for each calculation in order to complete the four-byte subtraction. The Y register rather than the X register is used as the loop counter, since if we used X we would be forced to check its value with a CPX #4 instruction. This would affect the carry flag (CMP, CPX and CPY all affect the carry flag), and we would lose all carry information from the subtraction. Instead we use the Y register, and decrement it until it reaches zero, thus avoiding the need for a compare instruction.

The overflow condition is picked up at line 310 using BVC (Branch if oVerflow Clear). If there is an overflow, the accumulator is loaded with &FF (line 320), and this is picked up by Basic after the code has run. In order to find the

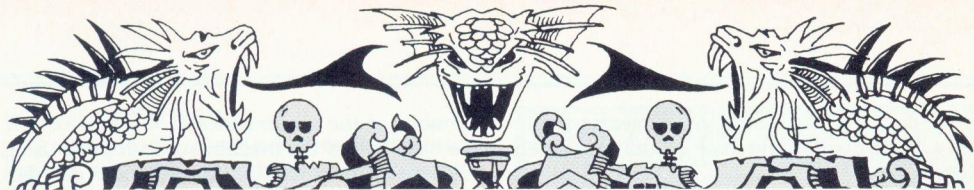
contents of the accumulator at the time of exit, we have called the machine code routine using USR rather than CALL. Line 450 both executes the code at &900, and sets the variable X to a four-byte value as follows:

X=ssyyxxaa

where aa represents the contents of the accumulator on exit, xx, the processor's X register, yy its Y register, and ss its status register. Line 480 checks to see if the accumulator contained zero on exit. If not it indicates that an overflow condition occurred, and a message to that effect is displayed.

Next month we will take a look at the 6502's more extensive set of logical instructions.

```
100 REM Four Byte Subtraction
110 REM Version B 0.4C
120 Int1=&70:Int2=&74:Ans=&78
130 MODE 7
140 FOR pass=0 TO 1
150 P%=&900
160 [
170 OPT pass*3
180 \      **FOUR BYTE SUBTRACTION**
190 LDX #0
200 LDY #4
210 SEC
220 .subloop
230 LDA Int1,X
240 SBC Int2,X
250 STA Ans,X
260 INX
270 DEY
280 BNE subloop
290 LDA #0
300 BVC ok
310 LDA #&FF
320 .ok
330 RTS
340 ]
350 NEXT
360 REPEAT
370 INPUT"First number ? "A$
380 A%=EVAL(A$)
390 INPUT"Subtract ? "B$
400 B%=EVAL(B$)
410 !Int1=A%
420 !Int2=B%
430 X=USR900
440 PRINT"Result (decimal)....";!Ans
450 PRINT"Result (hex).....&";STR$~!
Ans
460 IF (X AND &FF) <>0 THEN VDU7:PRINT
"***Result invalid*****"
470 UNTIL FALSE
```

ADVENTURE GAMES by Mitch ADVENTURE GAME



Disraeli is quoted as having said, "When I want to read a novel, I write one!" This could be your solution to the current dearth of adventures for the BBC micro. But how to go about it, 'Aye there's the rub'.

Re-inventing the wheel is never a good idea, so perhaps you should take a leaf out of the professional's book and use a custom built package to help you. Wherever possible, programmers draw suitable chunks of pre-written code from libraries of useful routines, and when these have been built into a simple language, things are made much easier. This month sees the release of just such a piece of software called ALPS - (Adventure Language Programming System).

Product	ALPS
Supplier	Summit Software PO Box 25, Portadown, Craigavon BT63 5UB. Tel. (0762) 42510
Price	£28.95 inc. VAT and p&p (disc) £27.95 inc. VAT and p&p (tape)

Using such a program to create adventures is not a new development. Previous examples available for the BBC micro include 'The Quill' and 'The Graphic Adventure Creator', both extremely successful and widely used by amateur and professional alike. The software house Delta 4 has produced a number of commercial games using The Quill, and these were generally well received. Graphics adventures on the Beeb have not been very successful - maybe because of the lack of memory. Perhaps that is the reason why the ALPS package has chosen to concentrate on text only adventures.

The package is supplied on a 16K ROM with a floppy disc containing extra tools. The games compiled under this system may be freely run by other machines not containing the ALPS ROM, so you are free to give copies of your masterpiece to your friends (or customers). The program permits players to use complex sentences such as, "PRESS THE SWITCH AND THROW THE BOMB EAST". Two adventure games are provided on the accompanying disc, 'Tiny' and 'Mysterious Mission'.

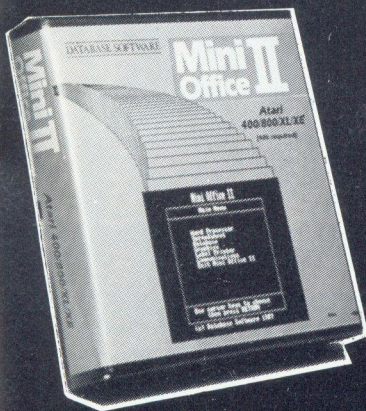
'Tiny' has been included as a simple exercise piece to help you understand the production of a small game. 'Mysterious Mission' is a full length adventure written by the author to demonstrate the capabilities of the system.

The ALPS language provides simple Basic-like structures using IF-THEN-ELSE plus procedures. While writing games using a custom built language is much easier than by other methods, it still requires a basic knowledge of the logic of programming to produce anything worthwhile. Understanding the concepts of 'flags', procedure calls and 'switches' is relatively simple, but essential when writing the smallest game. As proof of how easy it is to get it wrong let me cite the case of the game supplied, 'Mysterious Mission'.

This is a full sized adventure involving your travels through 120 locations. Beginning in a hotel room, you are soon besieged by assassins and surrounded by corpses. Early in the game you will be found languishing in a cell containing a locked door and a bed. In answer to my command 'STAND ON BED' the game responded: "You don't have a battery lamp, which is off", I then decided to blast my way out with the command 'THROW BOMB EAST'. As nothing appeared to happen I attempted to retrieve the bomb, only to find it had inexplicably disappeared from the locked room. Not with a bang or even a whimper. Feeling desperate I decided to 'KNOCK ON THE DOOR'. Again I was informed, "You don't have the battery lamp". True I didn't, but that was hardly relevant! This does serve to demonstrate that logic can be devilish difficult even for the professionals.

The supplied manual is adequate, and the various tools and error messages should ensure that with dedication you could create a pretty good game. Its main advantage, however, will be the use of a ROM, which ensures that more memory is available for program development. The Quill has many devotees who have written little extra routines to enhance the original, and these are available through specialist adventure groups. Whilst ALPS is perfectly suitable, it presents no great advantage over existing systems. **B**

Mini Office II



Mini Office II is a comprehensive integrated suite of programs setting a new standard in home and business software.

Most of the wide range of features – many of which are usually only available on software costing hundreds of pounds – are easily accessed by using cursor keys to move up and down a list of options and pressing Return to select.

Mini Office II takes over where the original, highly successful Mini Office left off, with numerous extra features, two additional modules, a program to convert existing Mini Office files to Mini Office II format, and a 60 page, easy to follow manual.

And it's at a price everyone can afford!

B, B+	Tape	£14.95
	5.25" disc	£16.95
Master	5.25" disc	£19.95
Master Compact	3.5" disc	£21.95

B, B+, Master,	4 by 32k Rom	
Master Compact	board	£59.95

6 powerful home and business programs in just one package – at a price that simply can't be matched!

- **WORD PROCESSOR:** Compose a letter, set the print-out options using embedded commands or menus, use the mail merge facility to produce personalised circulars – and more!
- **DATABASE:** Build up a versatile card index, use the flexible print out routine, do powerful multi-field sorting, perform all arithmetic functions, link with the word processor – and more!
- **LABEL PRINTER:** Design the layout of a label with the easy-to-use editor, select label size and sheet format, read in database files, print out in any quantity – and more!
- **SPREADSHEET:** Prepare budgets or tables, total columns or rows with ease, copy formulae absolutely or relatively, view in either 40 or 80 column modes, recalculate automatically – and more!
- **GRAPHICS:** Enter data directly or load data from the spreadsheet, produce pie charts, display bar charts side by side or stacked, overlay line graphs – and more!
- **COMMS MODULE:** Using a modem you can access services such as MicroLink and book rail or theatre tickets, send electronic mail, telex and telemessages in a flash – and more!

INSTANT Mini Office II

Now available on a power-packed rom board!

- Four 32k roms on one board packed with 128k of super fast machine code.
- Easy to install – no soldering needed.
- Split second application selection.
- Frees valuable ram space for much more data.
- Fully compatible with the whole BBC Micro range, Aries or Watford shadow ram board, and tape, DFS, ADFS and ECONET filing systems.

The full selection of Mini Office II is available through BEEBUG Retail at MEMBER'S DISCOUNT. Please refer to your BEEBUG Retail Catalogue

DATABASE SOFTWARE

Europa House, 68 Chester Road,
Hazel Grove, Stockport SK7 5NY

Personal Ads

"Elite" £7, BEEBUG's Quickcalc and Spellcheck I discs £5 each, Beebug's Toolkit I ROM £15, Watford monitor and disc drive double level stand £12. Data Base's Fun School 80T educational disc, Gemini's Beebplot tape, Acorn's Desk diary/planner tape - all for any reasonable offer. All items boxed with manuals. Upgrading forces sale. Tel. Harlow 37527.

WANTED Micro Aid Family History System. FOR SALE Hershey Characters Font Generator (2x80T discs) £7.50. Merlin Data Base (ROM + disc) £25. Normende high res. 12" green screen monitor £25. Poolswinner & Fixgen 1987/88 (tape) £7.50 or very near offer considered. Tel. (0705) 371018.

Tapes (£2 each) - System 15000, Brian Jacks Superstar Challenge, 3D Bomb Alley, Test Match Cricket/World Cup (Electron) + other games. Beebug (Vol.2 No.10, Vol.3 No.7) £1 each. TD ROM £10. Discmaster £10. Write to Simon Howard, 9 Springmeadow Lane, Uppermill, Oldham, Lancs OL3 6EP (enclose SAE).

WANTED educational and adventure game software for Master 128, suitable for novice age 7. Tel. Plymouth (0752) 778062

BBC Micro, Twin 100k Disc Drives, Aries B20 Shadow RAM, Microvitec Colour Monitor, Wordwise, masses of original software + books, at least 100 discs (some blank) + much more. Original cost £2000+, £650 o.n.o. Tel. (0277) 227652 after 6.30 pm.

Electron Peripherals c/w manuals/boxes, Plus 3 £120, 27x3.5" discs £20, Plus 1 £35, ROM Cart. £8, T2P3 £15, Toolkit £10, everything for £190. Also E2P £40, Electron £40. £260 the lot. Tel. Bristol (0272) 832983.

Shugart 40T double sided drive in original box with manual and formatting disc, £50. Also almost complete run of Acorn User to 1985 and Micro User from start to 1985, best offer for either collections. Tel. 01-953 9163.

Bargain BBC B issue 7, Watford DFS, Solidisc 4MEG, 256k Sideways ROM/RAM board. Much software including View 2.1, Iconmaster, Mega ROM + Manager ROM and lots of original games inc. Impossible Mission, Repton 3, Graham Gooch Cricket. Offers accepted. Tel. (0462) 50445, after 4.30 p.m.

Linear Graphics Plotmate A4 mint condition, sole reason for sale upgrade to A3. Complete with manual, ROM, pens, welcome disc etc. Offers to (0252) 24930.

Morley Teletext Adaptor with ATS ROM, utils disc and power supply unit, £60. Tel. 01-952 7244.

Acorn Electron computer, Plus 1, recorder, joystick and over 100 original top software titles, including View and ViewSheet ROMs. Many books, all manuals and leads complete. Almost new. Excellent bargain at £130. Tel. Bradford (0274) 672264.

Watford DDFS kit and ROM, no fitting instructions, £35. Tel. (0375) 370007, after 4.30 p.m. and weekends.

Acorn Z80 second processor with extra software, boxed £130. Watford ZIF socket unused £10. Vine micros master RAM write protect switch £4. Tel. 01-455 1069 eves.

BBC B series 7 1.2 O.S., Solidisc DDFS + 1770 and ADFS ROMs, 80T DS DD (1 Mb) disc drive, MP 165 printer, joysticks (Kempston and Delta 14b), SWR 32. All manuals and many books for BBC inc. complete set of BEEBUG mags and Acorn User from Mar. 1985. ROMs, discs and tapes, also cassette recorder. Black upright computer desk with shelves and draws, £800 ono. Tel. (093287) 3435 eves.

17 Originals on tapes for BBC B including Elite, Frak, the Music System tapes 1 and 2. Excellent condition, all documentation included. Starting from £2 each, £45 the lot (cost around £150 new). Tel. 01-427 2937, after 5 p.m.

Nightingale Modem, little used, £50. John Crabtree, 10 Pathfields, Dartmouth, Devon TQ6 9HL.

BBC B computer, issue 7, basic model in original box, £130. Tel. (0628) 26076.

Time Warp, Murom, Office Master and Mate. Offers accepted. Tel. (0923) 22548.

BBC MICRO USER in Portugal requires local contacts for mutual support. Please write in Portuguese or English to Daniel Ward dos Santos, Rua dos Ferroviarios No.15, 8375 S. Marcos de Serra, Algarve.

BBC B with DNFS and APTL ROM board (16k RAM), Microvitec lo-res monitor, Prism modem 1000 + Watford "Modem 84" ROM, View 3.0, Wordwise Plus, Disc Doctor, BEEBUG Toolkit. Software including various Acornsoft games, other games, Forth WordEase, SpellCheck, blank discs, £460 ono, will split. Tel. (0454) 323081.

BBC Master 128 in new/mint condition. Sensible price offered. Also hi-res monitor, colour or monitor. Also EMR Performer disc for test. Tel. Lincoln (0552) 752458.

→ 40

NEW TECHNICAL REFERENCE
GUIDE NOW AVAILABLE!
£12
Please contact us for further details.

SYSTEM Software

Dept. G
Fifth Floor Sheaf House
Sheaf Street
Sheffield S1 2BP
Tel. (0742) 768682

Access and Barclaycard welcome
Further information available

At last a machine code development system that *really* does outperform ADE

ADE+

Features

- Runs on BBC B, B+, Master, Compact, ADFS, DFS, ANFS, NFS
- Full use of all available RAM
- Intelligent memory management unit (MMU)
- User switches on all main features
- Automatic search for macros on disc
- Assemble absolute or relocatable code
- Source program tokenised or full text
- REPEAT..UNTIL, WHILE..WEND etc. high level language constructs
- Macros nestable to any depth
- Excellent error diagnostics
- Linker
- Editor and symbolic disassembler provided

A comparison of three 65C12 assemblers

Feature	ADE +	BBC Basic	MACROM
Number of pseudo-ops	64	4	36
Use of all available RAM	Yes	No	No
Macros	Yes	No	Yes
String functions	Yes	Yes	No
True macro libraries	Yes	No	No
Relocatable output	Yes	No	No
Linker with libraries	Yes	No	No
High level constructs	Yes	Yes	No
Use any editor?	Yes	No	No
Number of error reports	39+ warnings	3+ BASIC errs	20
65C00 extended opcodes	All	65C12 only	65C12 only
Switch off 65C12 opcodes	Yes	No	No
ROM size	32K	16K	16K
Disassembler	Symbolic	No	Simple
Label restrictions	No	Yes	Yes

ADE+

Versions and prices

Recommended for Master Compact...

ADE+ MMU and 65C00 series assembler on disc (3.5" ADFS). 32K sideways RAM required.

£42.00 + vat

Recommended for BBC B, B+...

ADE+ MMU and 65C00 series assembler on 2 16K EPROMs with DFS 5.25" utilities disc.

£46.00 + vat

Recommended for Master 128, turbo...

ADE+ MMU and 65C00 series assembler on EPROM cartridge with 5.25" DFS utility disc.

£49.00 + vat

Upgrade

ADE to ADE+ (upgrade to either disc, EPROM or cartridge, you must send in ADE ROM chip)

£27.00 + vat

Upgrade

ASM to ADE+ (upgrade to either disc, EPROM or cartridge, you must send in ASM ROM chip)

£35.00 + vat

Please add £1.25 P&P per unit.

