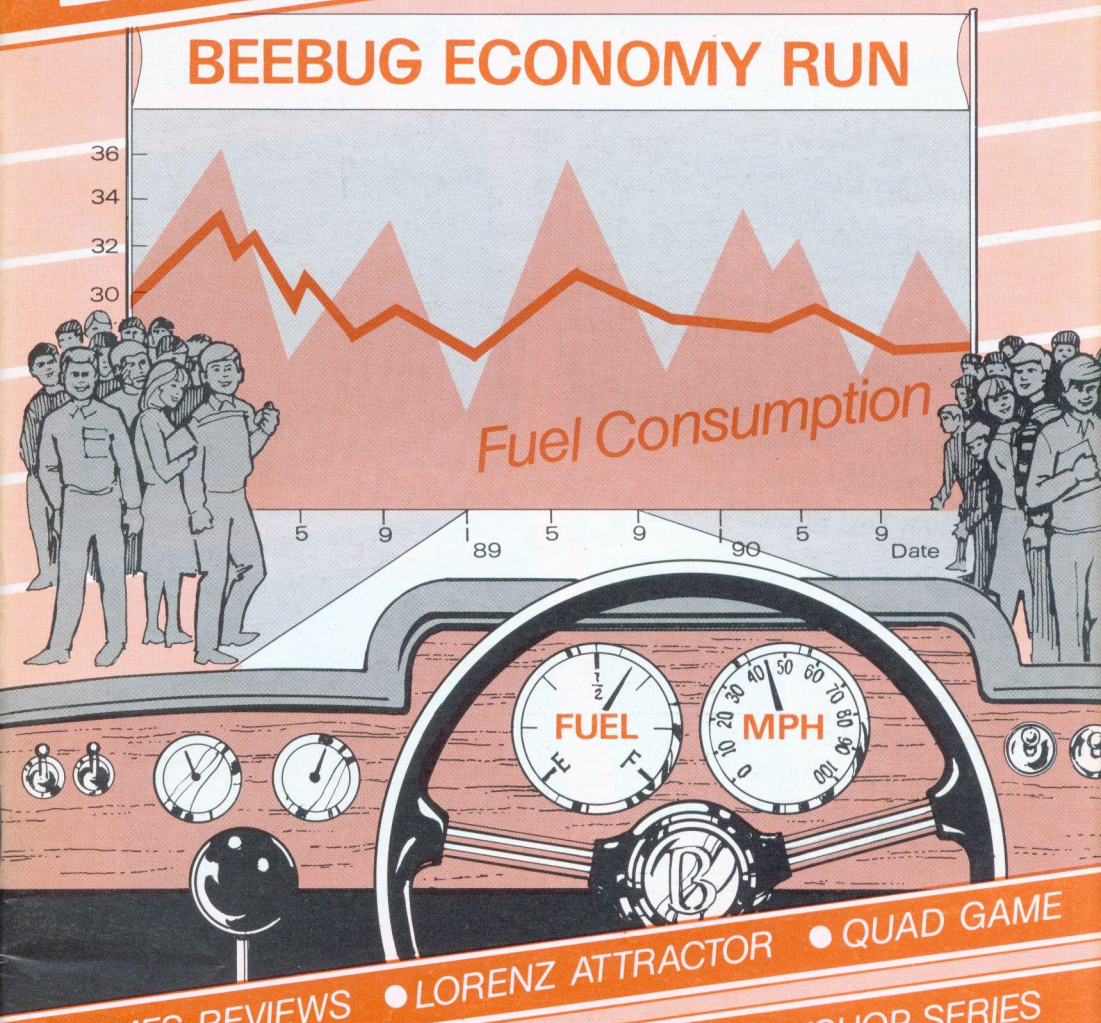
Vol.9 No.4 August/September 1990

BEEBUG

FOR THE BBC MICRO & MASTER SERIES



- GAMES REVIEWS
- LORENZ ATTRACTOR
- QUAD GAME
- MACHINE CODE MONITOR
- NEW WORKSHOP SERIES

BEEBUG Vol.9 No.4 August/September 1990

FEATURES

REVIEWS

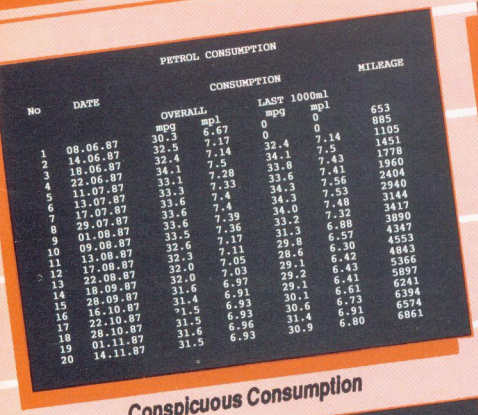
REGULAR ITEMS

HINTS & TIPS

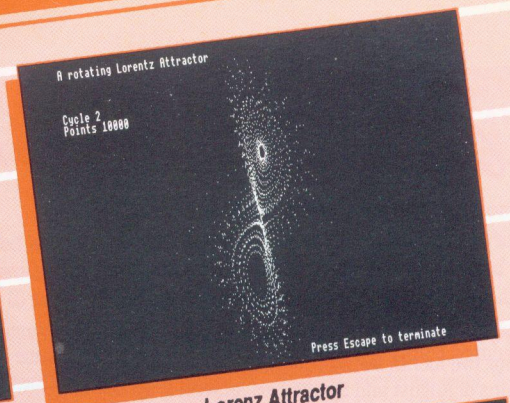
PROGRAM INFORMATION

All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly as printed, and checked carefully. When entering a listing, pay special attention to the difference between the digit one and a lower case l (L). Also note that the vertical bar character (Shift \) is reproduced in listings as |.

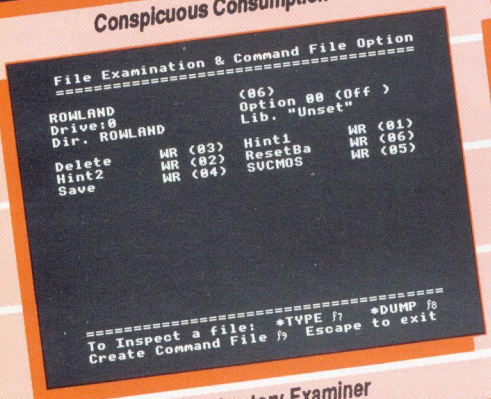
All programs in BEEBUG magazine will run on any BBC micro with Basic II or later, unless otherwise indicated. Members with Basic I are referred to the article on page 44 of BEEBUG Vol.7 No.2 (reprints



Conspicuous Consumption

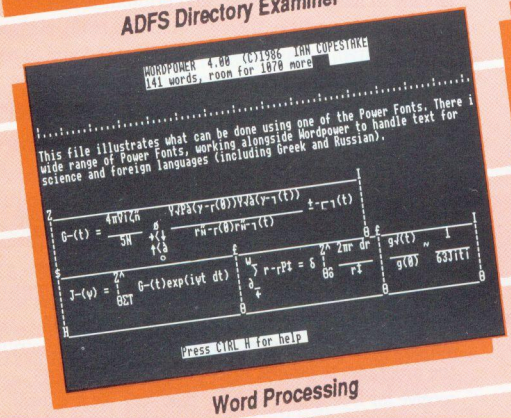


Lorenz Attractor

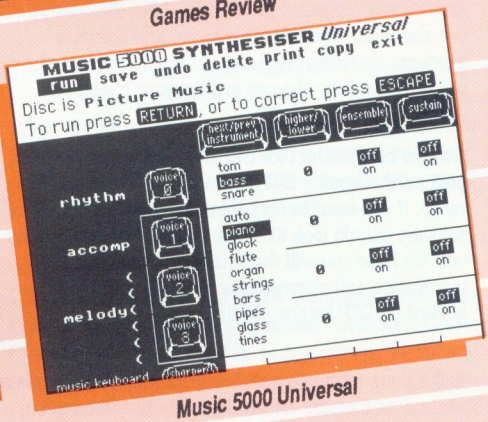


ADFS Directory Examiner

Games Review



Word Processing



Music 5000 Universal

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

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

Program will not function on a cassette-based system.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings

BEEBUG AT THE BBC ACORN USER SHOW

This is our two month issue of BEEBUG, covering both the months of August and September. As such it is the last issue before the BBC Acorn User Show which takes place in early September (see News page for details).

As is our custom, BEEBUG will have a large stand at the show manned by staff from all departments. However, on this occasion our stand will be in two parts back to back, with one half devoted to displays and demonstrations of our latest hardware and software products, while the other half will provide our usual service covering BEEBUG and RISC User magazines, retail sales and technical advice.

So, whatever your needs, if you want to discuss recent articles and programs in BEEBUG (and RISC User), find out about our own latest hardware and software products, or seek advice on your next purchase, there will be somebody who can answer your questions.

A major new product for BEEBUG is Ovation, a professional quality DTP package for the Archimedes range. If you want to find out what good quality DTP software should look like then visit our stand and find out. We hope you will find the experience both informative and interesting, even if you are not yet hooked on DTP (or the Archimedes).

And if you wish to do no more than pass the time of day with us, or just say 'hello', then we will still be more than pleased to see you.

CONTRIBUTING TO BEEBUG

BEEBUG is a magazine which, more than many, relies quite heavily for its content on the efforts and contributions of its readers. We value these highly, as they help to make the magazine the success that it is. Now much as we like to receive contributions, our interest is sometimes a little dented by the fact that some readers obviously enjoy setting us little puzzles. 'Guess whether this disc is 40 or 80 track, DFS or ADFS format' does wear a little thin after a while. So too does the challenge of the reader who would be anonymous: 'Can you find the text file on this disc in which I have put my name and address (if you are lucky), and I'll also let you try and work out which word processor I used'.

Now this no doubt all seems like a little harmless fun, but with the number of contributions which we receive we unfortunately have little time to play these games, much as we might wish to.

Can I therefore ask all intending contributors to send with their disc details of the format of the disc, a list of the files it contains, and the purpose or function of each. If you can supply printed listings, preferably already in magazine format, and printout of any accompanying article, then we would be really pleased. And of course, if you help us to understand and to use your contribution, then we are likely to be in a more sympathetic frame of mind when we come to evaluate it.

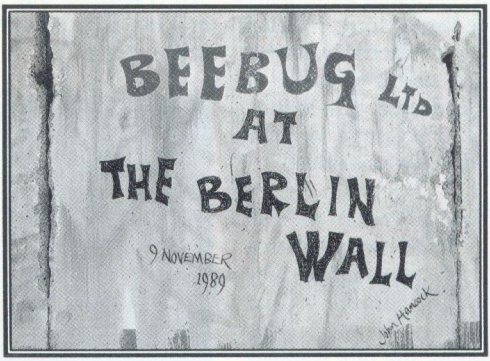
BEEBUG COPYRIGHT

Can I remind you that the copyright of all programs published in BEEBUG magazine, and on the magazine disc belongs to BEEBUG, unless explicitly stated otherwise. That means that as a subscriber to BEEBUG you are fully entitled to use any of our programs. However, this does not entitle you to copy and distribute the programs to others. After all, why should they receive for free what you have paid for? We are most concerned about the possibility of BEEBUG (or RISC User) programs appearing as Public Domain (PD) software, or on bulletin boards for downloading by all and sundry. We would therefore appreciate any information that members may be able to provide to allow us to track down any abuse of our copyright.
M.W.

News News News News News News

BEEBUG MAKES HISTORY

BEEBUG readers will no doubt be as amazed as we were when we received the photograph reproduced here, showing a section of the Berlin Wall, alas (!) so we are told, now removed. The picture was sent by BEEBUG member John Hancock who actually created the slogan himself. He says that it was about 200 yards from the Brandenburg Gate on the East German side of the wall and was seen by many thousands of tourists and East Berliners over several months until that section of the wall was recently dismantled. Apparently the BEEBUG display was unique, and no other company, East or West, took advantage of what must have been one of the world's prime advertising sites. What the East Berlin inhabitants made of it though we shall probably never know.