ADE+

The Ultimate Assembly Language Development Tool

ADE+ is a 65C00 series assembler system supporting all the mnemonics of the 65C12 used in the latest BBC microcomputers plus the additional 'Rockwell' instructions. ADE+ is fast, faster in fact than the in-built BASIC assembler and all rival products that we have tested. The assembler produces absolute code that can be "RUN or linker modules that can be merged with the output from other programs using the ADE+ linker. ADE+ supports a powerful linker which drastically cuts assembly time; a feature normally only found on minis and mainframes. The linker will even link the output from compilers with your assembly language programs. Full library support for both the linker and the assembler is provided - fast searching for unknown instructions in a random access macro library. ADE+ is a modular system with many modules to add later; i.e. a mouse based editor & a Z80 cross assembler! A print spooling system uses sideways RAM as a print buffer to eliminate waiting time; your listing runs off as a background job! Use the print spooler from BASIC or your own programs. ADE+ uses ALL available memory. With a second processor attached the IO processor spare memory is used as a buffer to reduce the amount of disc access. All available memory is handled by ADE+'s intelligent memory management module. Use your own favourite editor or the one provided. Assemble from disc or memory. Full utilities including librarians, converter for BBC BASIC etc. ADE+ must be the bargain of 1986/7!

RISC USER

The Archimedes Support Group

After a very careful look at the Archimedes, we have decided that we cannot possibly do full justice to the vast potential of the Archimedes range within the existing Beebug format. We would either end up starving the Archimedes of vital coverage, or would be taking too many pages away from the model B and Master machines, whose own potential is very far from exhausted. We have therefore launched a sister user group and magazine for the new machine.

Existing Beebug members may either transfer their membership to the new magazine or, for only an extra 60p per issue, extend their subscription to include both magazines. A joint subscription will enable you to keep completely up-to-date with all innovations and the latest from Acorn and other suppliers on the complete range of BBC micros. **RISC User** has a massive amount to offer, particularly at this time while documentation about the Archimedes is of such a limited nature.

Here are some of the topics covered in the first two issues of **RISC User**. Issue one is already out (and will be sent to all new subscribers) and issue two is due out within a couple of weeks:

ARM ADFS MENU - Keep this short machine code program resident in your Archimedes for rapid access to all files and directories.

ARCHIE'S WHITE MOUSE - How to use the mouse from Basic, and how to incorporate mouse and pointer into your own applications.

VISUAL EFFECTS - Some short Basic routines producing stunning visual effects.

SOUND SET-UP - How to use the Archimedes sound system - with information on incorporating the extra instrument files on the Welcome Disc.

OFF-THE-SHELF - A look at some of the products available now for the Archimedes - including Wordwise Plus, Deltabase, ANSI C, Clare's Toolkit, and Acorn's Twin Editor.

ACORN HOTLINE - News of the new operating system release, and of software and hardware due for imminent release.

HINTS & TIPS - Dozens of vital hints and tips covering undocumented features of the new machine, short cuts, bugs, and errors in the manuals.

Plus much more including a look at Aliases, how to use the Basic Editor and Emulator, and articles on Basic Libraries, Configuring the Archimedes, screen printer dumps, and using the Archimedes disc system.

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

As a member of BEEBUG you may extend your subscription to include the first year of **RISC User** for only £6.00 (*overseas see below) - less than HALF PRICE! The initial introductory subscription to **RISC User** is £12.50 to the

Name:..... Memb No:..... Address:	public, and the full rate from early 1988 will be £14.50.	SUBSCRIPTION DETAILS	
		Destination	Additional cost
		UK, BFPO & Ch Is	£6.00
		Rest of Europe and Eire	£8.00
		Middle East	£9.00
		Americas and Africa	£9.50

I wish to receive both BEEBUG and **RISC User**.

I enclose a cheque for £

Alternatively I authorise you to debit my ACCESS/Visa/Connect account /_____/_____/_____/_____/_____/_____/

Signed:

Expiry Date /_____/_____/

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

UPDATE

MEMBERS' DISCOUNT

MRH Software are offering Miscue Analysis (reviewed BEEBUG Education Vol.6 No.3) at 15% discount to BEEBUG members when ordered direct from MRH. The address is MRH Systems & Software, 20 Highfield Road, Kidderminster, Worcestershire DY10 2TL.

USER GROUPS

VICTORIA, Australia

VICBUG (the Victorian Acorn User's Group) may be contacted through Ted Robinson, 31 Curtin Ave West Brunswick 3055, Victoria, Australia or telephone (03) 386 7529).

WASHINGTON, Tyne & Wear

Newbug can now be contacted c/o N.Kidd, Multi-Purpose Centre, Ox Close Village, Washington, Tyne & Wear (tel. 051 4166407).

SPECIAL OFFER POLAROID PALETTE FOR THE B.B.C. MICRO

System includes
Autoprocessor,
Cutter Mounter,
Sample Films.

*The simple way to
instant 35mm
presentation
slides, prints
and overhead
transparencies.*



To take advantage of this limited period offer please send your order to: Robin Cartwright Polaroid (U.K.) Ltd., Ashley Road, St. Albans, Hertfordshire AL1 5PR.

- ☐ Please debit my Polaroid Account No:.....
☐ Cheque for full order value enclosed (£563.50 inc VAT).....
☐ Please debit my Access Account No:.....
Barclaycard Account No:.....

Name Address
..... Signature

ELECTRIC PROCESSOR/ILLUMINATED CUTTER MOUNTER £50 EXTRA (+ VAT)

 **Polaroid**

Polaroid is a registered trademark of Polaroid Corporation, Cambridge, Mass., USA

ADVERTISING IN BEEBUG

For advertising details, please contact
Yolanda Turuelo

on
(0727) 40303

or write to
Dolphin Place, Holywell Hill, St Albans, Herts. AL1
1EX

UK BULLETIN BOARDS

Continued from Vol.6 No.5

Maptel 24 Hour	0702 552941 (Essex) 300/300	Norview 24 Hours	0604 20441 (Northampton) 1200/75 Viewdata
Marctel 24 Hours	01 346 7150 (London) 300/300	OBBS Bradford WD 18.00-00.00	0274 480452 (Bradford) 300/300 & 1200/75
MBBS Leconfield 24 Hours	0401 50745 (Beverley) 300/300	OBBS Manchester 24 Hours	061 427 1596 (Manchester) 300/300 & 1200/75
MBBS Mitcham 24 Hours	01 648 0018 1200/75 & 300/300	Octopus 18.00-08.30	0272 421196 (Bristol) 300/300
Metrotel 24 Hours	01 941 4285 (London) 1200/75 Viewdata	OSI Lives! 24 Hours	01 429 3047 (London) 300/300
MG-NET Sun 17.00-22.00	01 399 2136 300/300	Owltel 24 Hours	01 927 5820 (London) 1200/75 Viewdata
MirrorWorld 24hrs	0483-844044 (Guildford) 1200	PBBS-NI 22.00-00.00	0762 333872 (N.Ireland) 300/300
Musitel Plus 24 Hours ÄPE NBBS Cheshire 24 Hours	0843 590000 300/300 & 1200/75 0936 77025 (Cheshire) 1200/75 & 300/300	Pete's Place 24 Hours	0206 862354 (Essex) 300/300
NBBS Essex 24 Hours	0277 228867 (Essex) 1200/75 & 300/300	PIP 24 Hours	0742 667983 (Sheffield) 300/300
NBBS London 21.00-08.00 WE 24 Hrs	01 883 5290 (London) 1200/75 & 300/300	ÄPE PSBBS 18.00-23.00	0443 733343 (Ferndale) 300/300
NNBBS London 24 Hours	01 455 6607 (London) 1200/75 & 300/300	React 24 Hours	0376 518818 (Essex) 300/300

To be continued

Please notify us of any changes or omissions to the information published here.

KEEP AHEAD IN COMPUTING WITH CHAPMAN AND HALL MICROCOMPUTER GRAPHICS

Michael Batty

Professor Batty provides a comprehensive treatment of graphics explaining all the technical methods used in its professional study, but makes these accessible to beginners. All the programs are written in BBC BASIC and the book contains many black and white/colour plates of the range of designs
February 1987 344pp, illus Hb 0 412 28530 4 £35.00 Pb 0 412 28540 1 £14.95

PRACTICAL INTERFACING WITH THE BBC MICRO

Geoff Bains

This book shows how, with a little ingenuity and a few pieces of inexpensive equipment, a BBC Micro can be made to control a whole range of devices to be found in the home or school laboratory. The author also explains the underlying principles—encouraging further experimentation.

1986 212pp Pb 0 412 27320 9 £7.95

COMPUTING WITHOUT PROGRAMMING

A guide to software packages on the BBC Microcomputer

Judith Citron

Demonstrates how a student or teacher, without any knowledge of computer programming, can implement projects on the BBC Microcomputer.

February 1987 268pp Pb 0 412 28160 0 £9.50



CHAPMAN AND HALL

11 New Fetter Lane, London EC4P 4EE

Business Ads

OMNIROM - the ultimate Utility ROM for the MASTER and Compact. Adds three new CONFIGURATION Commands - FOREGROUND COLOUR, BACKGROUND COLOUR, and FONT which integrates 3 new Fonts. ADFS Menu System, plus many other utilities, such as 65C12 Disassembler, Memory and Disc editors, BACKUP and VERIFY, BASIC STATUS and much, much more. Supplied on 16K ROM. Send £12.95 (incl. p&p) to **NUMBER 3 SOFTWARE, 3 Dairy Farm Court, Attleborough, Norfolk NR17 2BT.**

MULTIPROG - Store up to 200 Basic programs on one disc! Filenames up to 30 characters long! Load, Save, Delete and Index functions available through f-keys. For use with Acorn DFS or compatible £10. Send SAE for details to

M. French, Moorcroft, Green Lane, Crowborough, E. Sussex TN6 2DG.

MARKS AND STATISTICS for handling students marks. Widely used in schools, colleges. Spreadsheet format, calculates totals, sorts, analyses, normalises etc. Prints lists, histograms. BBC B, Master. 40-80 track. £17. *In-Form, 73 Woodfield Park, Colinton, Edinburgh EH13 0RA.*

"SHAREMATIC" software package for BBC Master 128. Automatically updates all 170 shares from BBC Ceefax service each day, Plots Graphs, Point & Figure, & Oscillator and more. £35 or SAE for brochure to: *Citymagic, Dept BB2, 40 Manor Road, Goldington, Bedford MK41 9LQ. Tel. (0234) 856050/67067.*

Watford 32k RAM/ROM board, expandable to 128k, fitted 16k CMOS RAM - see BEEBUG Vol.5 No.6, £25, also RAM AMP 6 ROM extension board £10. All manuals supplied. Tel. (0932) 226076.

Inter-base (Computer Concepts), £45. ROMIT (BEEBUG) £15 and an Acorn/Shugart full-height disc drive, £40. All complete and boxed with manuals. Tel. Southampton (0703) 845104.

Spellcheck II, boxed with dictionary disc and manual, £10. Tel. Stalbridge, Dorset (0963) 63497.

Brother HR10 daisywheel printer and TF10 tractor feed, six months old, boxed, £90 ono. Tel. 01-854 4932 eves, 01-921 2514 daytime.

BEEBUG magazines from Vol.1 No.1 up to date. All perfect and complete. Best offer secures. Tel. Portreath (0209) 842737.

BBC Master Compact one year old plus 15 discs, RS232, printer lead and external drive lead. Little used £325. Upgrading to an Archimedes. Tel. (0843) 584222.

Original software in mint condition including all manuals and documentation of ISO-Pascal (£24), Logotron Logo (£24), Wordwise Plus (£24) and Disc Doctor (£8). All for £55. Tel. (0273) 682240.

Novocad £35, Plotter driver £10, AMX Design £25, System ADE £6, Printmaster £6, the Music System £6. All manuals, boxed, as new. Tel. 01-500 5701.

Centronics 701 printer 132 column with cable & spare ribbon packs, £50 buyer collects. Tel. Nottingham (0602) 654426 eves.

BBC model B, Acorn DNFS disc interface, 128K SOLIDISC sideways RAM, two additional ROM sockets and software. DIN sound output and extension speaker sockets, write protect and composite colour/mono monitor switches, standard and advanced User Guides in hard covers. Price £300 ono Tel. Penzance (0736) 763549.

Viglen 40/80T switchable cased disc drive £50. BASF 40T DS disc drive, £50. Teac twin 40T DS disc drives in one casing, £100. Will swap 2 disc drives for Acorn Teletext adapter + ROM + PSU. Case 400/22b Professional modem (2400/1200 baud) ideal for new BT Gold, fully software controllable/hard switched control. Tel. Dinnington (0909) 565257.

BBC B issue 7 with ATPL ROM board, Floppywise ROM. £250. Wordwise Plus and Bruce Smith's Book £25, Watford ROMSPELL £18, Watford 32K Shadow RAM £45. All as new condition. Tel. (0234) 750050.

High resolution green screen composite video monitor with leads, also Teletype serial printer and leads. Tel. Stirling (0786) 85633 after 5pm.

For sale **Wordwise Plus ROM, Manuals, etc.**, £30. System Delta ROM, Card Index and Reference Manual £50. Tel. (04203) 5311 eves.

Viglen Disc Drive, 40 track, SS 100K, hardly used, £50. BBC Micro User Guide, Adv Disk User Guide, 6502 Assembly Lang Programming (Leventhal), BBC Elite, Quondam Adventure, Acheton User Guide £5. Upgrade to Blandford (0258) 52275.

For sale (or exchange) over **20 original games** for BBC B. Will swap on a 1 to 1 basis also. Tel. (0704) 37821.

Watford 32K shadow RAM board £40, Watford solderless ROM board + 16K Sideways RAM £25, original packaging and documentation. New keyboard (BBC B) £30, Acorn Basic II RO £10, Advanced User Guide £5. Upgrade to Master, hence sale. Tel. 01-380 9319 (Days).

BBC Software for sale. Mainly cassettes, few discs. Send SAE for list to J. Prinner, 10 Moor Park, Millon, Cumbria LA18 5DX.

AMS Super Art ROM disc and utility disc including manual and mouse for the Master 128, £25. Wordwise Plus (Master compatible) with Hi-Word disc, manuals, and cassette typing tutor and examples £25. PSU for single disc drive with 13 amp plug and plug for unit £10. Prefer purchaser to collect. Tel. 021-422 7068.

Juki 6100 daisywheel printer, mint condition, boxed with instructions and BBC connecting cable, Wordprocessor FOC to purchaser, £200. Tel. (0636) 813847 - no offers please.

BBC B discs: Mini Office II (80T) £10, Replica III £5. BBC B cassettes: Mini Office, Kansas word processor, Typeasy, Micro-Aid Memo-Calc, and French Abroad, all with instructions in original packing £3 each. Tel. (0403) 55400.

Vine Micros Master Replay £25, Master Overlay board £15, or both for £35. Advanced Computer Products Advanced 1770 DFS for BBC B £15, Advanced 1770 DFS for Master 128 £15. Mini Office II 80T (BBC) £9. Printwise 80T (Master/BBC B) £15. Tel. 01-831 2735 office hours only.

Interbase ROM and manual, etc, £40. Tel. (0903) 755412.

Acorn AP100 dot matrix printer £50. Watford ROM/RAM expansion card £15. Watford 32K RAM expansion card £30. Wordwise Plus and Bruce Smith's book £20. Computer Concepts TERMI £15. Computer Concepts Disc Doctor £15. Beebugsoft Toolkit £15. Tel. 01-328 0707.

In addition, we have more hints and tips for the Master and Compact.



Storing Variables and Procedures in SWR

If you are going to use this technique in your own programs, the following points should prove useful. Firstly, note that line 110 sets LOMEM. This must be set so that it will always be above the program plus the longest overlay. PROCinit sets the SWR for data use, and as currently supplied, it uses just banks 4 and 5. To make it use all four, alter the "5" in line 1020 to "7". Line 1030 stores the value of TOP in T%. This is used to tell the


```

10000 :
10010 REM *****
10020 :
10030 REM "CHART"
10040 :
10050 DEFPROCchart
10060 PRINT'"CHART OVERLAY"'
10070 ENDPROC

```

Save this file as "CHART"

```

10000 :
10010 REM *****
10020 :
10030 REM "EDITOR"
10040 :
10050 DEFPROCeditor
10060 PRINT'"EDITOR OVERLAY"'
10070 ENDPROC

```

Save this file as "EDITOR"

```

10000 :
10010 REM *****
10020 :
10030 REM "FILER"
10040 :
10050 DEFPROCfiler
10060 PRINT'"FILER OVERLAY"'
10070 ENDPROC

```

Save this file as "FILER"

Overlay Modules

program where in memory to load the modules. It copes with any length of program, and needs no customising. You will however, need to alter the number of overlays in line 1050, and the names of the program files on disc in which they are held (line 1110), so as to suit your own application.

```

LOADING CHART (100 bytes) INTO SWR AT ADDRESS 0
LOADING FILER (100 bytes) INTO SWR AT ADDRESS 100
LOADING EDITOR (103 bytes) INTO SWR AT ADDRESS 200
3 overlays now in SWR. Press any key for demo...

```

PROCloadswr on line 1140 loads all the modules into SWR, and again needs no customising. PROCmenu then allows the user to choose which procedure to execute, and this would be replaced by your own routines. The only thing to remember is that on exit from PROCmenu, the variable G% holds the

reference number of the procedure. In this case G% is set to 1, 2 or 3, depending upon the choice made by the user.

PROCappend is then called. This appends the procedure whose reference number is held in G%. Again this procedure needs no customising. Finally PROCexecute is called. This routine uses G% to determine which procedure should be called. If G%=1, PROCchart is called, if G%=2, Procfiler is called, and so on. Obviously the procedure calls made in PROCexecute will need to be altered to match those used in your own overlays.

Lastly, you will see that when option 4 is selected, to exit the program, two POKEs are made to memory in line 1320. These serve to erase the currently loaded module when the program is terminated, so that the program may, if the user wishes, be saved in its original form even after execution.

```

10 REM Program SWR Pcedure overlays
20 REM Version B 0.3
30 REM Author Graham Crow
40 REM BEEBUG November 1987
50 REM Program subject to copyright
60 :
100 MODE128
110 LOMEM=64000
120 REM LOMEM must be > top of control
130 REM program + longest overlay
140 PROCinit
150 PROCloadSWR
160 REPEAT
170 PROCmenu
180 PROCappend
190 PROCexecute
200 UNTIL FALSE
210 :
1000 DEFPROCinit
1010 REM SET UP banks of SWR for data
1020 FOR J%=ASC"4" TO ASC"5":OSCLI "SRD
ATA "+CHR$ J%:NEXT
1030 T%=TOP:REM Procs load at T%-2
1040 S%=0:REM SWR Address for overlays
1050 N%=3:REM No of overlays
1060 DIM F$(N%):REM Overlay filenames
1070 DIM L$(N%):REM Overlay lengths
1080 DIM R$(N%):REM SWR Addresses
1090 RESTORE 1110
1100 FOR J%=1 TO N%:READ F$(J%):NEXT
1110 DATA CHART,FILER,EDITOR

```



```

1120 ENDPROC
1130 :
1140 DEFPROCloadSWR
1150 REM Loads overlays into SWR
1160 FOR J%=1 TO N%:OSCLI("LOAD "+F$(J%
)+" "+STR$(T%))
1170 Z%=OPENUP F$(J%)
1180 L$(J%)=EXT#Z%:CLOSE#Z%
1190 PRINT"LOADING "F$(J%) " (" ;L$(J%);
" bytes) INTO SWR AT ADDRESS ";S%
1200 OSCLI("SRWRITE FFFF"+STR$(T%)+ "+"
STR$(L$(J%))+" "+STR$(S%))
1210 R$(J%)=S%:S%=S%+L$(J%)
1220 NEXT
1230 PRINT";N%," overlays now in SWR.
Press any key for demo...";:IFGET
1240 ENDPROC
1250 :
1260 DEFPROCmenu
1270 CLS:PRINT""MAIN MENU"
1280 PRINT"1...CHART""2...FILER"
1290 PRINT"3...EDITOR""4...END""Which?
";
1300 REPEAT G%=GET-48
1310 UNTIL G%>0 AND G%<5
1320 IF G%=4 CLS:?(T%-2)=&D:?(T%-1)=&FF
:END
1330 ENDPROC
1340 :
1350 DEFPROCappend
1360 REM Appends overlay at TOP-2
1370 len%=L$(G%):ad%=R$(G%)
1380 OSCLI("SRREAD FFFF"+STR$(T%-2)+"
"+STR$(len%)+ " "+STR$(ad%):ENDPROC
1390 :
1400 DEFPROCexecute:CLS
1410 IF G%=1 PROCchart
1420 IF G%=2 PROCfiler
1430 IF G%=3 PROCeditor
1440 PRINT"Press any key ";:IF GET
1450 ENDPROC

```

VARIABLE STORAGE

The second listing demonstrates a slight variant on the technique. This time SWR is used to store and retrieve variables of any kind. If you type in the program and run it, you will see a display similar to that in the accompanying figure. What the program does is to set up two banks of SWR for data storage. It then passes variables of the five different types shown, to the SWR, and reads them back to the screen.

If you take a look at the program you will see that there are five short procedures (starting at

line 1320), and five functions (starting at line 1610) which handle the saving and loading of variables. Each variable type has a dedicated procedure for storing the variable, and a function for retrieving it; and you may if you wish use only those procedure-function pairs appropriate to your needs, with the proviso that the double-byte procedure and function both call their single-byte counterparts.

DEMONSTRATION of WRITE/READ SIDEWAYS RAM

```

Byte:      &E
Double-byte: &FE99
Integer:    1234567
Real:      -1234.567
String:    Any string
>

```

Apart from this, all you need is line 110 which assigns the variable A at the very start of the program, together with the procedure PROCinit. It is essential for the correct operation of the program that the assignment of the variable A occurs before any other variable, procedure or function is defined. The procedure PROCinit at line 1000 serves three purposes. First of all it initialises the SWR for data storage. As listed, the program reserves banks 4 and 5 only, but you may alter this by changing line 1020. It then assigns the address of the variable A to the resident integer variable A%, and this will be used for passing data to and from SWR. Finally an area is set up for transferring string variables. The variable max% should be set to a value greater than the length of the longest string to be transferred between SWR and user RAM.

The demonstration routine is held in the procedure PROCdemo, defined at line 1140. This simply calls each procedure in turn to save a variable of the appropriate type, then it calls the five functions to read back the values stored, and print them out. A closer look at this procedure illustrates the way each of the procedures and functions are handled. Each procedure is called with two parameters. The first is the variable to be stored, and the second is a memory location in SWR. When recalling any variable, the corresponding function is

called with just a single parameter: namely the address in SWR at which the variable was stored. In this respect all variables are treated very much like Basic's indirection operators treat data. But although the problem of memory management is left to the user, just as it is when using the indirection operators, it is a relatively small price to pay for the very large resultant increase in available user RAM.

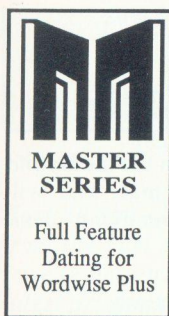
```

10 REM Program SWR variable storage
20 REM Version B 0.7
30 REM Author   Graham Crow
40 REM BEEBUG   November 1987
50 REM Program subject to copyright
60 :
100 MODE 128
110 A=0 :REM Assign A at start
120 PROCinit
130 PROCdemo
140 END
150 :
1000 DEFPROCinit
1010 REM initialsie SWR
1020 FOR J%=ASC"4" TO ASC"5"
1030 OSCLI "SRDATA "+CHR$ J%:NEXT
1040 :
1050 REM Assign address of variable A
1060 REM to some variable eg A%
1070 A%=LOMEM+3
1080 :
1090 REM Reserve string storage space
1100 REM Must be longest string + 1
1110 max%=100:DIM M% max%
1120 ENDPROC
1130 :
1140 DEFPROCdemo
1150 PRINT"DEMONSTRATION of WRITE/READ
SIDEWAYS RAM"
1160 :
1170 REM Write data to SWR
1180 PROCb(&E,&0) :REM Single byte
1190 PROCbb(&FE99,&1000) :REM Two byte
1200 PROCi(1234567,&2000) :REM Integ
1210 PROCr(-1234.567,&3000) :REM Real
1220 PROCs("Any string",&4000) :REM Str
1230 :
1240 REM Read data from SWR
1250 PRINT"Byte:"TAB(13)"&";~FNb(0)
1260 PRINT"Double-byte: &";~FNbb(&1000)
1270 PRINT"Integer:"TAB(13);FNI(&2000)
1280 PRINT"Real:"TAB(13);FNR(&3000)
1290 PRINT"String:"TAB(13)FNs(&4000)
1300 ENDPROC
1310 :
1320 DEFPROCb(num%,ad%) :REM Write byte
1330 ?A%=num%
```

```

1340 OSCLI "SRWRITE FFFF"+STR$~A%+" +1"
+" "+STR$~ad%
1350 ENDPROC
1360 :
1370 DEFPROCbb(num%,ad%):REM Write 2byt
1380 LOCAL b1%,b2%
1390 b1%=num%DIV256:b2%=num%MOD256
1400 PROCb(b1%,ad%):PROCb(b2%,ad%+1)
1410 ENDPROC
1420 :
1430 DEFPROCi(num%,ad%) :REM Write int
1440 !A%=num%
1450 OSCLI "SRWRITE FFFF"+STR$~A%+" +4"
+" "+STR$~ad%
1460 ENDPROC
1470 :
1480 DEFPROCr(num%,ad%) :REM Write real
1490 A=num
1500 OSCLI "SRWRITE FFFF"+STR$~A%+" +5"
+" "+STR$~ad%
1510 ENDPROC
1520 :
1530 DEFPROCcs(str$,ad%) :REM write str
1540 LOCAL len%
1550 $M%=str$
1560 len%=LEN(str$)+1:PROCb(len%,ad%)
1570 ad%=ad%+1
1580 OSCLI "SRWRITE FFFF"+STR$~M%+" "+
STR$~len%+" "+STR$~ad%
1590 ENDPROC
1600 :
1610 DEFFNb(ad%) :REM Read byte
1620 OSCLI "SRREAD FFFF"+STR$~A%+" +1"+
" "+STR$~ad%
1630 =?A%
1640 :
1650 DEFFNbB(ad%) :REM read 2 byte
1660 LOCAL b1%,b2%
1670 b1%=FNb(ad%)*256:b2%=FNb(ad%+1)
1680 =b1%+b2%
1690 :
1700 DEFFNi(ad%) :REM Read integer
1710 OSCLI "SRREAD FFFF"+STR$~A%+" +4"+
" "+STR$~ad%
1720 =!A%
1730 :
1740 DEFFNr(ad%) :REM Read real
1750 OSCLI "SRREAD FFFF"+STR$~A%+" +5"+
" "+STR$~ad%
1760 =A
1770 :
1780 DEFFNs(ad%) :REM Read string
1790 LOCAL len%
1800 len%=FNb(ad%):ad%=ad%+1
1810 OSCLI "SRREAD FFFF"+STR$~M%+" "+S
TR$~len%+" "+STR$~ad%
1820 =$M%
```

B



Brian Kemp presents a Wordwise Plus segment program which will insert a full version of the current date into any Wordwise text file.

Unlike View, Wordwise does not have an inbuilt facility to read the Master's clock. But we can easily implement such a feature using the word processor's flexible programming language. And as an added bonus, we can make its representation a little smarter than the output of the Master's clock normally allows.

The accompanying segment program should be typed into Wordwise, and saved away. To make use of it, just load it into any segment (say segment zero), then just press Shift-f0, and the date will automatically be inserted into your text at the cursor position. It should look something like:

14th. October 1987

with the "th", "st" or "nd" being supplied as appropriate, and with the name of the month appearing in full.

Alternatively, you can delete the last line of the segment program, and alter the penultimate line to read:

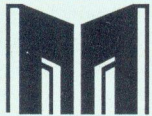
D\$=D\$+" "+M\$+" "+Y\$

and include in your text the sequence:

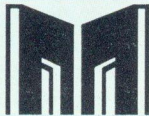
<f1>PSD\$<f2>

where <f1> means press function key f1 to produce a green code. This will print the string D\$ (which holds the formatted date) at the correct position in the text in both Print and Preview modes (i.e. options 6 or 7), and providing that the segment program has been run at some point previously, the correct date will be inserted.

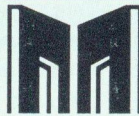
```
B%=0
A%=14
X%=&08
Y%=&04
CALL &FFFF1
D$=""
X%=&40C
REPEAT
D$=D$+CHR$?X%
X%=X%+1
UNTIL X%=&40E
M$=""
X%=&40F
REPEAT
M$=M$+CHR$?X%
X%=X%+1
UNTIL X%=&412
Y$=""
X%=&413
REPEAT
Y$=Y$+CHR$?X%
X%=X%+1
UNTIL X%=&417
.day
D%=VALD$
D$=STR$D%
IF D%=1 THEN GOTO first
IF D%=21 THEN GOTO first
IF D%=31 THEN GOTO first
IF D%=2 THEN GOTO second
IF D%=22 THEN GOTO second
D$=D$+"th."
GOTO month
.first
D$=D$+"st."
GOTO month
.second
D$=D$+"nd."
GOTO month
.month
IF M$="Jan" THEN M$="January"
IF M$="Feb" THEN M$="February"
IF M$="Mar" THEN M$="March"
IF M$="Apr" THEN M$="April"
IF M$="Jun" THEN M$="June"
IF M$="Jul" THEN M$="July"
IF M$="Aug" THEN M$="August"
IF M$="Sep" THEN M$="September"
IF M$="Oct" THEN M$="October"
IF M$="Nov" THEN M$="November"
IF M$="Dec" THEN M$="December"
.final
TYPE D$+" "+M$+" "+Y$
DISPLAY
```

Hints



Hints



Hints

BETTER BLANKING

Max Christian

Earlier hints on blanking EXEC files as they are executed have all suffered from snags of one kind or another. May I propose the use of VDU21 to blank the display and VDU6 to reinstate it. It has the great advantage that it will work in any mode, and does not move the cursor down the screen. To test it out, try putting the following into an EXEC file:

```
VDU21
```

```
VDU6:PRINT"This is an EXEC file."
```

When this is EXECed in it will print "This is an EXEC file", and the only instruction shown on the screen will be "VDU21".

INTERWORD AND SPELLMASTER

George Foot

If you use a !BOOT file to load a Spellmaster user dictionary, you may have noticed that the boot file will not boot up a second time without first switching off the machine. This is because the *DLOAD command sets up the dictionary in sideways RAM as if it were a ROM image. In Basic, an attempt to load over such an image would give the message "RAM occupied". The combination of Interword and Spellmaster cause the error message not to appear, and the machine to crash. To avoid this, use one of the methods given in the Master Hints, Beebug Vol.6 No.3. For example, you can include:

```
*SRWRITE 900 +1 8007 7
```

in your !BOOT file immediately before *DLOAD. This assumes that you are using bank 7 for your user dictionary.

CALLING THE EDITOR

Dennis Weaver

You may not have spotted in the documentation for the Master Editor that when calling the Editor using *EDIT, you can add an optional filename. Thus:

```
*EDIT textfile
```

will enter the Editor and load in the file *textfile*. This must be in ASCII format, and will therefore not work for tokenised Basic programs.

AUTOMATIC ASSEMBLER DATE STAMP

Lee Calcraft

The Master's TIME\$ function can be used to automatically date-stamp any block of machine code each time that it is assembled. This provides a useful way of indicating the version number of a block of code. Just use the construction:

```
EQU$ TIME$
```

It is as simple as that.

You can also take advantage of a feature of the operating system to print the version time and date by simply calling the address at which it resides, providing that it is both preceded and terminated by a zero byte. Here is a header which makes use of the idea:

```
100          \ Start of code
110 JMP start
120 EQU$ 0:EQUW &0A0D
130 EQU$ "Prog name - assembled on"
140 EQUW &0A0D:EQU$ TIME$:EQU$ 0
150 .start \ Start of main routine
```

The instruction EQUW &0A0D is used to force a carriage return and line feed both at the start, and after the first line of text; while the final EQU\$ 0 indicates to the operating system that the end of the message has been reached.

Now if you execute:

```
CALL start
```

the main routine will run. But if you use:

```
CALL start + 3
```

this will print a message giving the time and date of assembly and the name of the source program.

B

PRINTER SURVEY

(Part 1)

How do you choose a dot-matrix printer from the vast pool available? Geoff Bains tries to help with the difficult task in our latest printer survey.