SHOW TIME

Show time for BBC micro fans is fast approaching. First to kick off the season will be the latest *All Formats Computer Fair* at the New Horticultural Hall, Westminster, London from 4th to 5th September 1990. Bargain prices are expected to attract computer fans from all over Europe, and an Archimedes Village is expected to be a feature of this show.

The *BBC Acorn User Show*, the only show dedicated to the Acorn market, is also due to take place at the Westminster Exhibition Centre (New Horticultural Hall), London from 7th to 9th September 1990. The exhibition organisers claim that the venue will be transformed with wall to wall carpeting and quality stands. Acorn Computers will have a feature stand on the dais overlooking the main area, with on-going information sessions and displays. BEEBUG will also be at this show.

PANDA HELPS CHILDREN IN NEED

Panda Discs has announced a second *Children in Need* Music 5000 disc to be launched at the BBC Acorn User Show. The disc, containing pieces from various programmers, costs £6.00 inc. p&p from Panda Discs, Four Seasons, Tinkers Lane, Brewed, Stafford ST19 9DE. All profits from the disc will be donated to charity.

For Dutch enthusiasts (and everyone else), the *Big Ben Club* in Holland will be holding its 8th annual Yearly Open Day at the Community Centre 'De Kiekmure', Tesselschadelaan 1, Harderwijk, the same venue as was used in 1987. Several British magazines and software houses have regularly supported this event (including BEEBUG), and the day is always full of interest and well attended. Contact for the Big Ben Club is Harry Linsen, Gommerskepel 24, 2151 RA Nieuw-Vennep, Netherlands (FAX 010 (31) 7034-38895).

The *Computer Shopper Show*, 6th to 9th December 1990 at the Wembley Conference Centre, will also feature a major Acorn presence, and an Acorn Village for third party suppliers. Acorn will also be running a conference of seminars alongside the exhibition proper. BEEBUG will also have a stand at this show.

LET'S COMPUTE

Claimed to be the world's first computer comic, *Let's Compute* was launched by Database Publications in July this year. Aimed at the 8 to 14 age group, the monthly magazine aims to present the world of computing to youngsters in comic strip format - learn BBC Basic with cartoon characters ROM and RAM; get the lowdown on Logo. The magazine costs 99p.

ACORN NEWS

Acorn has recently announced details of its latest scheme to allow teachers and teaching support staff to purchase an A3000 or Archimedes 420 system at a discounted price. Full details of this scheme can be obtained from BEEBUG, or any approved Acorn dealer. B

Conspicuous Consumption (Part 1)

Ralph Maltby describes a program which monitors the fuel consumption of your car.

There have been a number of published programs to calculate the fuel consumption of a car. None that I have seen has offered all the features that I wanted, so I set about writing my own. It turned out to be rather longer than I had expected but it is at least reasonably easy to use and has proved to give useful warnings of deterioration in car performance. The results can be shown in tabular and graphical formats; the latter will be described in the next issue.

The program was originally written to take advantage of the Shadow RAM on the Master. However, it is possible to use it on a model B by splitting into two programs, with the second part covering the plot routines. Listing 1 gives the Master version excluding the plot routines for the graphical display. Modifications to suit the model B are in Listing 2. The next issue will cover the plot routines for both.

Fuel consumption is computed both as an overall figure from the first entry up to the present, and over the last 1000 miles or so. It is tabulated in miles per gallon and in miles per litre; dedicated Europeans could easily modify the program to show litres per 100 km, by replacing line 2740 with:

```
2740 D$=STR$(28409/mpgall%(entry%)):D$
=LEFT$(D$,4)
```

and similarly altering line 2770 but using F$ and mpglast% instead. Table 1 shows the data in tabular format; Figure 1 shows the graphical display which will be described in Part 2.

The "1000 miles or so" is chosen arbitrarily as a compromise between the excessive scatter produced by calculating at each tank-full and the need to show reasonably short-term variations. It is assumed that at regular intervals you will fill the tank to a well-defined level (e.g. completely full or until the pump cuts out); these I have termed reference Fills. At each reference fill the program goes back 1000 miles and then seeks the previous reference fill. It uses the data there for the start of the short-term calculation. How far back it has to go depends on how often reference fills are recorded but the result is better than making the calculation between subsequent fills.

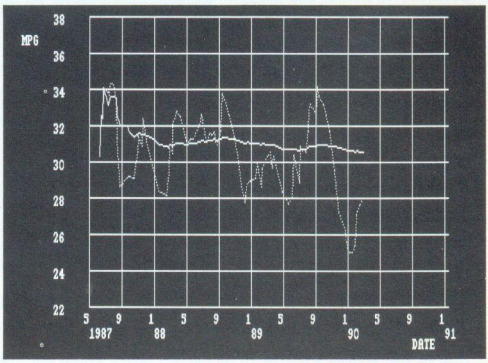


Figure 1

The example shown in Figure 1 illustrates the noticeable short-term effects of intermittent ignition trouble. Here the smooth line is the cumulative fuel consumption from new, and the jagged line is for the previous 1000 miles. The dips in December are caused by cold weather and shorter journeys, combined, in December 88, with the ignition trouble mentioned above. The latter can be seen again in April 89. The catastrophic dip in November 89 coincides with the conversion to "green" petrol during a cold snap.

USING THE PROGRAM

The most important point to consider, of course, is that you must keep a reliable record of mileage, dates and fuel bought. You must also try to keep your reference fills fairly frequent and regular. In between these reference fills you can have part-fills if this is more convenient, which must also of course be logged. Since there are still some pumps which deliver in gallons, the fuel added can be entered as

gallons or litres. It doesn't matter when this data is entered into the program, as long as you keep a careful record of all fills until the data file is updated. Each data entry session must finish with a reference fill, however. Obviously, the calculations produced by the program will not be accurate until the data is up to date.

The program starts by asking if the data file disc is ready. This can be the program disc if there is plenty of space on it. It then asks for the "car reference" - this will be used as the name for the data file. Anything will do but part of the car registration number is a suitable identifier.

If you are starting a new file you will be asked for the initial data; the program needs to know the date and mileage at the starting point, and obviously the tank must be filled to your chosen reference level at this point.

Once this has been done a menu appears, and there are plenty of prompts to guide you through. You will often see "Is this OK?" after entering data, because I found that I made a significant numbers of errors when entering a large amount of data. There is an *Amend* option for those errors which slip through.

Table 2 is a copy of the car log for the first few points of the example shown in Table 1. You may find it useful to use this for a trial run in order to get used to the system. On this month's magazine disc there is a data file called *Consump*, which contains about 90 entries, starting with those in Table 1.

PETROL CONSUMPTION

No	DATE	CONSUMPTION				MILEAGE
		OVERALL		LAST 1000ml		
		mpg	mpl	mpg	mpl	
1	08.06.87	30.3	6.67	0	0	653
2	14.06.87	32.5	7.17	0	0	885
3	18.06.87	32.4	7.14	32.4	7.14	1105
4	22.06.87	34.1	7.5	34.1	7.5	1451
5	11.07.87	33.1	7.28	33.8	7.43	1778
6	13.07.87	33.3	7.33	33.6	7.41	1960
7	17.07.87	33.6	7.4	34.3	7.56	2404
8	29.07.87	33.6	7.4	34.3	7.53	2940
9	01.08.87	33.6	7.39	34.0	7.48	3144
10	09.08.87	33.5	7.36	33.2	7.32	3417
11	13.08.87	32.6	7.17	31.3	6.88	3890
12	17.08.87	32.3	7.11	29.8	6.57	4347
13	22.08.87	32.0	7.05	28.6	6.30	4553
14	18.09.87	32.0	7.03	29.1	6.42	4843
15	28.09.87	31.6	6.97	29.2	6.43	5366
16	16.10.87	31.4	6.91	29.1	6.41	5897
17	22.10.87	31.5	6.93	30.1	6.61	6241
18	28.10.87	31.5	6.93	30.6	6.73	6394
19	01.11.87	31.6	6.96	31.4	6.91	6574
20	14.11.87	31.5	6.93	30.9	6.80	6861

Table 1

Specimen Entries

Date	Miles	Fuel	Reference?
29.05.87	394	Full	Start
08.06.87	653	38.8L	Y
14.06.87	885	29.7L	N
18.06.87	1105	31.1L	Y
22.06.87	1451	41.3L	Y
11.07.87	1778	49.0L	Y
13.07.87	1960	23.7L	Y
15.07.87	2282	25.9L	N
17.07.87	2404	32.0L	Y
20.07.87	2745	9.1G	N
29.07.87	2940	31.0L	Y

Table 2

PROGRAM INFORMATION

If you are intending to use the program on a model B you should be careful to retain the line numbers as used in the Master version of Listing 1. Listing 2 contains all the alterations and additions for the model B.

When running the Master version, one data file is created and this records the essential input data. Dates are converted to integers and the fuel bought is multiplied by 100 so that all the data can be recorded as integers. This economises on disc space and facilitates searching for amendments to data. All fuel data are converted to gallons for the records and for the purposes of calculation. A second data file is required for the model B to carry over the calculated data to the Plot program. Provision is made for this in Listing 2 which automatically prefixes the file name with a 'b'. Note that CLOSE#0 is avoided throughout because of the bug in the Master's DFS.

The procedure names are intended to be self-explanatory and there are no obscure constructions. The only function - FNkey(K$) - is simply a convenient way to deal with Z$=GET$; K$ is usually "YyNn" but "CcMm" also occurs. The first two characters return TRUE, the second two return FALSE and anything else is ignored.