The most important features to consider when buying an NLQ dot-matrix printer are its speed and the quality of its NLQ print. The speed of printers is always somewhat exaggerated by manufacturers and so all the printers in this review have been tested again, printing the same piece of typical text and the printer's actual speed calculated. This is given in the comparison table. The quality of print is a little more subjective, and so a rating out of ten for each machine is given in the table. All prices quoted in the table and in the text include VAT.

All these printers are (at least) Epson compatible. In effect, this means they are all usable with the vast majority of Beeb software, and offer all the 'standard' features - characters at 5, 6, 8.5, 10, 12, 14, and 17 per inch and effects such as bold, double strike, italics and proportional spacing. In addition they can produce bit mapped graphics at 480-960 dots per line.

SEIKOSHA MP-1300A1 (£516)

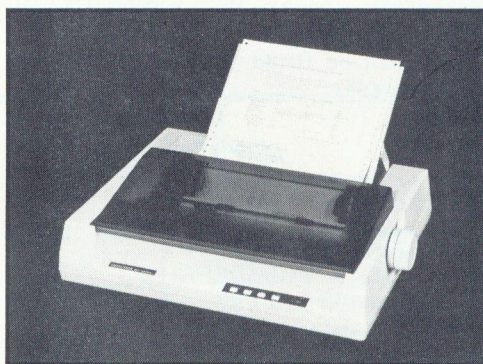
```
NLQ mode: ABCDEFGHI
LMNOPQRSTUVWXYZabcd
ghijklmnopqrstuvwxyz
123456789!"#$%&'()*
DRAFT mode: ABCDEFG
JKLMNOPQRSTUVWXYZab
efghijklmnopqrstuvw
xy0123456789!"#$%&'()
```

print out a typical page of text in NLQ mode - an actual speed of 39cps. In draft mode the speed is an excellent 131cps.

The Seikosha MP-1300A1 is the most expensive model reviewed here, but this printer is fast. It takes only

Although the MP-1300A1 is mostly well made, the tractor feed unit is rather flimsy and it must be removed to use cut sheet paper. The tractor also pulls the paper from the printer rather than push it in. This wastes a sheet each time a printout is torn off. Using cut sheet paper is very simple and fast as the MP-1300A1 has an automatic sheet feed.

The quality of the draft print is poor (though fast) but the NLQ printing is good - far better than Seikosha's cheaper model, the SP-1000A (see BEEBUG Vol.5 No.1).



The MP-1300A1 is IBM and Epson compatible and so it can produce all the extra foreign and graphics characters used by the IBM PC. A colour printing option is available for this machine for an extra £110. A small clip-on mechanism, a plug-in cartridge and a colour ribbon enable the MP-1300A1 to print in three colours and black.

It is doubtful if most Beeb users will find the MP-1300A1 worth the high price. However, if both speed and NLQ quality are important, this machine can justify its cost.

SEIKOSHA, c/o Data Distribution,
710 Birchwood Boulevard, Warrington, WA3 7PY

FUJITSU DX2100 (£500)

The DX2100 is a well designed and solidly built printer that looks ready for many year's service. It has both friction and tractor feed mechanisms and automatic paper loading.


```

NLQ mode: ABCDEFGHI
LMNOPQRSTUVWXYZabcd
efghijklmnopqrstuvwxyz
123456789!"#$%&'()*
DRAFT mode: ABCDEFGH
JKLMNOPQRSTUVWXYZab
cdefghijklmnopqrstuv
wxyz0123456789!"#$%&'

```

The DX2100's NLQ print is very good. It is dark and clear though a little thicker than some of the others. It is also produced at a

good rate of 27cps. In draft mode the print is also on the fast side (92cps) and extremely clear - in fact it is better than some cheap machines in NLQ mode. A wide carriage version (DX2200) is also available for £650 and a colour printing option can be fitted to either model for additional £63.

*FUJITSU, 2 Longwash Road,
Stockly Park, Uxbridge, UB11 1AB*

AMSTRAD DMP-3000 (£194)

```

!"#$%&'()*+,-./0123
456789:;<=>?@ABCDEFGH
IJKLMNOPQRSTUVWXYZ[\]
`abcdefghijklmnopqrstuvwxyz
NLQ MODE
!"#$%&'()*+,-./0123
456789:;<=>?@ABCDEFGH
IJKLMNOPQRSTUVWXYZ[\]
abcdefghijklmnopqrstuvwxyz

```

The DMP-3000 is nearly identical to the older, cheaper DMP-2000 (see BEEBUG Vol.5 No.1). The DMP-3000 was

introduced to accompany Amstrad's PC, and so this model is moulded in cream plastic rather than the charcoal grey used for the DMP-2000.

The DMP-3000 is, of course, IBM compatible but it also keeps the Epson compatibility of the older model.

One unusual feature of this printer is the paper path through the printer. Rather than enter and exit at the back of the printer and wind around a roller, the paper stays flat on its passage through the DMP-3000. The printhead acts down on it like a flatbed plotter. The advantage of this is that thicker paper or card can be printed, but it also means that the printer and paper take up a lot of room on a desk.



The Amstrad printers offer remarkable value. The DMP-3000 can print at a reasonable speed in both draft and NLQ modes, and the print quality is also excellent. There is no other printer available which can offer these features for this price.

AMSTRAD DMP-4000 (£2401)

Although the DMP-4000 is similar to the other Amstrad machines, it does not share their strange method of paper feed. This model looks and operates more like other 'normal' printers. The DMP-4000 is a wide carriage printer and can print 136 characters across paper 16in wide. However, the tractor feed pulls the paper through the printer rather than pushing it, like the MP-1300A1, with the same problems.

The DMP-4000 is also much faster than the other Amstrad models. The DMP-4000 can print at 112cps in draft mode and 26cps in NLQ mode - both reasonable speeds for an 80 column machine at this price. In draft mode the DMP-4000 can produce pages of text at the rate of three a minute. Otherwise, the DMP-4000 offers the same facilities as the DMP-3000 - the same reasonable draft print, good quality NLQ, and Epson and IBM compatibility. The DMP-4000 is built to a higher standard of construction than the other Amstrad model. Neither of these printers is really suitable for heavy and constant use, but more for running out the occasional letter or listing programs.

*AMSTRAD, 169 Kings Road,
Brentwood, Essex, CM14 4EF*

CANON A-50, (£379)

NLQ mode: ABCDEFGHIJ
LMNOPQRSTUVWXYZabcd
ghijklmnopqrstuvwxyz
123456789!"£\$%&'()*?
DRAFT mode: ABCDEFG
JKLMNOPQRSTUVWXYZab
cdefghijklmnopqrstuvw
xyz0123456789!@#\$%&'*~

Canon makes the KP-810 for Taxan (and the identical PW-1080). The A-50 is very similar. Like the

Taxan, the A-50 is available in a wide carriage version (the A-55) for £574. The A-50 is similar in quality and features to the older models but faster. Unlike the Taxan it is both IBM and Epson compatible and it can produce italic characters in NLQ mode.

NLQ mode can be selected from the front panel switches only as the printer is turned on. This of course means that any other control codes (such as italic style, or enlarged print) are then lost (as the machine is turned off).

The quality of print is excellent. The draft print is remarkably 'undotty' and the NLQ print is very neat and clear. The A-50 is quiet in operation (for a dot matrix printer), and it is very solidly built. There is a feeling of quality about this machine not to be found in the models from Amstrad.

CANON, Canon House, Manor Road,
Wallington, Surrey, SM6 0AJ

B

SUMMARY TABLE

Make & Model	Price	Draft speed		NLQ speed		NLQ Quality	Supplier
		Claimed	Actual	Claimed	Actual		
Amstrad DMP-3000	£194	105	50	25	13	6	Amstrad (0277) 228888
Amstrad DMP-4000	£401	200	112	50	26	6	Amstrad (0277) 228888
Canon A-50	£379	300	86	50	24	8	Cannon 01-773 3173
Diconix 150	£459	150	73	50	30	3	Norbain Micro (0734) 868855
Epson * LX-80	#	100	72	22	15	5	Epson 01-902 8892
Fujitsu DX2100	£500	220	92	44	27	5	Fujitsu 01-408 0043
Panasonic KX-P1091	£328	120	68	22	15	8	Panasonic (0753) 73181
Samleco DX-86	£276	120	63	35	17	2	Samleco (0753) 854717
Seikosha MP-1300A1	£516	300	131	50	39	7	DDL (0925) 821646
Star * NL-10	£285	100	76	30	17	9	Star Micronics 01-840 1800
Taxan * KP-810	#	140	79	30	21	8	Taxan (0344) 484646

* For comparison. # No longer available.

Prices include VAT. Speeds in cps. Quality rated out of 10 (10 = best)

The printer survey will be continued in the next issue of BEEBUG

Surac continues his discussion of high precision arithmetic by investigating the problems of multiplication.

Last month, I introduced the subject of high-precision arithmetic on the Beeb. The method worked on any number containing up to 250 digits; the limit is actually imposed by the maximum string length the computer can handle. This month, we will add long multiplication to last month's addition and subtraction. I will assume that you have read, or at least have, a copy of the previous article, which explained how numbers are stored, etc.

THE METHOD

We will carry out long multiplication in just the same way that we did (I did, anyway) at school - and still would if it was not for pocket calculators. We repeatedly multiply or divide one of the numbers by powers of 10, and then multiply each scaled value by the corresponding digit in the second number, summing the products. For instance:

$$1.234*456.7=(123.4*4)+(12.34*5) \\ +(1.234*6)+(0.1234*7)$$

We must also ensure that the result will fit in the available space. Unlike addition, multiplication can result in a much larger number than either of the multiplicands. For instance,

if you multiply 2 "n"-digit numbers, the result is up to "2n" digits long.

The routine must therefore check that the limits will not be exceeded, and send back a warning if necessary. The total number of integer digits in the 2 numbers must not exceed 250, or whatever smaller number the system was initialized to. You can also get the same sort of problem if there are too many decimal places in either number.

THE PROGRAM

Listing 1 does the job. It's not complete by itself, but has to be added to the routines I gave you last month. The main part is FNLongMult, which inputs the 2 numbers as "str1\$" and "str2\$". Its first act, at lines 20020-20040, is to work out whether the result will be positive or negative; if the 2 strings have opposite signs, the result is negative, otherwise it will be positive. Either way, they're both made into positive numbers - it makes the multiplication much easier - and "Pos" is a flag showing the eventual sign.

The function then checks that there won't be an overflow from too many integer or decimal places. If this is a risk, the routine sends back a minus sign ("-") and exits. If it's a danger, any calling routine should check for this error signal. Assuming that things are OK, line 20090 calls PROCMultLimit, if it's needed, to trim off any excess decimals before PROCAlignDP, from last month's Workshop, puts the two numbers into identical formats.

The main part of the multiplication routine starts at line 20110. Its first act is to set up a blank string ("prod\$") in which the answer will be built up. It is set to zeros in the same format as the two multiplicands. "Str1\$" is the number which is going to be multiplied and divided by powers of 10 through the long multiplication process. FNDPRight is used to scale it by the

highest power of 10 in str2\$, for example, if:

```
str1$=123.456
```

```
str2$=98765.2
```

the highest power in str2\$ is 5 (i.e. 10,000), so str1\$ is converted to:

```
12345600.000
```

Having set everything up, lines 20140 to 20200 pull out each digit in str2\$, starting at the most significant one (the one on the left). The function then adds that many copies of str1\$ into prod\$, and divides str1\$ by 10 to be ready for the next multiplying digit.

PROCm is actually used for the multiplication at each stage. It does it by repeated addition - slow but effective. Finally, with the result in prod\$, line 20210 sets the sign as needed before the function returns the answer.

Listing 1

```
20000 DEF FNLongMult(str1$,str2$)
20010 LOCAL D1,D2,I1,I2,L%,mult%,
      mult$,Pos,prod$
20020 Pos=TRUE
20030 IF LEFT$(str1$,1)="9"
      THEN str1$=FN10comp(str1$):
      Pos=NOT Pos
20040 IF LEFT$(str2$,1)="9"
      THEN str2$=FN10comp(str2$):
      Pos=NOT Pos
20050 I1=FNIntLen(str1$):
      I2=FNIntLen(str2$)
20060 IF (I1+I2)>(maxlen-2) THEN "-"
20070 D1=FNDecLen(str1$):
      D2=FNDecLen(str2$)
20080 IF (D1)>(maxlen-2)/2
      OR (D2)>(maxlen-2)/2 THEN "-"
20090 IF (I1+I2+D1+D2)>(maxlen-2)
      THEN PROCMultLimit
20100 PROCAlignDP
20110
      prod$=FNPAD(String$(FNMax(I1,I2),
      "0")+". "+String$(
      FNDecLen(str1$),"0"))
20120 str1$=FNDPRight(str1$,str2$)
20130 IF FNIntLen(str2$)=0
      THEN str1$=FNDPLeft(str1$)
20140 FOR L%=INSTR(str2$,".")-
      FNIntLen(str2$) TO LEN(str2$)
```

```
20150 mult$=MID$(str2$,L%,1)
20160 IF mult$="."
      THEN L%=L%+1:
      mult$=MID$(str2$,L%,1)
20170 mult%=VAL(mult$)
20180 IF mult%>0 THEN PROCm
20190 str1$=FNNDPLeft(str1$)
20200 NEXT
20210 IF NOT Pos
      THEN prod$=FN10comp(prod$)
20220 =prod$
20490 :
20500 DEF PROCMultLimit
20510 LOCAL d1,d2,decmax
20520 decmax=maxlen-I1-I2-2
20530 d1=D1-INT(decmax*D1/(D1+D2)+0.5)
20540 d2=D2-INT(decmax*D2/(D1+D2)+0.5)
20550 str1$=LEFT$(str1$,LEN(str1$)-d1)
20560 str2$=LEFT$(str2$,LEN(str2$)-d2)
20570 ENDPROC
20990 :
21000 DEF FNDPRight(s1$,s2$)
21010 LOCAL D,I1A,I1B,I2,temp$
21020 I1A=INSTR(s1$,".")-1
21030 I1B=FNIntLen(s1$)
21040 I2=FNMax(0,FNIntLen(s2$)-1)
21050 D=FNDecLen(s1$)
21060 temp$=LEFT$(s1$,I1A)
21070 temp$=RIGHT$(temp$,I1B)
21080 IF I2>D
      THEN temp$=temp$+RIGHT$(s1$,D)+
      String$(I2-D,"0")+ "."
      ELSE temp$=temp$+LEFT$(
      RIGHT$(s1$,D),I2)+
      "."+RIGHT$(s1$, (D-I2))
21090 =String$((maxlen-LEN(temp$)),
      "0")+temp$
21490 :
21500 DEF FNNDPLeft(s1$)
21510 LOCAL D,I
21520 I=INSTR(s1$,".")-1
21530 D=FNDecLen(s1$)
21540 =FNPAD(LEFT$(s1$,I-1)+ "." +
      MID$(s1$,I,1)+RIGHT$(s1$,D))
21990 :
22000 DEF PROCm
22010 LOCAL count%
22020 FOR count%=1 TO mult%
22030 prod$=FNLongAdd(prod$,str1$)
22040 NEXT
22050 ENDPROC
```

continued on page 54

HIGH RESOLUTION SCREEN DUMP FOR MODE 0

Although the BBC micro's screen display has a nominal resolution of 640 by 256 in mode 0, most printer dumps cannot match this resolution. David Lowndes Williams provides a routine to achieve extra high resolution on your printer with a so-called mode 00 dump.

This program will dump a mode 0 screen to most Epson compatible printers, the dump produced having twice the vertical resolution of the actual mode 0 screen. The dump produced is not distorted, and circles displayed on the screen are reproduced as circles on paper, not as ellipses.

NECESSARY HARDWARE

To use this program you need a dot matrix printer capable of graphics (bit image mode). The program as listed is set up for Epson (and compatible) printers; those with other printers will need to change the codes used.

USING THE PROGRAM

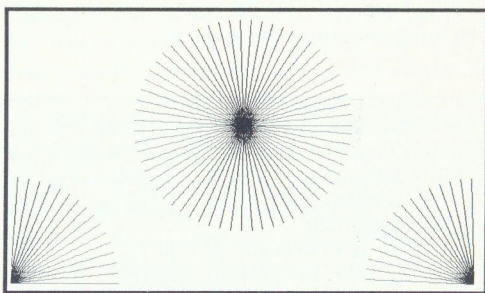
Make sure you have saved a copy of the Dump utility before starting to use it. The program reads in a spooled graphics file which can be created from any graphics program (which uses MOVE, DRAW, PLOT etc) by inserting *SPOOL <filename> before the graphics commands, and *SPOOL again at the end. As the graphics display is created, all the corresponding VDU codes are stored in the specified spool file. The dump routine first dumps the top half of the mode 0 screen and then dumps the second. A short demo program to create such a file is also included so that you can examine this if you are unsure of the technique.

When the dump program is run it will prompt for the entry of the filename of a spooled file (as described above). This file is then read and

expanded vertically so that only the top half of the original screen is displayed and this is then printed. After this the spooled file is read again while the bottom half of the screen is drawn, and then this is printed to complete the hard copy version. This means that tape users will have to rewind the tape to the beginning of the spooled file while the first half of the screen is printed (or alternatively save the spooled file twice).

LIMITATIONS

The dump program will cope with all PLOT, MOVE and DRAW commands, but it cannot cope very well with text, and cannot cope at all with any redefinition of the graphics origin or graphics windows.



Sample Printer Dump

PROGRAM NOTES

Line 170 reads in a byte from the spool file. If this byte is not a printable character PROCnotchr is called. Depending on the value of K, extra bytes may be required in which case PROCextra is called. This does not cater for user defined characters produced with VDU23, but this could be catered for by adding the line:

```
1045IF K=23 THEN PROCextra(9)
```

If the byte examined at line 180 is a printable character then it is printed. If it has the value 25 then PROCplot is called. This then reads the values of K, X and Y of the PLOT K,X,Y command. X and Y are both two-byte numbers which are read and deciphered by FNTwobyte. In the PROCplot procedure, line 1200 is the one which alters the Y co-ordinate of the PLOT command for the printer dump. If it is in the

top half of the screen, i.e. loop%=1, then the top half of the screen is transformed to the bottom half by Y=Y-512 and then expanded so that it fills the whole screen using Y=((Y-512)*2. If on the bottom half of the screen then Y=Y*2 expands the bottom half of the screen to fill the whole screen.

Each half of the original screen display is reproduced on the screen and then printed by PROCdump. This puts the printer into quad-density mode and each pixel is printed three times to achieved the required resolution.

Pressing Escape will terminate a printer dump and exit from the program, but printing of the current line will be completed first in order to leave the printer in a sensible state.

TECHNICAL NOTES

The program is configured for Epson and Epson compatible printers. Note that not all printers claimed as Epson compatible really are when it comes to graphics printing. However, all the VDU calls in the dump routine include REM statements stating the purpose of each string of numbers for users of other printers to modify as needed.

The screen dump simply scans the screen, in an 8 pixel band from left to right, working down the screen. The procedure PROCdump can also be used on its own as a mode 0 screen dump, but the dump will be twice as wide as it is long.

```

10 REM Program High Res Screen Dump
20 REM Version B2.1
30 REM Author David Lowndes Williams
40 REM BEEBUG November 1987
50 REM Program subject to copyright
60 :
100 ON ERROR GOTO 280
110 MODE0
120 :
130 file$=FNinput
140 FORloop%=1TO2
150 F=OPENIN(file$)
160 REPEAT
170 K=BGET#F
180 IF K<32 AND K<>25 VDU K:PROCnotchr
190 IF K>31 VDU K

```

```

200 IF K=25 PROCplot
210 UNTIL EOF#F
220 CLOSE#F
230 PROCdump
240 CLS
250 NEXTloop%
260 END
270 :
280 ON ERROR OFF
290 VDU1,27,1,64,1,24:REMreset printer
300 VDU3:CLS:REMstop output to printer
310 *FX229
320 REPORT:PRINT" at line ";ERL
330 IF ERR=222 PRINT'CHR$34;file$;CHR$
34;" does not exist on this drive."
340 END
350 :
1000 DEF PROCnotchr
1010 IF K=17 THEN PROCextra(1)
1020 IF K=18 THEN PROCextra(2)
1030 IF K=19 THEN PROCextra(5)
1040 IF K=22 THEN PROCextra(1)
1050 IF K=28 THEN PROCextra(4)
1060 IF K=31 THEN PROCextra(2)
1070 ENDPROC
1080 :
1090 DEF PROCextra(n%)
1100 LOCAL loop
1110 FOR loop=1 TO n%
1120 K=BGET#(F):VDU K
1130 NEXT
1140 ENDPROC
1150 :
1160 DEF PROCplot
1170 K=BGET#F
1180 X=FNTwobyte
1190 Y=FNTwobyte
1200 IF loop%=1 THEN Y=((Y-512)*2) ELSE
Y=Y*2
1210 PLOTK,X,Y
1220 ENDPROC
1230 :
1240 DEF FNTwobyte=BGET#F+(BGET#F*255)
1250 :
1260 DEF PROCdump
1270 LOCAL X%,Y%,L%
1280 REM prevent Escape from generating
an error.
1290 *FX229,32
1300 escape = FALSE
1310 VDU 2
1320 VDU1,27,1,64,1,24:REMreset printer
1330 VDU1,27,1,51,1,21:REMreset line sp
acing to 21/216ths of an inch.
1340 FOR Y%=1023 TO 0 STEP-32
1350 VDU1,27,1,90,1,128,1,7
1360 X%=0
1370 REPEAT

```



```

1380 IF INKEY(-113) THEN escape = TRUE
1390 L%=0
1400 IF POINT(X%,Y%)=1 L%=L%+128
1410 IF POINT(X%,Y%-4)=1 L%=L%+64
1420 IF POINT(X%,Y%-8)=1 L%=L%+32
1430 IF POINT(X%,Y%-12)=1 L%=L%+16
1440 IF POINT(X%,Y%-16)=1 L%=L%+8
1450 IF POINT(X%,Y%-20)=1 L%=L%+4
1460 IF POINT(X%,Y%-24)=1 L%=L%+2
1470 IF POINT(X%,Y%-28)=1 L%=L%+1
1480 VDUI,L%,1,L%,1,L%
1490 X%=X%+2
1500 UNTIL X%>1279
1510 VDUI,10:REM line feed.
1520 IF escape THEN GOTO 280
1530 NEXT Y%
1540 VDUI
1550 REMrestore Escape key.
1560 *FX229
1570 ENDPROC
1580 :
1590 DEF FNinput
1600 LOCALf$
1610 PRINTSPC25"Mode-00 screen dump."
1620 PRINTSPC22"By David Lowndes Willia
ms.""
1630 INPUT"Enter name of spool file:";f
$

```

```

1640 CLS
1650 =f$>L.

*****

10 REM Program SPLDemo
20 REM Version B1.0
30 REM Author David Lowndes Williams
40 REM BEEBUG November 1987
50 REM Program subject to copyright
60 :
1000 MODE 0
1010 :
1020 *SPOOL data
1030 :
1040 PRINTTAB(37,2)"Spokes!"
1050 FORI=0 TO 2*PI STEP 0.1:MOVE 640,5
12:DRAW640+300*COS(I),512+300*SIN(I):NEX
TI
1060 FORI=0 TO 0.5*PI STEP 0.1:MOVE0,0:
DRAW300*COS(I),300*SIN(I):NEXT
1070 FORI=0.5 TO PI STEP 0.1:MOVE1279,0
:DRAW1279+300*COS(I),300*SIN(I):NEXT
1080 :
1090 *SPOOL
1100 :
1110 END

```

WORKSHOP - LONG MULTIPLICATION (continued from page 52)

TRYING IT OUT

That's the main code. To try it out, add the instructions in Listing 2 together with last month's procedures. When you run the full program, it allows you to try different multiplications to see the result. For comparison, it also gives the answer using the Beeb's built-in routines; remember that, with anything more than 5-6 significant digits in each number, they're not as precise as the long arithmetic.

You'll also see just how slow the long routines are. As I pointed out last month, the code is meant to show the principle, rather than be as fast as possible.

The description of the routines for long addition, subtraction and multiplication has taken up more space than was expected, and we shall only publish a further Workshop covering long division if there is sufficient demand for this.

Listing 2

```

1000 MODE3
1010 INPUT "How many digits? "N%
1020 PROCInit(N%)
1030 REPEAT
1040 INPUT "'A? "A1$
1050 INPUT "B? "B1$
1060 A$=FNLongEq(A1$)
1070 B$=FNLongEq(B1$)
1080 PRINT "'A stored as:"'A$
1090 PRINT "B stored as:"'B$
1100 AB$=FNLongMult(A$,B$)
1110 PRINT "'A=";A1$;'B=";B1$
1120 IF AB$="-"
THEN PRINT "'No calculation
-risk of overflow"
ELSE PRINT"'A*B:"
'FNLongPrint(AB$)
1130 PRINT "Stored as:"'AB$
1140 PRINT "Conventional value:";
VAL(A1$)*VAL(B1$)
1150 UNTIL 0
1160 END

```


yourself to hard copy. The WYSIWYG Plus commands can also be used from Basic, assembler and several other commercial programs to good effect.

USING WYSIWYG PLUS

In Wordwise and Wordwise Plus, insertion of a command is simply a matter of preceding it with an embedded command start code (f1 - green) and following it with an embedded command end code (f2 - white). The same is true of Interword although a *INTER command must also be used to start the whole process off.

In View things are a little more complicated. Here highlight 2 (Shift-f5) starts the command and highlight 1 (Shift-f4) ends it. A *VIEW command is then used to preview the text, followed by a number 1-4 corresponding to the standard View commands: PRINT, SCREEN, SHEETS (to printer) and SHEETS (to screen).

There are over 50 commands that can be put into your text. Most are self explanatory - BOLD, CONDENSED, DOUBLESTRIKE, DRAFT, ELITE, EMPHASISED, ENLARGED, ITALICS, NLQ, PICA, PROPORTIONAL, SUBSCRIPT, SUPERScript and UNDERLINE. Most effects are switched on with the relevant star command and off again with the command followed by zero. All these commands show their effect on the screen as well as on the printer paper.

Daisywheel printers are covered too. The command *WHEEL modifies other 'effect' commands (such as *ITALICS) by introducing a pause during print out, and then prompting for a change of printwheel.

With WYSIWYG Plus, gone are the days of worrying about pound signs and other special characters. *BBC selects the normal Beeb character set on screen and ensures that the pound and hash signs are printed correctly. *LANGUAGE selects the nationality-specific character set from your printer and *UK and *US are provided as alternative commands for the two most often used languages.

Other commands alter the number of characters displayed and printed across the screen and paper. WYSIWYG Plus enables the Beeb to display 40, 48, 64, 80, 136 or 160 characters across a mode 3 or 0 screen. Of course, this entails the normal characters being expanded and condensed. This inevitably means a little distortion. At 160 characters across the screen you need a good monitor and good eyesight to read them. However, the possibility is there and very useful it is too.

Some of the WYSIWYG Plus facilities are purely printer effects. Justification, for example, is not performed on screen (that's too much to ask). Similarly *UNIDIRECTIONAL, *FORMFEED, *LINEFEED *QUIET, and *SKIP perform purely printer tasks.

PRINTER DRIVER GENERATOR

Of course, all these effects rely on your printer's ability to perform them. The WYSIWYG Plus ROM has a built-in driver for Epson compatible printers. For other printer standards a printer driver generator is also supplied.

The generator displays all the codes to select the various effects. Pressing f0 switches between on and off codes. Pressing f1 alternates between listing in decimal, hex or ASCII. In any mode the cursor is simply moved over the codes to be changed, and the new values or letters are typed in.

A few minutes at the keyboard with your printer manual will soon produce a suitable driver for your printer. This can then be saved onto disc, and loaded into the computer again when required with the command *DRIVER.

CONCLUSIONS

No ROM can perform miracles. However, WYSIWYG Plus combines the best features of a printer utility ROM with a valuable addition to the most popular Beeb word processors. If you want to improve your word processor but stick with your favourite program, WYSIWYG Plus is a worthwhile addition to your micro.

B

1st COURSE

Colouring your Beeb

First Course this month takes a look at the use of colour graphics on the Beeb as Mike Williams explains how to select and mix colours for your needs.

In our First Course series, I dealt not so long ago with many of the graphics possibilities that derive from the use of the PLOT instruction (and also MOVE and DRAW). Another important aspect

of graphics on the Beeb is the use of colour, and I propose now to deal with this, starting this month with the basics. As well as discussing some of the more subtle uses of colour, I will also show how you can achieve more shades and hues of colour than might be expected.

As you will know, the number of colours which you can use with any program is determined by the graphics mode that you choose. Modes 0 and 4 allow just two colours, modes 1 and 5 allow four colours, while mode 2 allows the full 16 colours of the Beeb. To talk of 16 colours though is a misleading. In practice your choice is limited to black and white, 6 'real' colours (comprising red, green, yellow, blue, magenta and cyan), and 8 'flashing' colours. These latter produce a regular alternating flashing between pairs of colours as listed in the User Guides. We shall concentrate on the six basic colours together with a judicious use of black and white.

The table shows the colours available in each mode (excluding the flashing colours) and the numbers by which these colours are referenced (screen modes 128 and over invoke shadow memory on the Master and Compact). As you can see, there are both foreground and background colours. Background colours always have numbers of 128 or over. The numbers are important as they are the means

by which you specify which colour you are using in any particular task.

For example, in modes 1 and 5 there are just four colours referred to by the numbers 0 to 3 if we are specifying the foreground, and 128 to 131 if specifying a background. There are four possible choices of colour for both foreground and background (black, red, yellow and white by default), and when either of these modes is selected the foreground colour is automatically set to 3 (white) and the background to 0 (black).

MODE	FOREGROUND	BACKGROUND	COLOUR
0(128)	0	128	Black
4(132)	1	129	White
	0	128	Black
1(129)	1	129	Red
5(133)	2	130	Yellow
	3	131	White
	0	128	Black
	1	129	Red
	2	130	Green
2(130)	3	131	Yellow
	4	132	Blue
	5	133	Magenta
	6	134	Cyan
	7	135	White

Table 1. Default colours in graphics modes

Once a particular mode has been selected, then any of the default colours for that mode may be chosen for the foreground and for the background. Moreover, if we don't like the default set of colours, then any of these may be changed to any of the other colours of which the micro is capable.

SELECTING COLOURS

In any graphics mode it is the GCOL instruction which is used to select a colour. The syntax of this is:

GCOL mode, colour

The value **mode** is nothing to do with the screen mode selected with the **MODE** command. Instead it determines the 'mode' of plotting. This can be quite complex so for now we will be using plot mode 0 for now, which just means draw or plot in the colour specified. The second value **colour** specifies the foreground or background colour to be used from that point on. These numbers are the ones given in the table. Changing a foreground or background colour in this way affects only future drawing; colours already on the screen remain as they are.

An example may not come amiss at this stage. The following program is quite simple, using a procedure which can be called to draw two triangles on a mode 5 screen with choice of colour for both triangles and the background. Please keep to the line numbering as given to allow for later additions.

```

100 MODE 5
110 INPUT"Background:" B
120 INPUT"Triangle 1: " C1
130 INPUT"Triangle 2: " C2
140 PROCtriangles(C1,C2,B)
150 END
160 :
1000 DEF PROCtriangles(c1,c2,b)
1040 GCOL 0,b+128:CLG
1050 GCOL 0,c1
1060 MOVE320,204:DRAW640,666
1070 DRAW960,204:DRAW320,204
1080 GCOL 0,c2
1090 MOVE320,820:DRAW640,358
1100 DRAW960,820:DRAW320,820
1110 ENDPROC

```

The choice of mode 5 allows a total of four different colours on screen together. The procedure uses variables **c1** and **c2** as the colours of the triangles, and **b** as the colour of the background. Remember that the **GCOL** instruction only specifies a colour; it does not affect the screen display. Thus to colour the background, line 1040 must also include a **CLG** instruction (to 'clear' the graphics background with the currently selected background colour).

Although mode 5 allows only four colours no errors will occur if numbers other than 0 to 3 are specified. Higher numbers just keep cycling

round the same set of colours. Thus colour 9 (4+4+1) would be the same as colour 1 (red).

CHANGING COLOURS

Given the use of a mode such as mode 5, you may not always wish to use the four default colours of black, red, yellow and white, and you may ask if it is possible to change these? The answer is 'Yes', but we need to be clear about the terminology we use.