Listing 1

```
 10 REM            >FuelCon
 30 REM Version B1.0
 40 REM Author  Ralph Maltby
 50 REM BEEBUG  Aug/Sept1990
 60 REM Program subject to copyright
 70 :
100 *SHADOW 0
```

```
  110 MODE7
  120 PRINTTAB(9,9);CHR$131;CHR$141;"FUE
L CONSUMPTION";TAB(9);CHR$131;CHR$141;"F
UEL CONSUMPTION";TAB(11,12)CHR$134"Maste
r version":Z=INKEY(150):CLS
  130 REPEAT PRINTTAB(0,15);CHR$131"Is t
he data file disc ready?(Y/N)":UNTIL FNk
ey("YyNn")
  140 DIM date$(200),date%(200),mile%(20
0),newfuel%(200),newfuel(200),mpgall%(20
0),mpglast%(200),bignewfuel%(200)
  150 PROCinit
  160 ON ERROR MODE7:PROCerror
  170 REPEAT PROCmenu
  180 CLS:PRINTTAB(5,14)CHR$131;"Do you
want to exit?(Y/N)"
  190 UNTIL FNkey("YyNn")
  200 PRINTTAB(5,14)CHR$131;"Program end
ed"STRING$(12," "):END
  210 :
 1000 DEF PROCinit
 1010 CLS:entry%=0:fuel=0
 1020 newfuel(0)=0:flag%=0
 1030 CLS:PRINTTAB(0,10);CHR$131;"Enter
car reference: ":INPUTTAB(22,10)file$:CL
S
 1040 PRINTTAB(7,10)CHR$131"Do you alrea
dy have";TAB(3,12)CHR$131"this data file
 on disc?(Y/N)":IF FNkey("YyNn") THEN CL
S:PROCreload(7):ENDPROC
 1050 CLS:PRINTTAB(4,10)CHR$134"Space wi
ll be reserved for";TAB(10)CHR$134"the d
ata file";TAB(2,15)CHR$134"Please make s
ure that there is ";TAB(8)CHR$134"room o
n your disc"
 1060 PRINTTAB(4,24)CHR$131"Press SPACE
BAR to continue":Z=GET
 1070 CLS:PRINTTAB(8,10)CHR$131"Are you
sure? (Y/N)";TAB(6,12)CHR$131"(it will w
ipe any file"'SPC9CHR$131"of the same na
me)":IF FNkey("YyNn") THEN PROCopenfiles
:CLS:PROCenter ELSE CLS:GOTO1040
 1080 ENDPROC
 1090 :
 1100 DEF PROCmenu
 1110 REPEAT CLS:PRINTTAB(8,3);CHR$129;"
OPTIONS AVAILABLE"
 1120 PRINTTAB(10,5);CHR$131;"1.Enter Da
ta"
 1130 PRINTTAB(10,7);CHR$131;"2.Tabulate
 results"
 1140 PRINTTAB(10,9);CHR$131;"3.Plot Dat
a"
 1150 PRINTTAB(10,11);CHR$131;"4.Amend D
ata"
 1160 PRINTTAB(10,13);CHR$131;"5.Exit"
 1170 PRINTTAB(7,19);CHR$134;"Enter opti
on number"
 1180 Z$=GET$:IF Z$="1" THEN CLS:PROCent
er
 1190 IF Z$="2" THEN CLS:PROCtab
 1200 IF Z$="3" THEN CLS:PROCplot
 1210 IF Z$="4" THEN CLS:PROCamend
 1220 UNTIL Z$="5":ENDPROC
 1230 :
 1240 DEF PROCenter
 1250 IF entry%=0 THEN PROCnew
 1260 entry%=0:REPEAT entry%=entry%+1:UN
TIL date%(entry%)=0:entry%=entry%-1
 1270 PRINT"Date of last reference fill
";date$(entry%)
 1280 entry%=entry%+1:lastentry%=entry%
 1290 CLS:PROCenterfuel:CLS:PROCenterdat
e:PROCentermile:PROCspan:PROCcalc
 1300 PROCsaveline(1):ENDPROC
 1310 :
 1320 DEF PROCnew
 1330 PRINT''''"Enter starting date"
 1340 PROCdate
 1350 INPUT''''"Enter starting mileage  "
mile%(0)
 1360 PRINT"Is this OK?"
 1370 IF NOT FNkey("YyNn") THEN CLS:PRIN
T"Re-enter miles":GOTO1350:ELSE CLS
 1380 bignewfuel%(0)=0:PROCsaveline(0)
 1390 ENDPROC
 1400 :
 1410 DEF PROCenterfuel
 1420 LOCAL Q$
 1430 IF flag% word$="next" ELSE word$="
last"
 1440 PRINTTAB(4,2);CHR$131;"Enter each
re-fill quantity"
 1450 PRINTTAB(4,3);CHR$131;"since ";CHR
$134;date$(entry%-1);CHR$131;"and end wi
th":PRINTTAB(8);CHR$131;word$+" referenc
e fill":PRINT
 1460 INPUT "Quantity ",newfuel
 1470 PRINT"      Is this OK?"
 1480 IF NOT FNkey("YyNn") THEN CLS:PRIN
T"Re-enter fuel":GOTO1460
 1490 PRINT"Gallons(G) or Litres(L)?"
 1500 IF FNkey("GgLl")THEN Q$="G" ELSE Q
$="L"
 1510 PRINT newfuel;Q$;"   Is this OK?(Y
/N)"
```

```
 1520 IF NOT FNkey("YyNn") THEN CLS:PRIN
T"Re-enter units":GOTO1490
 1530 IF Q$="L" THEN newfuel=.22*newfuel
 1540 newfuel(entry%)=newfuel(entry%)+ne
wfuel
 1550 bignewfuel%(entry%)=100*newfuel(en
try%)
 1560 PRINTCHR$131;"Was that a reference
 fill?"
 1570 IF NOT FNkey("YyNn") GOTO1460
 1580 ENDPROC
 1590 :
 1600 DEF PROCenterdate
 1610 PRINT'':PRINTCHR$131;"Enter date o
f reference fill":PRINTTAB(7)CHR$131"aft
er "CHR$134date$(entry%-1)
 1620 PROCdate
 1630 ENDPROC
 1640 :
 1650 DEF PROCentermile
 1660 INPUT"Enter miles at reference fil
l ",mile%(entry%)
 1670 PRINT"Is this OK? (Y/N)"
 1680 IF NOT FNkey("YyNn") THEN CLS:PRIN
T"Re-enter miles":GOTO1660
 1690 IF mile%(entry%)<1000 THEN mpglast
%(entry%)=0:begin%=0:temp%=entry%
 1700 ENDPROC
 1710 :
 1720 DEF PROCspan
 1730 temp%=entry%
 1740 IF mile%(entry%)<=1000+mile%(0) TH
EN mpglast%(entry%)=0:begin%=0:temp%=ent
ry%:ENDPROC
 1750 REPEAT entry%=entry%-1
 1760 UNTIL mile%(temp%)-mile%(entry%)>=
1000
 1770 begin%=entry%:entry%=temp%
 1780 ENDPROC
 1790 :
 1800 DEF PROCcalc
 1810 fuel=fuel+newfuel(entry%)
 1820 fuel%=100*fuel
 1830 plusfuel=0:entry%=begin%
 1840 REPEAT entry%=entry%+1
 1850 plusfuel=plusfuel+newfuel(entry%)
 1860 UNTIL entry%=temp%
 1870 mpgall%(entry%)=100*((mile%(entry%
)-mile%(0))/fuel)
 1880 IF mile%(entry%)<1050 THEN mpglast
%(entry%)=0 ELSE mpglast%(entry%)=100*((
mile%(entry%)-mile%(begin%))/plusfuel)
 1890 ENDPROC
 1900 :
 1910 DEF PROCreload(mode%)
 1920 LOCAL d$:U%=0:*FX229,1
 1930 PRINTTAB(0,12)CHR$134"Please wait
for loading to complete"
 1940 PRINTTAB(0,19)CHR$131"Loading entr
y":PRINTTAB(4)CHR$131"No."
 1950 U%=OPENUPfile$:entry%=-1
 1960 REPEAT entry%=entry%+1
 1970 IF mode%=7 THEN PRINTTAB(8,20)CHR$
131;entry%
 1980 INPUT#U%, date%(entry%),mile%(entr
y%),bignewfuel%(entry%)
 1990 newfuel(entry%)=bignewfuel%(entry%
)/100
 2000 d$=STR$(date%(entry%))
 2010 IF LEN(d$)=5 THEN d$="0"+d$
 2020 date$(entry%)=LEFT$(d$,2)+"."+MID$
(d$,3,2)+"."+RIGHT$(d$,2)
 2030 IF entry%>0 THEN PROCspan:PROCcalc
 2040 UNTIL date%(entry%)=0
 2050 lastentry%=entry%-1
 2060 CLOSE#U%:U%=0:*FX229,0
 2070 ENDPROC
 2080 :
 2090 DEF PROCsaveline(A%)
 2100 LOCAL d$:U%=0:*FX229,1
 2110 U%=OPENUPfile$
 2120 PTR#U%=(entry%)*15
 2130 PRINT#U%,date%(entry%),mile%(entry
%),bignewfuel%(entry%)
 2140 CLOSE#U%:U%=0:*FX229,0
 2150 IF A%<2 GOTO 2200
 2160 newfuel(entry%)=bignewfuel%(entry%
)/100
 2170 d$=STR$(date%(entry%))
 2180 IF LEN(d$)=5 THEN d$="0"+d$
 2190 date$(entry%)=LEFT$(d$,2)+"."+MID$
(d$,3,2)+"."+RIGHT$(d$,2)
 2200 entry%=0:fuel=0
 2210 IF A%>0 THEN fuel=0:FOR entry%=1 T
O lastentry%:PROCspan:PROCcalc:NEXT
 2220 ENDPROC
 2230 :
 2240 DEF PROCtab
 2250 VDU14:PRINTTAB(7)CHR$134;"Press SH
IFT to scroll"
 2260 PRINTTAB(6,2);"DATE";TAB(15);"CONS
UMPTION"
 2270 PRINTTAB(16,3);"OVERALL";TAB(28,3)
;"LAST1000ml"
 2280 PRINTTAB(15,4);"mpg";TAB(22,4);"mp
l";TAB(29,4);"mpg";TAB(36,4);"mpl" : PRI
```

```
NT
 2290 FOR entry%=1 TO lastentry%
 2300 PRINTTAB(0);entry%;TAB(4);date$(en
try%);TAB(14);mpgall%(entry%)/100;TAB(21
);INT(.22*mpgall%(entry%))/100;TAB(28);m
pglast%(entry%)/100;TAB(35);INT(.22*mpgl
ast%(entry%))/100
 2310 NEXT
 2320 PRINT'':VDU15:PRINTTAB(4);CHR$134;
"Do you want a print out?(Y/N)":IF NOT F
Nkey("YyNn") THEN CLS:ENDPROC
 2330 CLS
 2340 PRINTTAB(0,5)CHR$131;"Is the print
er connected and ON?"
 2350 IF NOT FNkey("YyNn") CLS:GOTO2320
 2360 PROCprinttab:ENDPROC
 2370 :
 2380 DEF PROCdate
 2390 INPUT "DAY",day$,"MONTH (1-12)",mo
nth$,"YEAR 19"year$
 2400 IF LEN(month$)<2 THEN month$="0"+m
onth$
 2410 IF LEN(day$)<2 THEN day$="0"+day$
 2420 date$(entry%)=day$+"."+month$+"."+
year$
 2430 date%(entry%)=VAL(day$+month$+year
$)
 2440 PRINTdate$(entry%);"       Is this
OK?"
 2450 IF NOT FNkey("YyNn") THEN CLS:PRIN
T"Re-enter date":GOTO2390
 2460 ENDPROC
 2470 :
 2480 DEF FNkey(K$)
 2490 LOCALK%:REPEAT:K%=INSTR("@"+K$,GET
$) DIV 2:UNTIL K%:K%=K%-2
 2500 =K%
 2510 :
 2520 DEF PROCprinttab
 2530 LOCAL A$,C$,D$,E$,F$,G$
 2540 VDU2,1,27,1,69
 2550 PRINTTAB(29)"PETROL CONSUMPTION"
 2560 PRINT':PRINTTAB(4)"No"TAB(11)"DATE
"TAB(41)"CONSUMPTION"TAB(69)"MILEAGE"
 2570 PRINT:PRINTTAB(29)"OVERALL"TAB(49)
"LAST 1000ml"
 2580 PRINTTAB(27)"mpg"TAB(35)"mpl"TAB(4
9)"mpg"TAB(57)"mpl"
 2590 FOR entry%=1 TO lastentry%
 2600 A$=STR$(entry%)
 2610 IF entry%>9 AND entry%<100 THEN A$
=" "+A$
 2620 IF entry%<10 THEN A$="  "+A$
 2630 C$=STR$(mpgall%(entry%)/100)
 2640 C$=LEFT$(C$,4)
 2650 D$=STR$(INT(.22*mpgall%(entry%))/1
00):D$=LEFT$(D$,4)
 2660 E$=STR$(mpglast%(entry%)/100)
 2670 E$=LEFT$(E$,4)
 2680 F$=STR$(INT(.22*mpglast%(entry%))/
100):F$=LEFT$(F$,4)
 2690 G$=STR$(mile%(entry%))
 2700 PRINTTAB(3)A$TAB(9)date$(entry%)TA
B(26)C$TAB(35)D$TAB(48)E$TAB(57)F$TAB(69
)G$
 2710 NEXT:PRINT'''':VDU3:CLS
 2720 ENDPROC
 2730 :
 2740 DEF PROCamend
 2750 flag%=1
 2760 PRINTTAB(11,7)CHR$131"Enter the nu
mber"TAB(4,9)CHR$131"of the entry to be
corrected"
 2770 PRINTTAB(0,19)CHR$134"This can be
found from the Tabulation"TAB(9,22)CHR$1
34"(<ESCAPE> for MENU)":INPUTTAB(18,14),
entry%
 2780 CLS:PRINTTAB(0,12)CHR$134"Now ente
r correct data for that entry":Z=INKEY(2
00):CLS
 2790 newfuel(entry%)=0
 2800 PROCenterfuel:CLS:PROCdate
 2810 PROCentermile:PROCsaveline(2)
 2820 ENDPROC
 2830 :
 2840 DEF PROCopenfiles
 2850 LOCAL N%
 2860 CLS:PRINTTAB(0,12)CHR$131"Please w
ait for files to be configured"
 2870 pad%=0:U%=OPENOUTfile$
 2880 FOR entry%=1 TO 600
 2890 PRINT#U%,pad%
 2900 IF entry%MOD12=0 PRINTTAB(3,23)"*"
TAB(2,23)" "
 2910 IF entry%MOD12=6 PRINTTAB(2,23)"*"
TAB(3,23)" "
 2920 NEXT:entry%=0:CLOSE#U%:U%=0
 2930 ENDPROC
 2940 :
 2950 DEF PROCerror
 2960 IF ERR=17 ENDPROC
 2970 VDU22,7:PRINT"ERROR"
 2980 IF U% THEN CLOSE#U%
 3000 REPORT:PRINT" at line ";ERL:END
 3010 ENDPROC
```

Continued on page 12

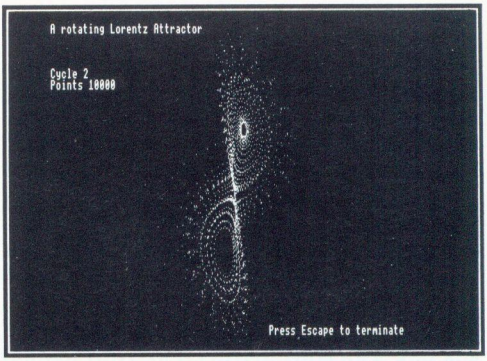
The Lorenz Attractor

by Danny Fagandini

As Jim Vernon explained in *Order Out Of Chaos* (BEEBUG Vol.8 No.9), the high speed iteration of simple algebraic equations on personal computers has made it possible for all of us to see mathematical mysteries that have lain hidden to us since the dawn of time. Chaos is now to hand, but it is a very special form of chaos, one with an internal order that we see expressed by Nature in snowflakes, ferns, the convolutions of sea shells and the battles of the species for the survival of the fittest. Deep down in Jim Vernon's programs there lies 4.669201660910... the bifurcation number. We know not from where it comes nor what it implies. It is known as the Feigenbaum number after the mathematician who first spotted its universality on the Los Alamos computers in the seventies.

Edward N. Lorenz had rather more rudimentary computer facilities at his disposal at MIT in the sixties. He was studying meteorology and wanted to model turbulence in the atmosphere. It was early days; computers were lumbering beasts and tedious to program. Lorenz cut, pruned and trimmed his equations until he had a simple skeleton model that his colleagues could not take seriously. A few lightweight equations could not possibly tell them anything they did not already know. The fact that numbers could be crunched from them in reasonable time seemed trivial.

However, Lorenz persevered and kept stressing the strange patterns that his crunching seemed to suggest. Looking at columns of numbers is an acquired taste, so few paid much attention. Those that did, however, could not escape what was obvious once prejudice had been overcome: simple equations, iterated thousands of times, can give totally unpredictable results. Far from working towards finer and better weather forecasting, Lorenz was proving that to be an impossible goal. Might not the flutter of a butterfly's wings in Chicago be the start of a deep depression in the Atlantic? We could no longer be sure that that had to be pure fantasy.



Screen display produced by listing 1

Years later mathematicians laid down the jargon of Chaos and talked at length about Attractors, Phase Space and Julia and Mandelbrot Sets. An attractor is a pattern that results from lengthy iteration of a set of equations. It can be a single point, a pattern of many points, a loop or even infinity. Lorenz was given his very own, personal, attractor by general acclaim. Modern computer graphics bring it home to us in colour and at speed, tantalisingly in two dimensions only. At least, so far...

Listing 1 is a program based on a version of Lorenz's equations published in the Canadian journal ALGOrithm. There is plenty of phase space left for experiment. The variable P% in line 120 holds the number of points plotted in each cycle. You can try altering this between 500 and 15000. Each cycle is plotted in a different colour; for a monochrome display, simply omit line 260.

Listing 1

```
  10 REM Program Lorenz
  20 REM Version B1.0
  30 REM Author  Danny Fagandini
  40 REM BEEBUG  Aug/Sep 1990
  50 REM Program subject to copyright
  60 :
 100 MODE0
 110 PROCtitle
```

```
  120 P%=5000:REM try 500 to 15000
  130 h=0.01:z=0.06:x=0.06:y=0.06
  131 frac=8/3:k=16:cycle=0:q=0
  140 VDU23,1,0;0;0;0;
  150 REPEAT
  160 PRINTTAB(5,8)"Press Space to rotat
e, else any other key to start"
  170 q=GET:UNTIL q
  190 VDU29,620;512;
  200 REPEAT C=0
  210 REPEAT cycle=cycle+1
  220 CLS:PROCtitle
  230 PRINTTAB(5,5)"Cycle ";cycle
  240 PRINTTAB(5,6)"Points ";P%*cycle
  250 C=C+1:IF C=4 C=5
  260 VDU19,1,C,0,0,0
  270 FOR N%=1 TO P%
  280 PROCformula
  290 IFq=32 THEN h=0.02:frac=7/3:PROCro
tate
  300 PLOT 69,INT(k*X),INT(k*Y)
  310 NEXT:UNTIL C=7
  330 UNTIL FALSE
  340 END
  350 :
 1000 DEF PROCformula
 1010 x=x+10*(y-x)*h:X=x
 1020 y=y+(28*x-y-x*z)*h:Y=y
 1030 z=z+(x*y-z*frac)*h
 1040 r=SQR(x*x+y*y)
 1050 ENDPROC
 1060 :
 1070 DEF PROCrotate
 1080 PRINTTAB(5,1)"A rotating Lorenz At
tractor    "
 1090 PRINTTAB(5,7)" "
 1100 alpha=ASN(ABS(y)/r)+(cycle+2)*(SGN
(x*y))*PI/20
 1110 X=r*COS(alpha)*SGN(x)
 1120 Y=r*SIN(alpha)*SGN(y)
 1130 ENDPROC
 1140 :
 1150 DEF PROCtitle
 1160 PRINTTAB(5,1)"The Lorenz Attractor
"
 1170 PRINTTAB(45,28)"Press Escape to te
rminate"
 1180 ENDPROC
```

Conspicuous Consumption (continued from page 10)

Listing 2

```
   10 REM          >ModCon
   20 REM Mods to FuelCon for model B
  100 REM Omit up to and including this
line
  120 PRINTTAB(9,9);CHR$131;CHR$141;"FUE
L CONSUMPTION";TAB(9);CHR$131;CHR$141;"F
UEL CONSUMPTION";TAB(11,12)CHR$134"Model
 B version":Z=INKEY(150):CLS
 2870 U%=OPENOUTfile$
 2890 PRINT#U%,0
 2990 IF V% THEN CLOSE#V%
 3020 :
 3030 DEFPROCplot
 3040 CLS:PRINTTAB(4,5)CHR$134"Calculate
d data will now be":PRINTTAB(10)CHR$134"
SAVED onto disc"
 3050 PRINTTAB(3,11);CHR$131;"Is data-fi
le disc ready? (Y/N)":IF FNkey("YyNn")=0
 ENDPROC
 3060 PRINT'''':PROCsavedata
 3070 CLS:PRINTTAB(1,5)CHR$134"Plot prog
ram will now be CHAINED"
 3080 PRINTTAB(3,11);CHR$131;"Is program
 disc ready? (Y/N)":IF FNkey("YyNn") THE
N PAGE=&1300:CHAIN"FPLOT-B"
 3090 ENDPROC
 3100 :
 3110 DEFPROCsavedata
 3120 *FX229,1
 3130 V%=OPENOUT("Temp")
 3140 PRINT#V%,lastentry%
 3150 PRINT TAB(2,20);CHR$131;"Recording
 entry No. "
 3160 FOR entry%=0 TO lastentry%
 3170 PRINTTAB(22,20);CHR$131;entry%
 3190 PRINT#V%,date$(entry%),mile%(entry
%),mpgall%(entry%),mpglast%(entry%)
 3200 NEXT:CLOSE#V%:V%=0:*FX229,0
 3210 CLS:PRINTTAB(8,12);CHR$131;lastent
ry%;" entries recorded"
 3230 PRINTTAB(4,20);CHR$134;"Press SPAC
E BAR to continue":Z=GET
 3240 entry%=entry%-1
 3250 ENDPROC
```

ADFS Directory Examiner and Command File Creator

Use Vivian Stevens' excellent utility to browse through the files on any ADFS disc, and form command files for subsequent loading of selected items.

One of the great advantages of the Advanced Disc Filing System (ADFS) is its hierarchical structure, which allows a sensible classification of files within appropriately named directories. But that property in itself can be a drawback unless you are well organised and you are prepared to be disciplined in the organisation of your discs.

```
    ADFS Directory Organiser and Map
    ================================

  Progeny Directories : 22

   3 CLARK            4 GIBBONS
   5 GORMAN           6 GROMBLE
   7 HERBERT          8 HOBSON
   9 HOLTON          10 JUPE
  11 KEATES          12 LASRUK
  13 MURPHY          14 PIBWORTH
  15 RANKIN          16 ROWLAND
  17 SEFTON          18 SHARMAN
  19 STEVENS         20 TAYLOR
  21 TEMPLEMAN       22 Williams

  Path-Length : 1

   $

  ================================
  Options:  Change disc f0 *CAT f1 *CDIR f2
  *BACK f4   Root f5   Parent f6  Progeny f7
```

When you catalogue a disc, you see only the files and directories present in the current directory. Since this can contain up to 47 sub-directories, and up to 128 levels of directories are permitted, they can all too easily become unmanageable. Remembering the location of a particular file can become a major problem.

The program listed here, *DirExam+*, will allow you to browse through the directory structure of an ADFS disc, examining files as you choose. In addition it offers a powerful facility by allowing the creation of command files which can then be used to load pre-selected files. This is likely to prove of particular interest to those using Edit on a Master for program development. A command file is effectively a list of files which can be used to load the contents of those files when the command file is executed (see later). To use the program, simply type it in, save it, and then run it.

USING THE PROGRAM

DirExam+ allows you to browse through the various directories on the disc, examining any file you choose, and if you wish, you can add the file details to a command file which the program sets up. The program is controlled by using the cursor keys together with Return. The available options are always shown at the foot of the screen.

The display includes only directory information (parent directory, current directory, and any progeny directories, together with the directory path), filtering the catalogue data to remove other filenames. It is therefore easier to get an overall impression of the hierarchy structure.

At any point a full catalogue may be requested, which will include a display of filenames in the current directory, and it is from this level that you have the option to inspect a chosen file (in *TYPE or *DUMP format), or to append its details to your command file, if one has been created. A file may be selected by reference number or by name, but in the latter case upper and lower case letters must be specified correctly for a match. A command file is created in the root directory on pressing f9 for the first time, and this is given the default name, *!Command_F*. Subsequent presses of the f9 key will prompt you for a filename, and append this file's details to the command file.

Pressing Escape at any time will take you to the directory display for the current directory, while Shift-Escape will terminate the program completely (but see later).

USING COMMAND FILES

If you are a keen programmer, then one way to organise your programming activity is to build up a library of useful routines in ASCII format (using *SPOOL or whatever) on disc, which can be appended to the program on which you are working.

An example might be an often used INPUT routine like the FNinput in DirExam+. When you begin coding your basic program, it is useful to browse through the routines in your personal library, and append them to your growing program file.

Inspecting the contents of files individually is easy enough, but if you need to append several files to one another, moving between directories, cataloguing directories and loading the relevant files can become tedious.

A far better solution is to set up a command file which will do the work for you. This can be done with *DirExam+* as described, and the resulting command file will contain the names of the files selected and the necessary commands for loading the contents of these files into Edit.