Taking mode 5 as an example the four colours referred to as 0, 1, 2, and 3 are strictly 'logical' colours. The mode 5 colours are always specified by these four numbers, but we can choose them to be any 4 of the Beeb's 16 'actual' colours. The distinction between logical and actual colours is quite important.

The way to select a different set of colours in a given mode is through the use of the **VDU19** command. The format of this is simply:

```
VDU19,logical,actual,0,0,0
```

where logical refers to one of the logical colour numbers as appropriate to the mode in use, and actual refers to a real colour (using the numbers given previously in the table for mode 2). Thus to change the default colour of red (colour 1) to blue in mode 5 you would put:

```
VDU19,1,4,0,0,0
```

From then on a reference to logical colour 1 will imply the use of actual colour 4. Master and Compact users can avoid the need to include the apparently unnecessary zeros by writing:

```
VDU19,1,4|
```

To illustrate the use of this we will revise our previous program so that the two triangles and the background may be chosen from any of the available colours. The procedure **PROCtriangles** should be modified to read as follows:

```

1000 DEF PROCtriangles(c1,c2,b)
1010 VDU19,0,c1,0,0,0
1020 VDU19,1,c2,0,0,0
1030 VDU19,2,b,0,0,0
1040 GCOL 0,130:CLG
1050 GCOL 0,0
1060 MOVE320,204:DRAW640,666
1070 DRAW960,204:DRAW320,204
1080 GCOL 0,1
1090 MOVE320,820:DRAW640,358
1100 DRAW960,820:DRAW320,820
1110 ENDPROC

```


(you may wish to edit your previous version using the Copy key).

In the procedure the logical colours of 0, 1 and 2 are used explicitly for the two triangles and the background respectively. The procedure uses three VDU19 calls to determine which actual colour is represented by each logical colour number. Whereas the first version of this program was limited to the four default colours of mode 5, this version allows the use of any of the 16 colours of which the Beeb is capable.

One point to note is that if you have already created part of a display with one setting of a logical colour, changing the associated actual colour by using VDU 19, will change the colour already on the screen. You cannot use VDU19 to increase the number of colours on the screen. That is a function of the screen mode chosen and remains fixed.

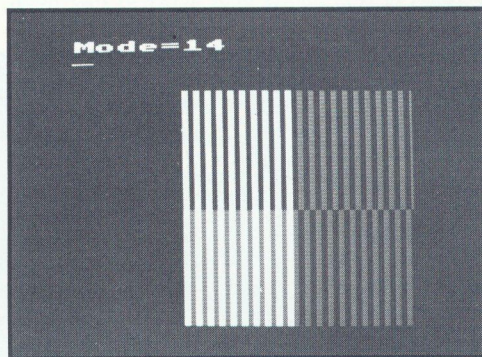
One useful trick here is a special form of the VDU19 command. If you have already created a screen display, then Ctrl-S (hold down the Ctrl key and press 'S') may be used in immediate mode to change any of the colours visible on the screen without the VDU command itself appearing on the screen. Simply press Ctrl-S, followed by the logical colour number, the actual colour number and three zeros, all without any commas. The colour specified will instantly change. Take View for example. To change screen colours, select mode 3 and try Ctrl-S 04000 followed by Ctrl-S 13000.

SIMPLE COLOUR MIXING

To conclude our first look at the use of colour, we will consider a comparatively simple way in which we can (rather crudely) increase the range of colours available. We have been using the GCOL command with plotting mode (the first parameter) set to 0. If you check with your User Guide you will find that this value can apparently be anything between 0 and 4. In fact, any value up to 255 is possible, and the effects produced are quite interesting, providing nothing less than striped paint!

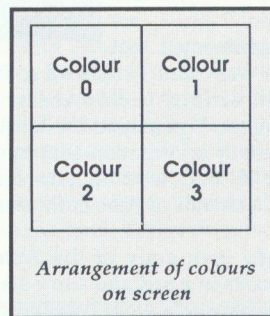
I do not propose to go into a lengthy discussion here of how these effects arise. Instead I have

put together a short program which I hope you will be able to use to investigate and select the shades that may be of use to you in a particular application. The program has been written in mode 5, which allows four logical colours, but could be readily adapted to other graphics modes. The program allows you to decide which four of the Beeb's colours are to be selected, and four coloured rectangles are displayed on the screen (arranged as in the diagram) to show the effects produced for each plotting mode from 0 to 127.



The first section from line 110 to line 18 determines the four colours to be used in the display. This dialogue is then cleared from the screen and the program loops through each plotting mode in turn to show the effects.

The plotting mode is also displayed on the screen for reference. The four rectangles are specified in turn as four graphics windows (note the semi-colons after each parameter in the VDU 24 command). GCOL and CLG are then used to select a background colour and use it to colour the window.



Striped colours may be used as foreground or background colours, and I would suggest you

continued on page 63

BOOKS FOR SERIOUS USERS

Detailed reference books are indispensable to the dedicated enthusiast. Two such comprehensive guides to the BBC micro and Master series have recently been published, one being an extensively revised and updated version of a previous best seller in this field. Peter Rochford weighs up their relative advantages.

**The New
Advanced
User Guide
by Dickens
and
Holmes,
published
by Adder at
£19.95.**

When the original Advanced User Guide was reviewed some four years ago in BEEBUG Vol.2 No.6, we commented that

it was surely destined to become the Bible for all serious Beeb owners. We weren't wrong either. It provided the kind of detailed and in-depth information that the User Guide never did, and was an enormous success with thousands of Beeb enthusiasts.

My own copy of the Advanced User Guide looks in a sad and sorry state these days, so the release of a New Advanced User Guide at this time is most welcome. This is a revised and updated version of the original book now covering the Model B, the B+, the Master, the Master Compact and the Electron.

Within the 440 pages of this spiral-bound book, are vast amounts of detailed information

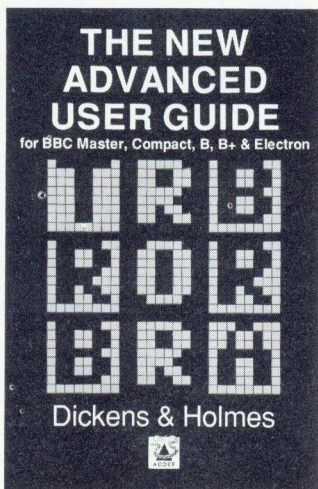
covering all aspects of the hardware and software incorporated in the range of BBC micros. There are in-depth details of the Basic assembler, with specific references to both the 65C02 and 65C12, all the FX/OSBYTE calls, implementation of paged ROM software, events and interrupts, shadow RAM, the video and serial ULAs, input/output ports and interfacing, using the Tube, the Master CMOS RAM and real-time clock, programming the 1770 disc controller, and much much more.

To supplement this information, there are several listings of example programs that in themselves provide useful utilities. These include a printer buffer program, and one for reading IBM PC format discs using any BBC computer that has a 1770 disc controller.

Not only has additional information been added in this new release of the Advanced User Guide, but the guide has also been re-organised, making reference to particular topics that much easier.

The New Advanced User Guide like its predecessor, is not a book for the beginner or casual user of the Beeb. It is aimed at the programmer or serious user who requires the kind of information not available from the User Guides supplied by Acorn for the various machines. You may well ask, however, if the Master owner who already has the Master Reference manuals will find enough new information in this book to justify its purchase. The answer must be a definite yes. Although this book contains information that may already be found in the Master manuals, there is a great deal here that you will not find in Acorn's offerings.

I for one am delighted at the release of this book. I have found the original over the last four years to be invaluable, and the dilapidated condition of my own copy is testimony to the use and attention it has received. This new book will take its place on my bookshelf and continue to be my main source of reference whenever I am urgently seeking that vital piece of information about the software or hardware of my machine.

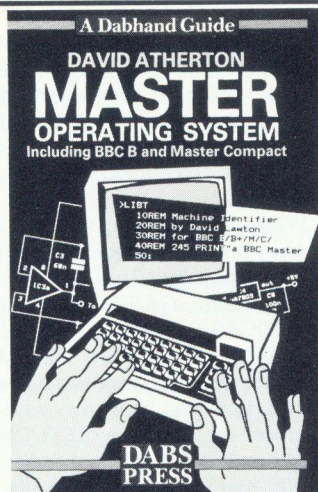


**Master
Operating
System
by David
Atherton,
published
by Dabs
Press at
£12.95.**

Like the Advanced User Guide, this publication from Dabs Press is aimed at the more serious user of the BBC range of machines. Written by David Atherton, formerly of the BBC's own software house, BBC Soft, it deals in the main with all aspects of the Master's operating system.

There are details of all new OS commands featured on the Master, the new 65C12 opcodes, all the new OSBYTE/OSWORD calls, sideways RAM programming, shadow RAM programming, paged ROM implementation, loading and saving of ROM images, driving the Tube in both directions, and filing system changes. The Tube section in particular deserves special mention as it really is excellent. There are a number of program listings provided in the book to illustrate various features and programming techniques. Some of these are quite useful utilities, especially the CMOS RAM editor and the clock ROM.

One of the most interesting and useful features of this book for me is the detailing of lists of differences between the various versions of the system software. For example, the 1770 DFS as



supplied as an upgrade for the Model B, or as standard with the B+ and Master has suffered with a variety of bugs. In consequence, there have been many versions released. Here, at last, are details of the known bugs in each release which many will find of interest. Apart from the bug aspect of the software, having a detailed list of the differences in the software supplied for all the different machines will be a boon for many programmers who wish to make sure that their software will run on all machines.

The Master Operating System is a spiral-bound book and has 270 pages. The layout is good and there is a reasonable index and useful glossary too. At the back of the book there are several appendices providing lots of lists and quick look-up guides to the VDU commands, OS commands, memory map etc, all most useful to programmers. Along with this there are some hardware details such as pin-outs for the machines' various ports and sockets.

Although this book is called the Master Operating System, it is also claimed to be of interest to Model B, B+ and Electron owners. I would have to say that I find this only slightly true. Yes, there is information here that may be of interest to owners of Acorn machines other than the Master, but the book most certainly caters in the main for Master owners.

David Atherton's 'The Master Operating System' does contain a wealth of information for those who need it, and I think it is reasonably priced too. Check carefully, though, if contemplating the purchase of this book and you already have the Master Reference manuals parts 1 and 2. I personally would consider it hard to justify buying the book in this case, but do decide this for yourself. **B**

ARCHIMEDES COMPETITION - THE WINNING PROGRAM

As promised, we have been able to include on this month's magazine cassette/disc, a copy of the winning program by Mr P. Barrett in our alphametics competition to win an Archimedes. The program solves alphametics, as defined in the original competition, very quickly indeed. A fully annotated version of Mr Barrett's program is available as a photo-copied listing on receipt of an A5 SAE. We will also include a copy of Mr Barrett's article, explaining the techniques he used in writing his winning program. Lack of space has prevented us from publishing this in the current issue of the magazine.

ADVANCED BASIC

Do you fancy running Basic V, the latest Archimedes version of BBC Basic, on your Beeb? Dave Somers takes a look at Tubelink's Advanced Basic which enables you to do just that, within limitations.

Product	Advanced Basic
Supplier	Tubelink P.O. Box 641, London NW9 8TF. Tel. 01-205 9393
Price	£29.95 incl. (disc only) £34.95 incl. (disc + EPROM)

Advanced Basic from Tubelink provides you with a version of Archimedes Basic to run on a BBC Micro. However, you **MUST** have a 6502/65C102 second (or Turbo) processor for this to function correctly. The program will also work on an Electron if you have the PMS E2P second processor, or an adaptor to enable one of the above to be used. The disc version contains a ROM image for loading into Sideways RAM, or the EPROM version can be installed as usual. In both cases a 56 page A5 typeset instruction manual is provided, and this contains details of all the facilities available. Advanced Basic is an extension of HiBasic 87, which is the fastest and most accurate version, so a copy of this is also provided on the disc (under licence from Acorn), for loading into sideways RAM.

Entering Advanced Basic is simplicity itself with the single command ***ABASIC**. There is a slight pause while the code is copied over the Tube before the familiar chevron **>** prompt appears.

Advanced Basic offers most of the facilities provided by Basic V with a few exceptions - mainly due to the lack of hardware on the Beeb which is specific to the Archimedes. Commands for controlling sound such as **TEMPO**, **BEATS**, etc., cannot be used, nor the

TINT command and the four parameter **COLOUR** commands. If you attempt to use them, an error message is given. Entry-time syntax checking for missing quotes and brackets, and operations on whole arrays are not provided either.

Why change to Advanced Basic? There are several new programming structures available, with multiple line **IF-THEN-ELSE-ENDIF** statements, **CASE-OF-WHEN-OTHERWISE-ENDCASE** blocks, **WHILE-ENDWHILE** loops (which are a bit like **REPEAT-UNTIL** loops except that the condition is tested at the top, and not the bottom), and functions and procedures can now **RETURN** values via their parameters (as opposed to the other versions of BBC Basic where you can only send values).

Error trapping is now easier with additional commands at your disposal. You can, for instance, trap errors and handle them within loops, functions, or procedures without losing the nesting of such loops. The error message itself can be manipulated as it is contained in a string, **REPORT\$**, thus making error handling code very easy to implement.

One of the more useful facilities is the ability to install function and procedure libraries. When developing software, you often build up a standard set of functions and procedures that you use frequently. By using the **INSTALL** or **LIBRARY** function, you can load a program containing these into memory. However, any functions or procedures are not shown in the listing of the program, but should your program make reference to one of them, the installed file is searched, and if the function or procedure is found, it is executed.

To use any mouse commands, you will need either the **AMX Super ROM** or **Nidd Valley's Chauffeur ROM** to be present in your machine. A simple driver program has to be loaded to interface Advanced Basic with the relevant ROM before Basic V style keywords (such as **MOUSE**) will work. However, no attempt has been made to drive an on-screen pointer, which may lead to difficulties when trying to run programs written on the Archimedes.

Enhanced graphic commands such as CIRCLE, RECTANGLE, ELLIPSE, etc., can only be used effectively on a B or B+ if the Acorn Graphics Extension ROM is fitted. No such problem exists on the Master or Compact, where such facilities are already provided.

A unique feature specific to Advanced Basic is the EDITO command. It takes the same parameters as the LISTO command, and is used to select the program format when it is converted to text for use with the Editor.

The in-built assembler is an enhanced version of that on a Beeb or Master and a host of additional options is available. You can choose to assemble code directly to memory in either the 2nd processor or I/O processor, to shadow RAM, sideways RAM, or even to a disc file. The EQUate facility has been enhanced with EQUr to assemble a 5 byte real number, and EQUf to link in a pre-assembled file.

Like Basic V, Advanced Basic has a keyword help facility, taking the format of HELP <keyword>. Unfortunately, due to memory limitations, the help text exists on disc and needs to be placed in the current library. This results in a long pause whilst the file is searched for the text to display. One advantage of such a system is that you can append your own help text onto the end of this file. So, for example, you could detail parameters for the

*FX calls. Details of the file structure are given in the manual, although you will require View to make any amendments.

A whole host of new operators has been introduced. Binary values can now be entered directly by prefixing them with "%" - akin to the way that hex numbers are prefixed with "&". Integer variables can now be shifted arithmetically any number of bits either to the left or to the right. There is also a new indirection operator to allow for the poking and peeking of real numbers.

FILE TRANSFER

A utility is provided on the disc to transfer software from Beeb to Archimedes and vice versa using the serial RS423 port.