To set up a command file requires that the ASCII values of the appropriate key presses be written to "!Command_F" on disc. For example, to create the file from scratch, Edit has to be called, so the sequence of key presses written to the command file is "*EDIT"+CHR$(13). This is held as a string variable, cf$, and is written to disc character by character.

When a file is to be appended to those already resident in Edit, I have chosen to have the new file added "at the bottom" rather than having it inserted at the start. Therefore the key press Ctrl-Cursor-Down, or CHR$(174) is inserted into the command file before a new file is appended. The upshot of this convention is that, when the command file is executed, the cursor appears at the beginning of the last file to be appended, which looks a bit strange when you first encounter it.

Any file can, in theory, be loaded by a command file, but in DirExam+, only files in the top 23 levels of the hierarchy can be included in the command file. This is because the path (the root to the current directory) is also held in a string variable prior to being written to the disc. The string length is limited to a maximum 255 characters, and each directory name could have ten characters in it. ("$."+22 directories (or a terminal filename) of ten characters, each with a "." separator added, =255). This is no great disadvantage, in fact, and if you wish, you could get around the limitation by deleting "+(10 AND l%<23)" in line 1550. If you do this, you will have to discipline yourself to keeping the names of the directories through which you move to reach your chosen file, to the bare minimum for easy identification, and obviously you will have to end each directory name with the "*" wildcard character.

There is no limit as to how far down the hierarchy you may browse; you may go as far as the ADFS will allow directories to exist. The variable level% indicates the level (vertically) at which the current directory is found on the disc; for the root directory, *level%=0*. At each level the program displays the path to the current directory. This is worked out each time it is needed rather than being held as a string variable, and is not therefore limited by how many directories are encountered en route. Note that paged mode is utilised to give you the opportunity to inspect the display at leisure, and you will need to press the Shift key in order to scroll some displays.

On leaving the program (press Shift-Escape to do this) you will be presented with the options of immediately executing the command file or of loading the file into the editor for direct alteration. If you opt for the latter, do not forget that the characters you see will look strange because they represent key presses, and I would suggest reading the relevant section in the reference manual (R 16.1) if you propose to use this facility.

PROGRAM NOTES

The listed version of the program uses the ON-PROC structure, which works in Basic IV, but will not be recognised by earlier versions of Basic. If you have an earlier machine with ADFS but do not have Edit, you may well find the directory-browsing features of this program useful, in which case you will have to rewrite lines 180 and 2060, using an IF-THEN-ELSE structure to redirect the program. Note also the space after *BACK in line 1500.

```
   10 REM Program DirExam+
   20 REM Version B1.4
   30 REM Author  Vivian Stevens
   40 REM BEEBUG  Aug/Sept 1990
   50 REM Program subject to copyright
   60 :
  100 ON ERROR GOTO 220
  110 MODE134:PROCinit
  120 PROCscreen("    ADFS Directory Org
aniser and Map   ")
  130 A%=0:Y%=0:X%=&70:IF(USR(&FFDA)AND&
FF)<>8PRINTTAB(7,10)"For use with ADFS o
nly"'':VDU23,1,1;0;0;0;:END
  140 PROCdisc
  150 REPEAT
  160 IFkey$=CHR$(27)PROCscreen("    ADF
S Directory Organiser and Map   ")
  170 PROCcontrol:PROCkey(key$)
  180 ON ASC(key$)-245 PROCdisc,PROCcat_
command,PROCcdir,PROCback,PROCroot,PROCp
arent,PROCchild
  190 UNTIL FALSE
  200 END
  210 :
  220 IF ERR=17 AND NOT INKEY-1 THEN key
$=CHR$(27):GOTO150
  230 IF INKEY-1 THEN PROCend:END
  240 MODE7:REPORT:PRINT" at line ";ERL
  250 *FX4
  260 END
  270 :
 1000 DEF PROCcontrol
 1010 CLS
 1020 IF level%=0 parent$="None" ELSE pa
rent$=level$(level%-1)
 1030 PRINT"   Parent  Directory   : ";p
arent$'"   Current Directory   : ";level
$(level%)'
 1040 PROCfindprogeny
 1050 PROCprompt("  (Paged Mode-Press Sh
ift to scroll)")
 1060 PROCprogeny:PROCpath
 1070 control%(0)=245+(1 AND level%=0 AN
D cfp%=0)
 1080 control%(1)=247:control%(2)=245+(3
ANDlevel%<47)
 1090 control%(3)=245+(4ANDback%<>0)
 1100 control%(4)=245+(5ANDlevel%>0)
 1110 control%(5)=245+(6ANDlevel%>0)
 1120 control%(6)=245+(7ANDcount%>0)
 1130 PROCprompt(FNmenu):key$=""
 1140 FOR N%=0 TO 6
 1150 OSCLI"KEY"+STR$(N%)+" "+CHR$(contr
ol%(N%)+(control%(N%)=245))
 1160 key$=key$+CHR$(control%(N%))
 1170 NEXT:*FX21,0
 1180 ENDPROC
 1190 :
 1200 DEF FNfileinfo(f$)
 1210 LOCAL X%,Y%
 1220 $output%=f$
 1230 cntrl%?0=output%MOD256
 1240 cntrl%?1=output%DIV256
 1250 X%=cntrl%MOD256
 1260 Y%=cntrl%DIV256
 1270 A%=5:A%=(USR(&FFDD)AND&FF)
 1280 =A%
 1290 :
 1300 DEF FNgetfile
 1310 LOCAL name$
 1320 VDU26:VDU23,1,1;0;0;0;
 1330 PRINTTAB(5,23);:name$=FNinput(10,3
3,126)
 1340 VDU23,1,0;0;0;0;
 1350 PRINTTAB(5,22)"Is the filename cor
rect? (Y/N)"
 1360 PROCkey("yYnN")
 1370 =name$
 1380 :
 1390 DEF FNsetup
 1400 LOCAL ch%,cf$,ptr%
 1410 ch%=OPENOUT("$.!Command_F")
 1420 cf$="*EDIT"+CHR$(13)
 1430 FOR ptr%=1 TO LEN(cf$)
 1440 BPUT#ch%,ASC(MID$(cf$,ptr%,1))
 1450 NEXT
 1460 cfp%=PTR#ch%:CLOSE#ch%
 1470 =cfp%
 1480 :
 1490 DEF FNmenu
 1500 menu$=" Options:  Change disc "+CH
R$(control%(0))+" *CAT "+CHR$(control%(1
))+" *CDIR "+CHR$(control%(2))+"  *BACK
"+CHR$(control%(3))+"   Root "+CHR$(cont
rol%(4))+"   Parent "
 1510 menu$=menu$+CHR$(control%(5))+"  P
rogeny "+CHR$(control%(6))+" "
 1520 =menu$
 1530 :
 1540 DEF FNmenu2(p%,level%)
 1550 menu$=" To Inspect a file:  *TYPE
"+CHR$(253)+"   *DUMP "+CHR$(254)+"   "+
MID$("Add to Command File Create Command
 File ",((20 AND p%)+1),20)+CHR$(245+(10
 AND level%<23))+"  Escape to exit  "
 1560 =menu$
 1570 :
 1580 DEF FNname(count%,name$)
 1590 LOCAL p%,ptr%
 1600 ptr%=-1:p%=INSTR(name$,"*")
```

```
 1610 IF p% name$=LEFT$(name$,p%-1)
 1620 REPEAT:ptr%=ptr%+1
 1630 UNTIL ptr%=count%+1 OR INSTR(dir$(
ptr%),name$)=1
 1640 IFptr%>count%name$=""ELSEname$=dir
$(ptr%)
 1650 =name$
 1660 :
 1670 DEF FNappend(file$)
 1680 LOCAL ch%,cf$,ptr%,path$
 1690 ch%=OPENUP("$.!Command_F")
 1700 path$=FNpath
 1710 cf$=CHR$(174)+CHR$(146)+path$+file
$+CHR$(13)
 1720 PTR#ch%=cfp%
 1730 FOR ptr%=1 TO LEN(cf$)
 1740 BPUT#ch%,ASC(MID$(cf$,ptr%,1))
 1750 NEXT:cfp%=PTR#ch%:CLOSE#ch%
 1760 =cfp%
 1770 :
 1780 DEF FNpath
 1790 LOCAL path$,ptr%
 1800 ptr%=0:path$="$."
 1810 REPEAT
 1820 ptr%=ptr%+1
 1830 IF ptr%<=level% path$=path$+LEFT$(
level$(ptr%),INSTR(level$(ptr%)," ")-1)+
"."
 1840 UNTIL ptr%>level%
 1850 =path$
 1860 :
 1870 DEF FNinput(len%,low%,hi%)
 1880 LOCAL input$
 1890 block%?0=buffer%MOD256
 1900 block%?1=buffer%DIV256
 1910 block%?2=len%
 1920 block%?3=low%
 1930 block%?4=hi%
 1940 A%=0
 1950 Y%=block%DIV256:X%=block%MOD256
 1960 CALL&FFF1:input$=$buffer%
 1970 =input$
 1980 :
 1990 DEF PROCcat_command
 2000 PROCscreen(" File Examination & Co
mmand File Option")
 2010 REPEAT
 2020 PROCprompt("  (Paged Mode-Press Sh
ift to scroll)")
 2030 OSCLI"."
 2040 PROCprompt(FNmenu2(cfp%=0,level%))
 2050 PROCkey(LEFT$(CHR$(27)+CHR$(253)+C
HR$(254)+CHR$(255),4+(level%>22)))
 2060 IF key$<>CHR$(27) ON ASC(key$)-252
 PROCtype,PROCdump,PROCcommfile
 2070 UNTIL key$=CHR$(27)
 2080 ENDPROC
 2090 :
 2100 DEF PROCfindprogeny
 2110 LOCAL N%,A%,C%,X%,Y%,file$
 2120 FOR N%=0 TO 9 STEP 3:control%!N%=0
:NEXT
 2130 count%=0
 2140 REPEAT
 2150 control%?1=string%MOD256
 2160 control%?2=string%DIV256
 2170 control%?5=1
 2180 X%=control%MOD256:Y%=control%DIV25
6
 2190 C%=0:A%=8:CALL(&FFD1)
 2200 IF control%?5<=0 ?(string%+1+?stri
ng%)=13:file$=$(string%+1):IF FNfileinfo
(file$)=2 count%=count%+1:dir$(count%-1)
=file$
 2210 UNTIL control%?5>0
 2220 ENDPROC
 2230 :
 2240 DEF PROCprogeny
 2250 LOCAL ptr%,pdir$
 2260 IF count%=0 pdir$="None" ELSE pdir
$=STR$(count%)
 2270 PRINTTAB(3,3)"Progeny Directories
: ";pdir$'
 2280 VDU28,0,19,39,8
 2290 ptr%=0
 2300 REPEAT
 2310 IF ptr%<>count% ptr%=ptr%+1:PRINTT
AB(23-19*(ptr%MOD2)+(ptr%>9));ptr%;" ";d
ir$(ptr%-1);
 2320 UNTIL ptr%=count%:PRINT':VDU28,0,1
9,39,3.
 2330 ENDPROC
 2340 :
 2350 DEF PROCpath
 2360 LOCAL ptr%
 2370 PRINT"   Path-Length : ";level%+1'
 2380 ptr%=0:PRINT"    $";
 2390 VDU28,0,19,39,VPOS+3-(2ANDcount%>=
22)
 2400 REPEAT
 2410 ptr%=ptr%+1
 2420 IF ptr%<=level% PROCpath1
 2430 UNTIL ptr%>level%:VDU28,0,19,39,3
 2440 ENDPROC
 2450 :
 2460 DEF PROCpath1
 2470 IF POS+LEN(LEFT$(level$(ptr%),INST
R(level$(ptr%)," ")-1))>37 PRINT'"    ";
```

```
 2480 PRINT".";LEFT$(level$(ptr%),INSTR(
level$(ptr%)," ")-1);
 2490 ENDPROC
 2500 :
 2510 DEF PROCcdir
 2520 LOCAL name$
 2530 PROCprompt("  Input new directory
title: ")
 2540 VDU26
 2550 REPEAT
 2560 VDU23,1,1;0;0;0;
 2570 PRINTTAB(29,22);
 2580 name$=FNinput(10,33,126)
 2590 VDU23,1,0;0;0;0;
 2600 UNTIL name$<>"" AND INSTR(name$,"*
")=0
 2610 OSCLI"CDIR "+name$
 2620 VDU28,0,19,39,3
 2630 ENDPROC
 2640 :
 2650 DEF PROCback
 2660 LOCAL temp%
 2670 *BACK
 2680 temp%=back%:back%=level%+1
 2690 level%=temp%-1
 2700 ENDPROC
 2710 :
 2720 DEF PROCroot
 2730 *DIR $
 2740 back%=level%+1:level%=0
 2750 ENDPROC
 2760 :
 2770 DEF PROCparent
 2780 *DIR ^
 2790 back%=level%+1:level%=level%-1
 2800 ENDPROC
 2810 :
 2820 DEF PROCchild
 2830 LOCAL name$,dir%
 2840 PROCprompt("  Input directory to b
e made ""current""    Name or Number:")
 2850 VDU26
 2860 REPEAT
 2870 REPEAT
 2880 VDU23,1,1;0;0;0;
 2890 PRINTTAB(18,23);:name$=FNinput(10,
33,126)
 2900 VDU23,1,0;0;0;0;
 2910 UNTIL NOT(VAL(name$)>count% OR nam
e$="0")
 2920 IF VAL(name$)=0 name$=FNname(count
%,name$) ELSE name$=dir$(VAL(name$)-1)
 2930 UNTIL name$<>""
 2940 OSCLI"DIR "+name$
 2950 back%=level%+1:level%=level%+1
 2960 level$(level%)=name$
 2970 VDU28,0,19,39,3
 2980 ENDPROC
 2990 :
 3000 DEF PROCtype:PROCtask("TYPE "):END
PROC
 3010 :
 3020 DEF PROCdump:PROCtask("DUMP "):END
PROC
 3030 :
 3040 DEF PROCtask(task$)
 3050 LOCAL name$
 3060 REPEAT
 3070 PROCprompt("     Which file do you
 wish to see?")
 3080 name$=FNgetfile
 3090 UNTIL key$="y" OR key$="Y"
 3100 PROCprompt("    *"+LEFT$(task$,4)+
"ing file; Shift to scroll")
 3110 CLS:OSCLItask$+name$:VDU7
 3120 PROCprompt(" End of File - Press S
PACE to continue")
 3130 REPEAT:UNTIL GET=32:CLS
 3140 ENDPROC
 3150 :
 3160 DEF PROCcommfile
 3170 LOCAL name$
 3180 IF cfp%=0 cfp%=FNsetup ELSE REPEAT
:PROCprompt("     Which file do you wish
 to add?"):name$=FNgetfile:UNTIL key$="y
" OR key$="Y":cfp%=FNappend(name$)
 3190 ENDPROC
 3200 :
 3210 DEF PROCkey(p$)
 3220 REPEAT
 3230 key$=GET$
 3240 UNTIL INSTR(p$,key$)
 3250 ENDPROC
 3260 :
 3270 DEF PROCprompt(prompt$)
 3280 VDU26
 3290 PRINTTAB(0,22);SPC(79);TAB(0,22);p
rompt$;
 3300 VDU28,0,19,39,3
 3310 ENDPROC
 3320 :
 3330 DEF PROCscreen(title$)
 3340 LOCAL a$:VDU26
 3350 VDU23,1,0;0;0;
 3360 COLOUR1:VDU19,1,3;0;19,0,4;0;14
 3370 a$=CHR$32+STRING$(38,"=")
 3380 PRINTtitle$'a$TAB(0,21)a$
 3390 VDU28,0,19,39,3:CLS
 3400 ENDPROC
```

Continued on page 34

BEEBUG Survey
Word Processing

We conclude our survey of word processors for the BBC micro and Master series by looking at two later offerings, *View Professional* and *Wordpower*. View Professional is interesting in that it also functions as an integrated spreadsheet and database as well as a word processor. It was written by Mark Colton (the author of View) and originally published by Acorn. However, the current version of View Professional is now marketed by Colton Software. View Professional is also interesting as in the form of Pipedream it is also available in versions for the Cambridge Computer's Z88, the PC, and as Pipedream 3 for the Archimedes range.

Wordpower has proved particularly popular in the educational market, and has also carved out something of a niche for itself for applications which require different character sets, as with foreign languages and in scientific word processing. The use of so-called *Powerfonts* has considerably enhanced the value of this word processor.

As before, we have asked two committed users of View Professional and Wordpower to give us their views. Again, all prices quoted include VAT.

View Professional in use

Kai S. Ng explains the reasons for his choice of View Professional as his preferred word processor.

View Professional V2.0 (Colton Software) £75.38

About 8 months ago when I was looking around for a computer upgrade to provide me with the tool to write up research project articles, the length of which can vary from a few pages to dozens of chapters, I thought very hard before I purchased View Professional (see also BEEBUG Vol.6 No.4) to complement the 7 year old model B system I own.

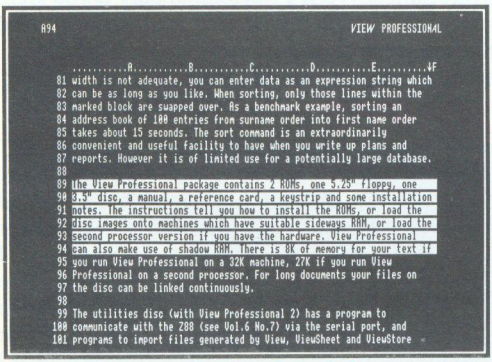
I use View Professional in mode 3. Cursor movements which bring new text to the display window area are performed very rapidly because View Professional uses hardware scrolling when appropriate, or if a screen update is necessary, the process responds immediately to any keyboard entry. Cursor movements are slower if you want some columns and rows permanently displayed.

All features expected of a word processor are implemented satisfactorily: reset line width and tabs; justify left, centre and right; word wrap; case swap; insert, erase, delete characters or lines; move, copy and delete a block; overtype or insert mode for text entry; word count, search and replace, and more. When marking a block, a whole line is highlighted, so juggling text in your document usually requires additional editing.

View Professional uses the red function keys to give immediate access to the most frequently used commands. Alternatively, the commands can be typed in at the top of the screen after invoking the command line interpreter with the backslash '\' character, (double backslash will give you a backslash in your text). The command mnemonics take some getting used to, but the reference card and the generous 190 page spiral bound manual provide excellent support. Immediately the command interpreter is invoked, you can enter any * command to run your operating system utilities. You can even *BUILD a sequence of View Professional commands and execute them in a batch - exactly like running an MS-DOS batch file on a PC. This is great, for example, when you want to make consistent changes in your documents. Along similar lines you can define up to two command sequences to be activated by Shift-Copy and Ctrl-Copy.

The editing screen has line numbers down the left side and tab stop markings along the top. A ruler facility is not provided, though one can be made by defining a command sequence.

Pressing Escape takes you to the options screen which you will need to refer to for your page format settings. The single line running header and footer can be aligned in 3 portions; left, centre and right.



View Professional worksheet

Page breaks show up distinctively on the screen. In a multi-file document, the page breaks of all previous files in the chain are noted by the firmware so that the last line of the current page can be determined from the last break, which might be half way down the text of the preceding file.

View Professional's filing commands can do clever things like merge files, save text in plain format, spool the output and exchange spreadsheet values with suitably dimensioned files.

The line and tab settings in View Professional make up a regular array of 'slots' to provide an underlying worksheet to your word processing. Pressing the Tab key will move the cursor to the next slot on the line. In this way all slots can be referred to uniquely by a co-ordinate, thus forming the basis of a spreadsheet. And View Professional immediately becomes a spreadsheet by simply invoking the expression command to allow you to assign a mathematical formula to your current slot.

All the principal features of a spreadsheet are implemented: arithmetic and logical functions (and scientific functions as well if you have a second processor); defining the leading '£' and trailing '%' characters; decimal places; position alignment; minus signs can be shown in accountancy brackets or scientific "-" form; slot references can be automatically updated when expression slots are copied.

Using View Professional as a spreadsheet differs from using it as a word processor only in the way a slot is filled. When a line of text produced by the expression command is wider than the column width, its display is truncated. On the other hand a line of text entered directly is displayed over the subsequent vacant slots until it is truncated by either a non-empty slot or by the wrap-width for the current column.

The most important feature I have not yet mentioned is the sort command. This powerful command turns an area of your text into a mini-database of sheet format. Each record assumes one line and one line only, with the fields divided up by the columns. If the column width is not adequate, you can enter data as an expression string which can be as long as you like. When sorting, only those lines within the marked block are swapped over. As a benchmark example, sorting an address book of 100 entries from surname order into first name order takes about 15 seconds. The sort command is an extraordinarily convenient and useful facility to have when you write up plans and reports. However, it is of limited use for a potentially large database.

The View Professional package contains 2 ROMs, one 5.25" floppy, one 3.5" disc, a manual, a reference card, a keystrip and some installation notes. The instructions tell you how to install the ROMs, or load the disc images onto machines which have suitable sideways RAM, or load the second processor version if you have the hardware. View Professional can also make use of shadow RAM. There is 8K of memory for your text if you run View Professional on a 32K machine, 27K if you run View Professional on second processor. For long documents your files on the disc can be linked continuously.

The utilities disc (with View Professional 2.0) has a program to communicate with the Z88 (see Vol.6 No.7) via the serial port, and programs to import files generated by View, ViewSheet and ViewStore (though not in the other direction). Also included is a printer driver editor utility which facilitates up to 8 types of highlights, 10 character translations and microspacing.

Though View Professional offers more facilities than most single word processor packages available on any microcomputer, Colton Software has no plan to produce a spelling checker for the Beeb (though Pipedream on the Arc does include one). However, Computer Concepts' Spellmaster can be used to check any saved text file. If you have been contemplating a desktop publishing package but you do not have an up-market printer, you may find View Professional very attractive to revamp your existing system. Because of the wide variety of features available, I would recommend View Professional only if you were already familiar with the basic concepts of word processing.

Wordpower

Jim Hudson explains why he finds Wordpower the best word processor for his needs.

Wordpower (Ian Copestake Software) £40.65
Powerfonts from £21.85
Educational site licence (inc. one Powerfont)
£100 plus VAT, until 31st August 1990.

Wordpower will run on Electrons, BBC Bs, Compacts, Masters and the Archimedes range. It is compatible with DFS, ADFS and Network filing systems. My own particular setup is a Master + Epson FX-1000, so my version consists of a set of ROM images of the word processor, the enhanced printing package (more later) and my chosen 'Powerfont' (again see later). It is not necessary to order the printer or Powerfont packages but, as I will explain, they are very useful indeed. Everything arrived neatly bundled with a ream of manuals, function key strips, advice and quick reference sheets. The manual is very clearly written and has a first rate index.

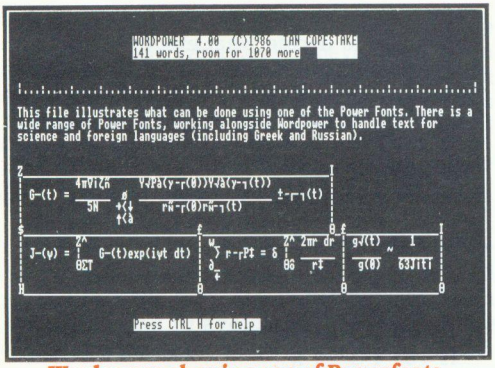
Wordpower, the printer package and Powerfonts are loaded into sideways RAM and the 'startup' disc is auto-booted. After a couple of seconds you are faced with Wordpower's only screen, which can be set up precisely how you want it.