FINAL COMMENTS

Advanced Basic enables you to take advantage of many of the facilities offered by Basic V on the Archimedes at a very modest cost. Using Advanced Basic is most addictive - all those extra commands and facilities enable quite complex Basic to be written with as little fuss as possible. However, there are some limitations imposed, as not all of the facilities of Basic V are available. It should be possible to write programs in Advanced Basic that will work on the Archimedes, but not all Archimedes programs will work in Advanced Basic. **B**

COLOURING YOUR BEEB (continued from page 59)

experiment by referring back to my previous articles on the use of PLOT (see introduction), and experiment with some of those programs to see what you can achieve. Next month we will look at the meaning of the plotting modes as listed in the User Guide, and some of the more subtle colour effects that can result.

```
100 MODE 5
110 INPUT"Colour 0: " c0
120 VDU19,0,c0,0,0,0
130 INPUT"Colour 1: " c1
140 VDU19,1,c1,0,0,0
150 INPUT"Colour 2: " c2
160 VDU19,2,c2,0,0,0
```

```
170 INPUT"Colour 3: " c3
180 VDU19,3,c3,0,0,0
190 CLS
200 FOR mode=0 TO 127
210 PRINTTAB(0,2)"Mode=";mode
220 VDU24,320;512;632;820;
230 GCOL mode,0+128:CLG
240 VDU24,640;512;960;820;
250 GCOL mode,1+128:CLG
260 VDU24,320;204;632;508;
270 GCOL mode,2+128:CLG
280 VDU24,640;204;960;508;
290 GCOL mode,3+128:CLG
300 G=GET
310 NEXT mode
320 END
```




Jonathan Temple will be familiar to many BEEBUG readers as the author of some of our most enjoyable games programs. In this issue we present Jonathan's latest game, Dracula. Can you resist the challenge?

Bound by an ancient curse you, Count Dracula, must collect seven gold rings before nightfall. Pursuing you are several angry villagers, intent on putting an end to your reign of terror.

Once you have collected the rings you must wait for the sunset and darkness - at which point you may take your revenge on the now frightened villagers. Then the sun rises, and once again you are in danger.

There are seven castles to work your way through, with a bonus of 5000 points awarded on completion of the seventh. Also, having wreaked your vengeance on the villagers, you will be awarded a bonus depending on how many you managed to catch. The playing keys are the usual Z/X for left/right and */? for up/down.

The program is fairly straightforward to type in, but do make sure you save a copy before attempting to run it. All you have to do now is wait for the shades of night, and your reign of terror can begin.

If you find the game too difficult, try increasing the value of T% at line 2650 which will increase the time allowed for each screen. Likewise, decreasing this value will make the game more difficult.

```

10 REM Program DRACULA
20 REM Version B2.2
30 REM Author Jonathan Temple
40 REM Beebug November 1987
50 REM Program subject to Copyright
60 :
100 MODE 2:ON ERROR GOTO 3120
110 PROCchars:PROCenvs:PROCinit
120 REPEAT:S=1:S%=0:D%=3
130 REPEAT:PROCintro
140 REPEAT:PROCscreen
150 REPEAT N%=N% EOR1
160 PROCdrac:PROCchase:PROctime
170 UNTIL E%
180 IF E%=1 PROCkilled
190 UNTIL E%=2 OR D%=0
200 IF E%=2 PROCbonus
210 UNTIL D%=0
220 PROCend
230 UNTIL FALSE
240 END
250 :
1000 DEFPROCbonus
1010 F%=-1:Q%=10:V%=0
1020 T%=250:REPEAT
1030 REPEAT N%=N% EOR1
1040 PROCdrac:PROCchase
1050 T%=T%-1
1060 UNTIL T%=0 OR E%=1
1070 IF E%=1 PROCget
1080 UNTIL T%=0
1090 VDU 4,17,128,12
1100 V$=STRING$(V%,V$(0)+CHR$(11))
1110 COLOUR5:B%=V%*25+S*100-5000*(S=7)
1120 PRINTTAB((20-V%)/2,14) V$
1130 PROCprint("BONUS: "+STR$(B%),320,4
44,4,6)
1140 TIME=0:REPEAT UNTIL TIME>400
1150 S%=S%B%:S=S+1:IF S>7 S=1
1160 ENDPROC
1170 :
1180 DEFPROCget
1190 E%=0:MOVE J%,K%:SOUND 17,1,5,2
1200 IFL%=64 PRINTV$(0) ELSE PRINTV$(1)
1210 RESTORE2720:FOR M%=1 TO RND(2)
1220 READ J%(N%),K%,L%(N%):NEXT
1230 K%(N%)=860-128*RND(4)
1240 M%(N%)=0:MOVE J%(N%),K%(N%)
1250 IFL%(N%)=64 PRINTV$(0) ELSE PRINTV
$(1)
1260 V%=V%+1
1270 ENDPROC
1280 :
1290 DEFPROCdrac
1300 A%=X%:B%=Y%:C%=Z%
1310 IF INKEY-73 IFPOINT(X%,Y%-48)=8 IF
X%MOD64=0 Y%=Y%+32:GOTO1350
1320 IF INKEY-105 IFPOINT(X%,Y%-96)=8 I
FX%MOD64=0 Y%=Y%-32:GOTO1350
1330 IF INKEY-98 IFPOINT(X%-8,Y%)MOD8=0
IFPOINT(X%-8,Y%-96)>0 OR POINT(X%-40,Y%
-96)>0 X%=X%-32:Z%=1:GOTO1350

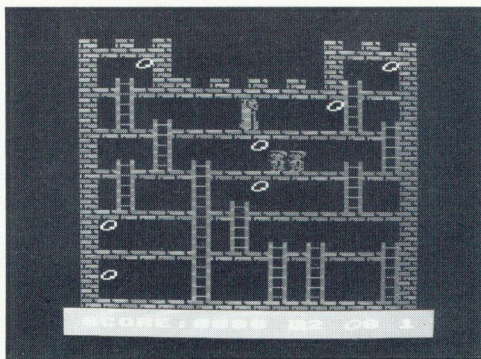
```



```

1340 IF INKEY-67 IFPOINT(X%+64,Y%)<>P%
IFPOINT(X%+64,Y%-96)>0 OR POINT(X%+96,Y%
-96)>0 X%=X%+32:Z%=0
1350 IF X%<>A% OR Y%<>B% GCOL3,1:MOVE A
%,B%:PRINT D$(C%):MOVE X%,Y%:PRINT D$(Z%
)
1360 IFPOINT(X%+56,Y%-12)=10 GCOL3,10:M
OVE X%,Y%:VDU231:R%=R%+1:PROCscore(50):S
OUND 17,1,100,4
1370 ENDPROC
1380 :
1390 DEFPROCscore(N%)
1400 S%=S%+N%:VDU4,31,7,30
1410 PRINT LEFT$("0000",4-LEN(STR$ S%))
;S%;TAB(16,30);R%
1420 VDU 5
1430 ENDPROC
1440 :
1450 DEFPROCtime
1460 T%=T%-1:IF T%>90 ENDPROC
1470 IFT%MOD5>0 ENDPROC
1480 SOUND 1,1,T%,4
1490 IFT%=90 VDU19,14,3,0;
1500 IFT%=60 VDU19,14,1,0;
1510 IFT%=30 VDU19,15,0,0;
1520 IFT%=0 VDU19,14,0,0;:E%=1-(R%=7):
SOUND 1,4,30,10
1530 ENDPROC
1540 :
1550 DEFPROCchase
1560 J%=J%(N%):K%=K%(N%)
1570 L%=L%(N%):M%=M%(N%)
1580 IFM% PROCcladder ELSE PROCwall
1590 J%(N%)=J%(N%)+L%(N%)
1600 K%(N%)=K%(N%)+M%(N%)
1610 GCOL 3,5:MOVE J%,K%
1620 IFABS(J%-X%)<64 IFABS(K%-Y%)<96 E%
=1:ENDPROC
1630 IF L%=64 PRINT V$(0) ELSE PRINT V$
(1)
1640 MOVE J%(N%),K%(N%)
1650 IF L%(N%)=64 PRINT V$(0) ELSE PRIN
T V$(1)
1660 IFABS(J%(N%)-X%)<64 IFABS(K%(N%)-Y
%)<96 J%=J%(N%):K%=K%(N%):L%=L%(N%):E%=1
1670 ENDPROC
1680 :
1690 DEFPROCcladder
1700 IFM%=32 IFPOINT(J%,K%)<>8 PROCsele
ct:ENDPROC
1710 IFM%=-32 IFPOINT(J%,K%-64)<>8 PROC
select:ENDPROC
1720 IFPOINT(J%-8,K%-64)=P% IFRND(N%*2+
2)<2 OR K%=Y%-32 PROCselect
1730 ENDPROC
1740 :
1750 DEFPROCselect
1760 M%(N%)=0
1770 IFPOINT(J%+64,K%)=P% L%(N%)=-64:EN
DPROC
1780 IFPOINT(J%-8,K%)=P% L%(N%)=64:ENDP
ROC

```



```

1790 IFJ%<X% L%(N%)=F%*64 ELSE L%(N%)=-
F%*64
1800 ENDPROC
1810 :
1820 DEFPROCwall
1830 IFPOINT(J%,K%)=8 IFY%*F%>K%+32 OR
RND(Q%)=1 L%(N%)=0:M%(N%)=32:ENDPROC
1840 IFPOINT(J%,K%-96)=8 IFY%<K%*F%+32
OR RND(Q%)=1 L%(N%)=0:M%(N%)=-32:ENDPROC
1850 IFL%=64 IFPOINT(J%+64,K%)=P% L%(N%
)=-64:ENDPROC
1860 IFL%=-64 IFPOINT(J%-8,K%)=P% L%(N%
)=64
1870 ENDPROC
1880 :
1890 DEFPROCkilled
1900 GCOL 3,1
1910 FOR Y%=Y% TO 96 STEP-32
1920 MOVE X%,Y%:PRINT D$(Z%)
1930 MOVE X%,Y%-32:PRINT D$(Z%)
1940 FOR N=1 TO 30:NEXT,
1950 FOR N=60 TO 10 STEP -10
1960 SOUND 1,1,N,4:NEXT
1970 MOVE X%,Y%:PRINT D$(Z%)
1980 SOUND 0,1,100,2
1990 D%=D%-1
2000 TIME=0:REPEAT UNTIL TIME>100
2010 ENDPROC
2020 :
2030 DEFPROCend
2040 *FX 15
2050 VDU 4,17,128,28,4,13,15,11,12,26
2060 PROCprint("GAME OVER",352,636,1,3)
2070 TIME=0:REPEAT UNTIL TIME>200
2080 ENDPROC
2090 :
2100 DEFPROCintro
2110 VDU 4,17,128,26,12,17,7
2120 PROCprint("CASTLE "+STR$(S),352,44
4,4,6)
2130 TIME=0:REPEAT UNTIL TIME>300
2140 ENDPROC
2150 :
2160 DEFPROCprint(T$,X,Y,A,B)
2170 VDU 5

```



```

2180 GCOL 0,A:MOVE X,Y:PRINT T$
2190 GCOL 0,B:MOVE X-8,Y-4:PRINT T$
2200 VDU 4
2210 ENDPROC
2220 :
2230 DEFPROCscreen
2240 VDU 4:RESTORE 2680
2250 FOR N%=1 TO S
2260 READ P%,C%,Q%:NEXT
2270 VDU 19,8,4;0:19,9,1;0:19,10,3;0:19
,11,3;0:19,13,5;0:19,15,4;0:19,14,4;0:
2280 VDU 17,143,12,17,130,28,0,31,19,29
,12,26,17,7
2290 PRINTTAB(1,30) "SCORE:";
2300 R%=0:PROCscore(0):VDU 4
2310 PRINTTAB(12,30) CHR$226:D%," ";CHR
$231:"0 ";S
2320 L=1:Y%=96:GCOL0,14
2330 FOR N%=1 TO 30
2340 MOVE 0,Y%:DRAW 1279,Y%
2350 Y%=Y%+L:L=L*1.16:NEXT
2360 VDU 17,128,28,1,28,18,7,12,28,2,7,
4,3,12,28,14,7,17,3,12,26,17,P%,17,128+C
%
2370 PRINTTAB(1,2) STRING$(26,W$+N$);TA
B(18,2) STRING$(26,W$+N$);TAB(1,28) STRI
NG$(18,W$);
2380 FOR N%=7 TO 23 STEP 4
2390 PRINTTAB(2,N%)STRING$(16,W$):NEXT
2400 PRINTTAB(5,2) STRING$(5,W$+N$);TAB
(2,3) STRING$(3,W$);TAB(3,2) W$
2410 PRINTTAB(14,2) STRING$(5,W$+N$);TA
B(15,3) STRING$(3,W$);TAB(16,2) W$
2420 IFS=2 OR S>5 PRINTTAB(10,23) STRIN
G$(5,W$+N$)
2430 M%=6:IF S=3 OR S=5 OR S=7 M%=3:VDU
17,128,28,5,6,14,5,12,26,17,132:PRINTTAB
(4,4)STRING$(11,W$)
2440 IFS=4 OR S=5 OR S=6 VDU31,8,12,224
2450 IFS=7 PRINTTAB(8,8) STRING$(4,W$+N
$)
2460 FOR N%=6 TO 12 STEP 2
2470 VDU 31,N%,M%,224:NEXT
2480 L=RND(-S):COLOUR 8:COLOUR 128
2490 PRINTTAB(3,6) STRING$(5,CHR$225+N$
);TAB(15,6) STRING$(5,CHR$225+N$)
2500 FOR N%=1 TO 4:Y%=6+N%*4
2510 FOR M%=1 TO 4:X%=RND(8)*2+1
2520 PRINTTAB(X%,Y%) STRING$(5-(Y%=22),
CHR$225+N$)
2530 NEXT,:COLOUR 10
2540 PRINTTAB(4,4) CHR$231;TAB(17,4) CH
R$231
2550 FOR N%=1 TO 5:Y%=4-(N%=5)+N%*4
2560 VDU 31,RND(8)*2,Y%,231:NEXT
2570 X%=608:Y%=764:Z%=0
2580 VDU 17,7,17,130,5
2590 GCOL 3,1:MOVE X%,Y%
2600 PRINT D$(Z%):GCOL 3,5
2610 RESTORE 2720:FOR N%=0 TO 1
2620 READ J%(N%),K%(N%),L%(N%)
2630 M%(N%)=0

```

```

2640 MOVE J%(N%),K%(N%):PRINT V$(N%)
2650 NEXT:N%=0:E%=0:R%=0:T%=250:F%=1
2660 ENDPROC
2670 :
2680 DATA 6,4,2,2,4,2
2690 DATA 5,4,3,1,6,3
2700 DATA 6,4,3,2,4,3
2710 DATA 5,4,3
2720 DATA 128,188,64,1088,188,-64
2730 :
2740 DEFPROCinit
2750 DIM D$(1),V$(1),J%(1),K%(1),L%(1),
M%(1)
2760 W$=CHR$224:N$=CHR$10+CHR$8
2770 D$(0)=CHR$226+N$+CHR$227+N$+CHR$22
8
2780 D$(1)=CHR$232+N$+CHR$233+N$+CHR$23
4
2790 V$(0)=CHR$229+N$+CHR$230
2800 V$(1)=CHR$235+N$+CHR$236
2810 ENDPROC
2820 :
2830 DEFPROCenvs
2840 ENVELOPE 1,1,0,0,0,0,0,90,-1,-2,
-3,97,97
2850 ENVELOPE 4,10,20,20,16,1,1,0,0,0
,-4,97,0
2860 ENDPROC
2870 :
2880 DEFPROCchars
2890 VDU 23;10,32;0;0;0;
2900 VDU23,224,-5,-5,249;191,191,175;
2910 VDU23,225,195,195,195,195,-1,195,1
95,195
2920 RESTORE 3050
2930 FOR C%=226 TO 231:VDU 23,C%
2940 FOR B%=0 TO 7:READ V%:VDU V%
2950 NEXT,:RESTORE 3050
2960 PRINT"PLEASE WAIT"
2970 FOR C%=232 TO 237:VDU 23,C%
2980 FOR B%=0 TO 7:READ V%
2990 R%=0:FOR N%=0 TO 3
3000 IF (V% AND 2^N%) R%=R% OR 2^(7-N%)
3010 IF (V% AND 2^(7-N%)) R%=R% OR 2^N%
3020 NEXT:VDU R%:NEXT:PRINTTAB(10,10);2
37-C%:NEXT
3030 ENDPROC
3040 :
3050 DATA 118,42,47,110,109,109,119,123
3060 DATA 120,122,126,122,122,122,122,1
22
3070 DATA 58,123,123,-5,-5,-5,123,50
3080 DATA 126,224,212,190,122,60,24,61
3090 DATA 110,110,40,110,52,124,100,54
3100 DATA 14,17,33,33,65,70,70,56
3110 :
3120 ON ERROR OFF
3130 MODE 7:PRINT':REPORT
3140 PRINT" at line ";ERL':END

```

B

BEEBUG Technical BEEBUG Technical

MASTER COMPACT PANEL UTILITY

Why does the Control Panel utility supplied on the Compact Welcome disc fail to run on my machine?