The majority of my word processing is connected with science teaching; it is vital that I have immediate access to scientific characters, which can be displayed on-screen and printed. Powerfonts have been designed to overcome just such a problem. However, the catalogue did not contain precisely what I wanted, so one telephone call, a covering letter and 2 days of waiting resulted in my own 'personalised' Powerfont, with no compromise. Life would have been easier if I had required a Ukranian font with native keyboard layout; that was in stock! This does show two of Wordpower's great strengths; its ability to fit an individual's requirements and the fact that Ian Copestake is never more than a telephone call away, always willing to help and advise. I cannot think of another such commercial package where it is so easy to speak to the author.

The word processor itself has all of the features that I would expect. Text is continually formatted, the section where the cursor is operating from can be held in the middle of the screen, no fiddling around at the bottom of the screen with Wordpower. It is very straightforward to delete, insert or move characters, words, lines or blocks. When large areas of work are being deleted Wordpower always asks for confirmation, as indeed it does when I attempt to save a file. In addition to such basic features, Wordpower has a number of powerful utilities included as standard.

My favourite is its ability to split the screen into two windows and have a different file

loaded into each one. Naturally you can work on each file, as if it was the only one present. I have also made a great deal of use of the *Edit* facility: here a file is automatically backed up, under a slightly different name ensuring that I do not accidentally delete the wrong 1500 words (a quick glance at the manual shows that I must also try and use the mail-merge package one day: it can access data from all common databases, or rely on the one that can be generated within Wordpower itself). Before printing I check my spelling through "Spellmaster"; it does not work in the immediate mode, but is otherwise compatible.



Wordpower showing use of Powerfonts

The above facilities are controlled by the function keys and a set of Ctrl keys; for example, Ctrl-O toggles between overtype and insert mode. If this all gets a bit tough to remember then Crtl-H brings up a series of "Help Screens". Failing that I occasionally have to consult the manual. It is very rare that I now have to take such a drastic step! The keys do seem to fall logically to hand, giving the word processor a feeling of quiet efficiency; it is very 'Hudson friendly'.

Printing is equally versatile; text can be printed from memory or disc, and the latter option means that I can continue to work on a different document while one is being printed. Page breaks, headings, footers and the number of copies required can be readily specified. Items such as page lengths and line spacings are controlled from within the text, using plain English commands. However, when I access Powerfont NTQ, the quality of the printout is taken into a different league.

The printing package is based on PMS's Multifont NTQ. All of its features are retained, so I can print text that is proportionally/ microspaced, of varying widths and heights, with a variety of backgrounds. The addition of the Powerfont ensures that my scientific characters are printed to the same quality. Colleagues are amazed at the standard of the printout. Additional fonts can be obtained from Ian Copestake, and of course all of PMS's Multifonts and Publisher Fonts are compatible.

Wordpower can be used at a variety of levels; I have had pupils come in, auto-boot the disc, type the odd paragraph, print and leave. At the other end of the scale I am, at present, composing a 6th form test, using four fonts and microspaced printing. Wordpower is not perfect, but I have yet to find a word processor for the Master that gets close to it in terms of usability and quality of output.

SUMMARY

That concludes our survey of word processors for the BBC micro and Master series. We hope that you have found it to be of interest. No doubt enthusiastic users of some of the products which we have covered may feel our authors have omitted to describe what they consider to be some very important feature. If readers have any particular comments on these (or any other word processors) which may be of use to others then we would be pleased to hear from them.

ADDRESSES

Colton Software,
Broadway House, 149 St Neots Road,
Hardwick, Cambridge CB3 7QJ.
Tel. (0223) 211472

Ian Copestake Software,
10 Frost Drive, Wirral,
Merseyside L61 4XL.
Tel. 051-648 6287

BEEBUG WORKSHOP

Searching (Part 1)

Bernard Hill re-launches our popular Workshop series with the first part of a discussion of searching techniques, and the solution to the aptly name travelling salesman problem.

A great many practical problems are concerned with searching for answers. Sometimes the search is through a list of stored values, as in a database, and this has been covered in previous Workshop articles (for example, see BEEBUG Vol.4 No.2) In this article we start to look at a different type of search: the search for a solution to a problem as opposed to a pre-stored answer.

PERMUTATIONS

Occasionally we have a problem in which we need to search through all the different arrangements of some things and select the best in some sense or other. As a trivial example, let's consider anagrams (in the widest sense). An algorithm to run through all the permutations of some set of numbers (or characters) is a bit tricky - until you've seen the answer! In fact the key to writing this algorithm is of course recursion. A permutation of N items is just a matter of selecting the first item and permuting the rest; then changing the first item and permuting the rest until we've had all our possible choices of first item. Actually in programming terms it's easier to keep the last item fixed as follows:

A B C D E F

1. Keep the F at the back and list all permutations of ABCDE (e.g. ABCDEF, BACDEF etc.).

2. Now swap F and A, list all permutations of the new first five (FBCDE) and then swap back.

3. Repeat step 2 exchanging the B with the F, then the C with the F, then D with F and finally E with F.

All we need to do before we code up this program is to decide what exactly we do with our permutation when we have it. If we want to write out the complete set of anagrams of a word then we just print out the letters in their new order. But in a general problem we would include some calculation or evaluation function here. We shall look at this later.

Here is the algorithm:

```
 10 INPUT word$
 20 N=LEN(word$)
 30 DIM letter$(N)
 40 FOR i=1 TO N
 50 letter$(i)=MID$(word$,i,1)
 60 NEXT
 70 PROCperm(N)
 80 END
 90 :
100 DEF PROCperm(n)
110 LOCAL i
120 IF n=1 THEN PROCwrite:ENDPROC
130 PROCperm(n-1)
140 FOR i=1 TO n-1
150 PROCswap(i,n)
160 PROCperm(n-1)
170 PROCswap(i,n)
180 NEXT
190 ENDPROC
200 :
210 DEF PROCswap(a,b)
220 LOCAL t$
230 t$=letter$(a)
240 letter$(a)=letter$(b)
250 letter$(b)=t$
260 ENDPROC
270 :
280 DEF PROCwrite
```

```
290 LOCAL j
300 FOR j=1 TO N
310 PRINT letter$(j);
320 NEXT
330 PRINT
340 ENDPROC
```

This program prints out the N! (factorial N, N!=1x2x3..xN) permutations - albeit in a strange-looking order. Before we leave this program, however, it's worth a second look.

For the sake of clarity the program uses procedures and long variable names, with integer (%) variables. But we can speed up this program by an astonishing factor of 3 and and I think that this in itself is worth a digression.

All programs are made faster by using the resident integer variables whenever possible. In this case the gain is about 25%, but the largest gain is to be found in the use of static strings rather than an array of dynamic strings. Instead of line 30 we will use:

```
30 DIM W 255
```

since that's the maximum string length. Now setting $W=word$ we can access the individual letters as W?0, W?1, etc. But starting at 0 is a nuisance, and to avoid changing our whole algorithm we define a new variable T:

```
30 DIM W 255:T=W-1
```

Now we have the n'th character as "T?n", and rather than print out the letters one by one (PROCwrite above) we can simply PRINT $W. The one other time (and space) saver is the removal of the local variable in PROCswap. The only absolute need for LOCAL variables is when you need to create a 'clean' variable of that name distinct from other previous incarnations of it (as with "i" in PROCperm - it MUST be distinct from the "i" which is used inside the recursive call in line 160 above).

Finally, making PROCswap "in-line" and converting all variables to resident integers yields Program 1 and gives an increase in speed of over 3 times. But compare the legibility of the programs! It's a sad fact that on the Beeb program readability (and hence maintenance time) is diametrically opposed to speed: and in this program, where the number of permutations can be so large, speed can be very important. What I tend to do in situations like this is to keep one 'legible' program and compress it with BEEBUG's TOOLKIT PLUS (or equivalent) to produce running code. Program 1 as written is really very difficult to follow and would be a nightmare to modify when I want to use the algorithm another time.

```
  10 REM PROGRAM 1 - ANAGRAMS
  20 REM Version B1.0
  30 REM Author  Bernard Hill
  40 REM BEEBUG  Aug/Sept 1990
  50 REM Program subject to copyright
  60 :
 100 DIM W% 255:T%=W%-1
 110 INPUT $W%
 120 N%=LEN$W%
 130 PROCperm(N%)
 140 END
 150 :
1000 DEF PROCperm(L%)
1010 LOCAL I%
1020 IF L%=1 THEN PRINT $W%:ENDPROC
1030 PROCperm(L%-1)
1040 FOR I%=1 TO L%-1
1050 X%=T%?I%:T%?I%=T%?L%:T%?L%=X%
1060 PROCperm(L%-1)
1070 X%=T%?I%:T%?I%=T%?L%:T%?L%=X%
1080 NEXT
1090 ENDPROC
```

Before we leave the topic of permutation listing, I want to introduce a real problem which depends crucially on the ability to try all the permutations: the *travelling salesman problem*.

THE TRAVELLING SALESMAN PROBLEM

A salesman has five cities to visit, say, Birmingham, Carlisle, Derby, Exeter and Fishguard. He lives in Aberystwth so that every journey is a permutation of BCDEF sandwiched between two A's: e.g. AEBCDFA. The respective distances are given in the following mileage table:

	A	B	C	D	E	F
A	xxx	120	233	138	202	56
B		xxx	192	39	161	176
C			xxx	189	355	276
D				xxx	196	194
E					xxx	231

In this problem the names of the places are not relevant to the problem, let's just call them places 0 (Aberystwyth) to 5 (Fishguard). Besides the alteration from letters to numbers and initialisation of the variables, the changes to the program above are in the innermost procedure of the permutation. Instead of just writing the string, this time we need to evaluate the distance of the current permutation and compare it against the minimum. Should it be less, then we have to remember the current permutation and distance for printing at the end. Program 2 solves this problem and is easily adapted to other towns and number of towns by changing the DATA statements at the end.

This problem is very important and I have had cause to implement this very algorithm in a real situation. The head of a drilling machine moves slowly and to maximise throughput it was necessary to minimise the distance travelled in drilling a number of disparate points. Note, however, that since n! increases very rapidly (10! is about 3.5 million) this is not a feasible way to solve this problem for more than a handful of points. The problem is in fact the subject of much research and other algorithms have been evolved to give 'good' (as opposed to 'best') solutions for large numbers of points.

```
  10 REM PROGRAM 2 Travelling Salesman
  20 REM Version B 1.0
  30 REM Author  Bernard Hill
  40 REM BEEBUG  Aug/Sept 1990
  50 REM Program subject to copyright
  60 :
 100 READ N
 110 DIM dist(N,N),p(N),best(N)
 120 FOR i=0 TO N-1:FOR j=i+1 TO N
 130 READ dist(i,j):dist(j,i)=dist(i,j)
 140 NEXT:NEXT
 150 min=1E6
 160 FOR i=1 TO N:p(i)=i:NEXT
 170 PROCperm(N)
 180 PRINT "Solution  : ";
 190 FOR i=1 TO N
 200 PRINT CHR$(best(i)+65);
 210 NEXT
 220 PRINT " = ";min;" miles"
 230 END
 240 :
1000 DEF PROCperm(n)
1010 LOCAL i,t
1020 IF n=1 THEN PROCmin:ENDPROC
1030 PROCperm(n-1)
1040 FOR i=1 TO n-1
1050 t=p(i):p(i)=p(n):p(n)=t
1060 PROCperm(n-1)
1070 t=p(i):p(i)=p(n):p(n)=t
1080 NEXT
1090 ENDPROC
1100 :
1110 DEF PROCmin
1120 LOCAL i,d
1130 d=dist(0,p(1))
1140 FOR i=1 TO N-1
1150 d=d+dist(p(i),p(i+1))
1160 NEXT
1170 d=d+dist(p(N),0)
1180 IF d<min THEN min=d:FOR i=1 TO N:b
est(i)=p(i):NEXT
1190 ENDPROC
1200 :
1210 DATA 5:REM no of non-base towns
1220 DATA 120,233,138,202,56
1230 DATA 192,39,161,176
1240 DATA 189,355,276
1250 DATA 196,194
1260 DATA 231
```

Let me leave you to try your hand at some other permutation-related problems:

A. THE EIGHT QUEENS PROBLEM

How to arrange eight queens on a chessboard so that none attacks any of the others either diagonally or orthogonally.