The Control Panel utility supplied on the Compact Welcome disc enables the default settings stored in EEPROM to be altered. Unfortunately the contents of the EEPROM can become corrupted. If the Control Panel program does not find valid data in the EEPROM it simply stops. If this happens you must press 'R' while turning on the computer. This will reset the EEPROM to the default settings.

UPGRADING THE BBC TO A MASTER 128

Is it possible to upgrade my BBC B to the full specifications of the Master 128?

The simple answer to this is no. However, there is very little that the Master 128 can do and a fully expanded BBC B cannot. The Master 128 is effectively a BBC model B fitted with a 32K RAM card and 4 banks of Sideways RAM. The major difference is that these facilities are included on the Master's circuit board dispensing with the need for additional piggy-backed circuit boards of any description.

Although it is possible to expand your BBC by fitting an ADFS, a 1770 disc interface and some Sideways RAM, many of the Master's extra facilities, such as the internal real time clock, the numeric keypad, and the well-designed ROM cartridge system, cannot be implemented easily.

Bearing in mind that the cost of enhancing a BBC model B will often rise above that of purchasing a Master 128, BEEBUG members may decide to trade in their existing BBC micro for a Master 128 through our special trade in scheme (please phone for details).

CHOOSING A ROM BOARD

I have an unexpanded BBC model B with a few utility ROMs installed. I no longer have any free ROM sockets inside my machine. Which ROM board should I purchase, and what facilities does it provide other than increasing the number of ROM sockets inside the machine?

There are a number of ROM boards available on the market. The first type, such as the Watford Electronics Solderless ROM board and the ATPL ROM board, offer another twelve ROM sockets and the facility of up to 16K bytes of Sideways RAM. Additional switches may be

purchased to write protect the RAM, and a rechargeable battery (to provide battery backed RAM) is an optional extra.

Another type of ROM expansion is the Watford Electronics ROM/RAM board. This board allows another four ROMs to be inserted into the machine. Depending upon which board you purchase you have from 32K bytes to 128K bytes of Sideways RAM resident on the board and the appropriate software to save ROM images to disc and load them back into the Sideways RAM. This does not infringe any copyright laws providing that you have in fact purchased the ROM personally, and that you do not copy somebody else's ROM.

Although this method is not as convenient as having all the ROMs installed in the machine at the same time, it does prevent the usual ROM-clashing and workspace problems associated with sideways ROM software.

If you have a BBC B+ you must purchase a ROM board specifically for your machine such as the ATPL Sidewise + board. This will allow up to ten ROMs to be installed. All of the ROM boards mentioned are available from BEEBUG Retail.



POSTBAG



POSTBAG

ARE WE BEING FRAMED?

Of late, programs for different printer fonts have become fashionable. These are all well and good for greetings cards, menus etc, but somehow the trimmings and borders have been forgotten. So how about a program to draw a variety of borders, scrolls and corners to go with the font programs?

Ernest Ratcliff

This sounds an excellent idea. Maybe a Frame Designer, with some pre-defined frames for good measure, would fit the bill. If any BEEBUG member can help we would certainly be pleased to consider publishing the resulting program.

BEEBUG TOO FAR AHEAD

My main complaint about BEEBUG at the moment is that you are leaving some of us behind. I realise that you are in a 'no win' situation and have to cater for those who have the ability or the time to develop their knowledge, but please give more consideration to computer duffers like me

John Davies

Mr Davies has the hit the nail on the head with his comments. Technical expertise and experience with the Beeb continues to move ahead, and yet there are many happy micro users with neither the time nor inclination to pursue this course.

We do try to provide for less experienced users, particularly through our First Course series, and also through reviews and other articles. We also try very hard to ensure that the use of even the most technical of programs is explained in simple enough terms for all to benefit. It is also our experience that many readers find that through re-reading back issues later on, they find more and more in the magazines that is of interest to them.

GAMES IN BEEBUG -THE CASE FOR....

I read with dismay your intention of cutting down the listed games that have always been prominent in your magazine.

I am a retired person and have a BBC B mainly for relaxation and hobby purposes. I have limited programming knowledge and most technical jargon doesn't really interest me. I rather suspect quite a high percentage of home computer owners are in a similar position to myself.

I like the chore of typing in these games listings etc, as well as making the occasional adjustment which falls within my capabilities. I feel that each copy should contain at least one good quality game or feature of interest.

F.Winter

AND AGAINST

I have been with BEEBUG since Vol.1 No.5, if I can remember back to my first issue. Starting with a Z80, then Z81 I progressed to a BBC model A, subsequently upgraded to a model B. This I gave to the grandchildren to play with, and I am now the proud owner of a Master.

I am 65 years of age and in my view you are quite right in cutting down on games.

- 1.It's a waste of space.
- 2.You can buy as many as 100 on disc for as little as a fiver.
- 3.The best and hardest games cost at the dearest about £15, and I am sure the people who play them can afford this.

R.Avenell

The editorial in the October issue regarding BEEBUG's games content has provoked several letters, and opinions as one might expect are mixed. As stated last month, we shall be publishing one games program in most issues of BEEBUG, providing we have games of a sufficiently high standard. We will also publish reviews of games as appropriate. Perhaps more important, we shall, as we have always done, try to provide a magazine which always contains a wide variety of material, with something of interest for everyone.

B

HINTS HINTS HINTS HINTS HINTS

and tips *and tips* *and tips* *and tips* *and tips*

6502 DECIMAL MODES

Andrew Benham

Whilst decimal mode on the 6502 can be very useful, there is an important fact to realise on BBC Bs and Masters - the operating system does not expect the processor to be in decimal mode, so the D flag must be cleared before calling OS routines, otherwise very strange results occur. Fortunately the interrupt routines are better coded, so there is no need to mask interrupts.

SPACE FIRE

K.V. Felgate

A short routine here for those requiring laser-fire in their space games.

```
10MODE0
20ENVELOPE 2,1,-6,0,0,110
,0,0,50,0,0,-3,100,100
30PRINTTAB(32,30)"Press s
pace"
40IF INKEY=99 PROCshot
50GOTO40
100DEFPROCshot:Y%=10
110SOUND 1,2,190,3
120FOR B%=0 TO 640 STEP 80
130Y%=Y%+55
140FORA%=0TO 1
150MOVE B%-70,Y%-50
160PLOT 6,B%,Y%
170MOVE 1350-B%,Y%-50
180PLOT 6,1280-B%,Y%
190NEXT: NEXT:ENDPROC
```

Lines 10-50 demonstrate the use of the procedure given in lines 100-190. INKEY=99 in line 40 detects the space bar - change this according to your own requirements.

PROTECTING BASIC PROGRAMS

David Garvie

Sometimes it is useful to protect programs so that pressing Break followed by OLD will result in the message "Bad Program". Using the method below, it is still possible to restore the program afterwards.

To protect the program, the first two lines of the program should be:

```
10 Q%=TOP-1
20 ?Q%=0
```

Then to protect the program, simply RUN it. To restore the program, type:

```
?Q%=&FF
```

VDU 22 MODE CHANGE

G.J. Leysmon

If VDU22 is being used to change mode, you should select the mode that uses the most memory first in the program, and then use VDU22 for all subsequent mode changes. This prevents your variables becoming corrupted.

NOT A LANGUAGE

A Garbett

When upgrading from a BBC B to a Master, I noticed that some ROMs wouldn't work, giving the error message "Not a language" if I attempted to run them. The solution is to alter the 6th byte in the ROM from &82 to &C2. Assuming that you have saved a copy of the ROM onto disc as filename <file>, then you can change

the value required by:

```
*LOAD <file> 1000
?&1006=&C2
*SRWRITE 1000 +4000 8000 4
```

The 4 in the last line is the number of the sideways RAM bank to transfer the ROM image into, and can be changed if required. The modified copy could be re-saved to disc, with:

```
*SAVE <file> 1000 +4000 4
```

DEAD CENTRE

Colin M. Pither

How often have you tried to work out the correct values to use with either TAB(n) or TAB(n,n) so you can print text centrally on the screen? This short EXEC file can be used to provide the answer quickly. Once the file has been created, just type *CENTRE, and reply to the two prompts to get the right answer.

```
CLS:P.'"TEXT CALCULATOR"
:I.'"40 or 80 column scr
een: ",s:I.'"Enter text:
"text$:tp=INT(((s-(LEN(t
ext$)))/2)-1):P.'"Tab po
sition =":tp
```

A NEW HELP

Yo Tomita

Typing *HELP. instead of *HELP (don't forget the final dot), will list (in order of ROM socket) the individual HELP information for ALL currently active ROMs.

☐

BEEBUG MEMBERSHIP

Send applications for membership renewals, membership queries and orders for back issues to the address below. All membership fees, including overseas, should be in pounds sterling drawn (or cheques) on a UK bank.

MEMBERSHIP SUBSCRIPTION RATES

£ 7.50 - 6 months (5 issues) UK only
 £14.50 - 1 year (10 issues) UK, BFPO, Ch.I
 £20.00 - Rest of Europe & Eire
 £24.50 - Middle East
 £27.00 - Americas & Africa
 £29.00 - Elsewhere

The new rates are effective from
 1st December 1987
 Renewals will still be accepted at
 the old rates until this date.

BACK ISSUE PRICES

until 30th Nov '87

Volume	Magazine	Cassette	5"Disc	3.5"Disc
1	£0.40	£0.40	-	-
2	£0.40	£0.40	£2	-
3	£0.50	£0.50	£3	-
4	£0.60	£1	£4.25	£4.25
5	£1	£2	£4.75	£4.75
6	£1.30	£3		

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

FURTHER DISCOUNTS

We will allow you a further discount:

Five or more: deduct £0.50 from total
 Ten or more: deduct £1.50 from total
 Twenty or more: deduct £3.50 from total
 Thirty or more: deduct £5.00 from total
 Forty or more: deduct £7.00 from total

POST AND PACKING

Please add the cost of p&p:

Destination	First Item	Second Item
UK, BFPO + Ch.I	40p	20p
Europe + Eire	75p	45p
Elsewhere	£2	85p

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

BEEBUG MAGAZINE is produced by BEEBUG Ltd.

Editor: Mike Williams
 Assistant Editor: Kristina Lucas
 Production Assistant: Yolanda Turuelo
 Membership secretary: Mandy Mileham
 Editorial Consultant: Lee Calcraft
 Managing Editor: Sheridan Williams

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

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £40 per page, but please give us warning of anything substantial that you intend to write. A leaflet 'Notes to Contributors' is available on receipt of an A5 (or larger) SAE. Please submit your contributions on disc or cassette in machine readable form, using "View", "Wordwise" or other means, but please ensure an adequate written description is also included. If you use cassette, please include a backup copy at 300 baud. In all communication, please quote your membership number.

BEEBUG Ltd (c) 1987

Magazine Disc/Cassette

NOVEMBER 1987 DISC/CASSETTE CONTENTS

MODE 7 MENU GENERATOR - completely automatic generation of up to 9 menus each with as many as 15 different entries.

EXPLORING ASSEMBLER (PART 5) - useful example program on four-byte arithmetic in the latest part of our introduction to machine code programming.

DRACULA - another colourful and addictive arcade-style game from Jonathan Temple.

EPSON PRINTER CHARACTER DEFINER - use this program to define new characters for your printer.

BEEBUG WORKSHOP

PRECISION ARITHMETIC (Part 2) - additional routines for high precision multiplication.

HIGH RESOLUTION MODE 0 DUMP - a mode 0 printer dump with twice the normal vertical screen resolution.

AUTO-CONFIRM UTILITY - accidental erasure of files could be a thing of the past with this short utility.

BARRY CHRISTIE VISUALS

FORMATTED LISTINGS - learn how to incorporate VDU codes in your programs for customised listings.

THE MASTER SERIES

STORING VARIABLES AND PROCEDURES IN SWR - make the most of the Master's (and Compact's) sideways RAM by dynamically storing procedures and variables.

DATE STAMPING WORDWISE PLUS - capitalise on the Master's real-time clock for easy date stamping of your Wordwise files.

ARCHIMEDES COMPETITION - the winning program in this very popular competition to win an Archimedes.

MAGSCAN - Bibliography data for this issue of BEEBUG

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

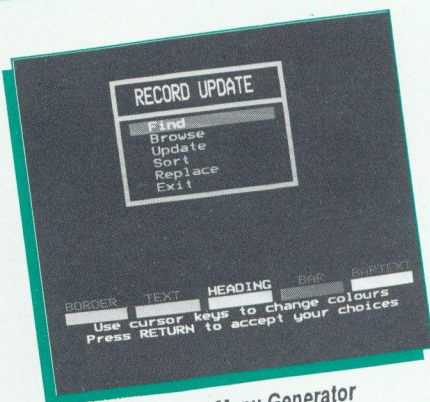
SUBSCRIPTION RATES
6 months (5 issues)
12 months (10 issues)

	5" Disc	3.5" Disc	Cassette
UK ONLY	£25.50	£25.50	£17.00
	£50.00	£50.00	£33.00

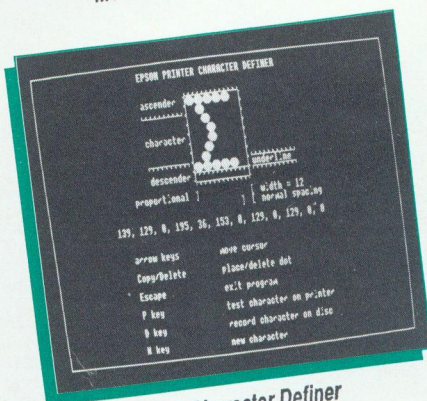
	5" Disc	3.5" Disc	Cassette
OVERSEAS	£30.00	£30.00	£20.00
	£56.00	£56.00	£39.00

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

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



Mode 7 Menu Generator



Epson Character Definer



TRADE-IN YOUR BBC MICRO FOR

A New Master 128, Compact or Archimedes

Don't get behind the times with old equipment. We are offering an efficient trade-in service for people wishing to exchange their BBC Micro for a brand new Master 128, Compact or Archimedes. Depending upon the age of your computer we are able to offer a trade-in discount of up to £225 off the price of new equipment. This would bring the purchase price of a Master 128 down from £435.91 to £210.91.

ARCHIMEDES PRICES Including VAT

Code	Mem Price
0190G 305 Entry System	918.85
0192G 305 Colour System	1148.85
0193G 310 Entry System	1006.25
0195G 310 Colour System	1236.25

Members special offer

When buying a new Archimedes we will give you absolutely free:

"Zarch" the complete Lander game, Printer lead and Lockable 3.5" disc box with 10 discs all worth over £60.

U.K. carrier delivery £7

DISCOUNT

BBC Micro issue 4 or older without DFS £125
BBC Micro issue 7 or newer without DFS £175
BBC Micro issue 4 or older with DFS £175
BBC Micro issue 7 or newer with DFS £225

By DFS we mean an Acorn or Watford single density DFS. Computers must obviously be in good working condition to qualify for this offer.

BEEBUG members receive an additional 5% discount on all our products bringing the trade-in price of a Master 128 to as little as £189.11.

MASTER & COMPACT PRICES Including VAT

Code	Price	Mem Price
0196G		
Compact + Col. Monitor	575.00	546.25
0197G		
Compact + Mono Monitor	440.00	418.00
0198G		
Compact TV System	390.00	370.50
0199G		
Compact Entry System	375.00	356.25
0200G		
Master 128	435.91	414.11

U.K. carrier delivery £7.

If you wish to take advantage of this service please send your machine to us and enclose the order form attached. We recommend that you send your machine by courier (we suggest you check for insurance), and phone us to check availability on new machines. Alternatively, you are welcome to call personally at our showroom (address as below).

BEEBUG

BEEBUG Ltd.,
Dolphin Place, Holywell Hill,
St Albans, Herts AL1 1EX.
For More Information
Telephone 0727-40303

I wish to trade in my BBC Micro Serial No. _____
Issue _____ with/without DFS for a Master 128/Compact Entry/TV/Mono/Colour system/Archimedes.

I enclose a cheque for £ _____ Please debit my Access/Visa by £ _____

My BEEBUG Membership No. is _____ Card No. _____

I claim 5% discount. Card expiry _____

I wish to join BEEBUG and enclose £12.90 (UK only) and claim 5% discount.

Name _____

Address _____

Post Code _____

Signature _____



Archimedes 300 Series Now Available Ex-stock