Hints:

1. A permutation of numbers 1-8 can represent the row numbers of the queens in columns a-h where they could not attack each other orthogonally.

2. Queens on the same SW-NE diagonal have the difference of their co-ordinates the same. Queens on the NW-SE diagonal have the sum of their co-ordinates the same.

In the next article in this series we'll have a look at a more efficient program to solve this problem when we look at Backtracking Algorithms.

B. PERMUTATION SUMS

How many arrangements of the digits 1-9, and the symbols '*' and '=' are there so that they make a true product such as:

48*159=7632

Monix: A Machine Code Monitor (Part 1)

by Richard Taylor

Monix is a program which is intended to make machine code just that little bit easier while still offering something for Basic users. It is essentially a machine code monitor and has many utilities which might be found on a ROM, including a disassembler, a memory viewer, editor, mover and search routine, but you don't need Sideways RAM to use it. Adaptations can be made to suit your requirements, and machine code knowledge is not needed to do this.

Type in the program as listed keeping the same line numbers: this is important so that part two can be added correctly next month. When saved as MONIX1 and run, it will assemble a machine code file called MN. This is a fully working program: part 2 will add additional features and functions.

To use Monix, type *MN to load the assembled code. Make sure that any program in memory is saved first, as otherwise it is likely to be deleted. Press Break and 'Monix' will announce itself as if it were a ROM and then it will push up PAGE as far as it needs to. For ease of use there are several ways to access 'Monix'. You can type *CODE, *LINE, CALL old page value (usually &1900) or JSR old page value (from machine code). 'Monix' can be used inside any Basic or machine code program, but note that it does not cater for the additional opcodes of the Master's 65C02.

```
12A0  20 50 43 20 20 20 53 50   PC  SP
12A8  20 57 6B 0D 2B 41 64 64   Wk.+Add
12B0  72 65 73 73 86 26 2B 1E   ress.&+.
12B8  0B 2B 1F 00 18 0A 2B 7F   .+....+.
12C0  7F 7F 0D 41 58 59 50 86   ...AXYP.
12C8  2B 3D 20 26 2B 53 74 72   += &+Str
12D0  69 6E 67 86 2B 0D 41 6E   ing.+.An
12D8  79 20 6B 65 79 2B 0D 0D   y key+..
12E0  84 4D 69 6B 65 27 73 20   .Mike's
12E8  54 65 73 74 00 00 00 0D   Test....
12F0  0D 83 57 20 56 20 44 20   ..W V D
12F8  41 20 45 20 2A 20 4A 20   A E * J
1300  4E 20 53 20 42 20 4D 20   N S B M
1308  55 20 50 20 51 20 84 2B   U P Q .+
1310  50 72 65 73 73 20 42 72   Press Br
1318  65 61 6B 0D 2B 4D 6F 6E   eak.+Mon
1320  69 78 20 42 31 2E 38 43   ix B1.8C
1328  0D 0D 2B 69 73 20 6C 69   ..+is li
1330  6B 65 6C 79 20 74 6F 20   kely to
1338  62 65 20 64 65 6C 65 74   be delet
1340  65 64 2E 20 50 72 65 73   ed. Pres
1348  73 20 42 72 65 61 6B 20   s Break
1350  61 6E 64 0D 27 4D 6F 6E   and.'Mon
1358  69 78 27 20 77 69 6C 6C   ix' will
1360  20 61 6E 6E 6F 75 6E 63    announc
```

Memory display using the 'V' command

Six of the fourteen options are implemented in this first part, the rest being added in part two. To save space unwanted routines can be left out giving more free memory. To do this, simply type in the label as it appears in the program and then *JMP keyrts*. For example, if you donot require the memory mover then type:

```
.move JMP keyrts
```

and then leave the rest of the routine out. However, note that if you do not require the disassembler you must type:

```
.dis JMP keyrts:.dis1 RTS
```

because it is accessed in the menu screen. Also note that the string and byte searches both use a core routine starting *.bgsrch*, so this can be deleted only if both are not needed. To save more space try lowering the value of PAGE into the DFS workspace (PAGE can often be set as low as &1200), though this is quite unnecessary on a Master where PAGE is set at &E00 by default.

USING THE PROGRAM

On entering the monitor, the menu screen will come up. The top two lines show the following: Accumulator, X register, Y register, the Processor Status Register (with a bitwise display to show the flags), the Program Counter (or more correctly the return address from the monitor), the Stack Pointer, and the Current Work Address (which will be explained later).

The next four lines show the stored user zero page variables (from &70-&8F), and in part two the next eight lines will show a small disassembled area around the work address. The line starting 'Monix' is available for you to overwrite with a quick memo. Lastly there is a list of all the one letter options. This menu screen can always be recovered at any time by pressing Escape.

At any time that an address or a 2 byte value is to be input, the current work address can be quickly entered by pressing Return straight away. Otherwise, pressing Return at any stage during any input will fill the rest of the input with zeros.

PART ONE OPTIONS

W - Adjust Work Address.
Input any chosen address. From then on pressing Return when a 2 byte value is needed will enter this address automatically.

V - Memory Viewer.
Input an address and the screen will show the memory around this area. The address in hex is in the first column, followed by the next eight bytes in hex and then the same eight bytes in ASCII. If the byte is under &20 or over &7E a dot is printed (these are the non-printing characters). The following keys may be used:

A	move down through memory
Z	move up through memory
Shift	move quickly showing only the page being looked at
Escape	exit Memory Viewer

A - Adjust Registers.
Input either A, X, Y or P to adjust the Accumulator, the X register, the Y register or the Processor status register respectively. Then input the hex value required.

*** - Star Command.**
This allows any star command to be entered, making disc access, FX commands or any other useful facility accessible easily from the monitor.

N - Note.
A quick note can be easily entered from here. It will always be shown on the main menu screen.

Q - Quit Monitor.
Press Q to leave the monitor. The monitor will return the machine to the mode previously in use, and if the monitor was entered from a program it will return to the original place from which it was accessed.

PROGRAM NOTES

Lines 100-210 run the source code and *SAVE the monitor according to its size and the PAGE value. The variables are set up in lines 1000-1130. Notice my abbreviations for the OS calls, and that the beginnings of labels with these letters have the following meanings:

.bg	beginning of
.lp	loop
.en	end of
.mt	meet
.cse	and
.ccl	carry set and clear
.out	out of loop and routine

MAIN PROGRAM

.monitor saves the registers to the stack, and also the user variables (&70-&8F) and the screen mode. It then checks to see if a prompt to enter should be given.

.bg is the beginning of the program proper. It changes to mode 7, prints out the registers from the stack and the saved area of &70-&8F, and then calls the disassembler to complete the menu screen.

.gtchr detects key presses and calls the relevant routine.

.quit restores the screen mode, the user variables and the register values and then returns the machine to where it was before.

OPTION ROUTINES

.chwrkps allows the work address to be changed.

.view is the memory viewer, it involves moving *memps* (the current address) up or down by 8 bytes and then adding or subtracting an offset of &60 to find the bytes to be displayed at the top or bottom of the screen.

.adjust asks which register is to be changed and then places the new value in the correct place on the stack.

.star gets a string from *.rdline* and then sends it to the CLI (Command Line Interpreter).

.note also takes any string and places it at *.noteline* where it will be printed on the menu screen.

SUBROUTINES

These are the main subroutines vital to the running of the program. Many of them would be useful in any machine code program.

.slctwrd allows a 2 byte value (an address) to be entered. It gets a string from *.rdline* and

translates it to a number using the *.sbslct* routines. It also allows the work address to be input easily. The 2 byte value is stored through indirection with *slctvec* to the required memory address, the value in X on entering being placed in *slctvec*. To store the two byte value in &72-&73 enter *.slctwrd* with X equal to &72.

.slct is identical to the above except that it gets an input of one byte and can put the value to a non-zero page address if required.

.rdline allows an ASCII string to be typed in and stored in *.inline*. It is quite long because it allows Delete to be used.

.row prints out eight bytes in hex starting from the address contained in *memps1*.

.ascrow does the same but prints the ASCII characters of the bytes.

.prnm prints to the screen the hex value in the accumulator.

.bitwise prints the binary value of the accumulator.

.rd calls the OSRDCH to get a character from the keyboard (like G=GET). If Escape is pressed, the stack pointer is restored to its value when Monix was first called, allowing an easy exit from the nest of subroutines.

.prtxt prints out a line of text starting at .tbs plus an offset in X. It finishes on reaching a '+'.

.keyrts gives the *Any Key* message before returning to *.bg* via the technique described in *.rd*.

.prwrd prints out the address contained in *memps* and *memps+1*.

.keys detects key presses by using the negative INKEY value in X.

.back, *.beep* and *.spc* move the cursor back, give a beep and print a space respectively.

.init prints the message when *MN is typed. It also sets up the Break intercept needed to change the OSHWM which Basic takes as the value of PAGE. Finally, *.tbs* onwards is the text used by the program.

```
   10 REM  Program MONIX
   20 REM  Version B1.8C
   30 REM  Author  Richard Taylor
   40 REM  BEEBUG  Aug/Sept 1990
   50 REM  Program subject to copyright
   60 :
  100 MODE7:PRINTCHR$134"Machine Code Mo
nitor Monix"'CHR$134"by Richard Taylor"'
CHR$132"Version B1.8C"''"Code assembling
 : Please wait"
  110 DIM code &C00
  120 PROCassemble
  130 diff=code-PAGE
  140 nwpage=(((end-1) DIV256)+1)*256
  150 end=end+diff
  160 end$=STR$~(end)
  170 code$=STR$~(code)
  180 page$=STR$~(PAGE)
  190 init$=STR$~(init)
  200 PRINT'" *SAVE MN "+code$+" "+end$+
" "+init$+" "+page$
  210 OSCLI("SAVE MN "+code$+" "+end$+"
"+init$+" "+page$)
  220 nwpage$=STR$~(nwpage)
  230 PRINT'"Monitor assembled and saved
 to run at"'~page$". The new PAGE settin
g will be "~nwpage$"."'"Type *MN to load
  and then *CODE, *LINE,CALL &"page$", o
r JSR &"~page$" (from machine"'"code) to
 access."
  240 PRINT'"This assembler program need
 not be run  again unless used with a di
fferent      original PAGE value or IF c
hanges are made to the program."
  250 PRINT'CHR$136"HAVE YOU SAVED THIS
PROGRAM!"
  260 END
  270 :
 1000 DEFPROCassemble
 1010 FORA%=4TO6STEP2
 1020 owr=&FFEE:onw=&FFE7
 1030 oby=&FFF4:ord=&FFE0
 1040 oas=&FFE3:ocl=&FFF7
 1050 var=&70:var1=&71:var2=&72:cnt=&73
 1060 sp=&74:amnt=&75
 1070 lgth=&77:wrkps=&79
 1080 mask=&7B:lim=&7C
 1090 memps=&80:memps1=&82:retadr=&84
 1100 jsra=&86:jsrx=&87:jsry=&88
 1110 opcode=&89:byte1=&8A:byte2=&8B
 1120 slctvec=&8C:retadr=&8E
 1130 P%=PAGE:O%=code
 1140 [
 1150 OPTA%
 1160 .monitor PHP:PHA:TXA:PHA:TYA:PHA
 1170 CLD:CLI:LDX#&1F:.lp
 1180 LDA&70,X:STAretmem,X:DEX:BPLlp
 1190 LDAprmvar:BEQstart
 1200 :
```

```
1210 LDX#0:JSRprtxt
1220 JSRrd:CMP#ASC"Y":BEQntquit1
1230 JMPquit1:.ntquit1
1240 :
1250 .start LDA#135:JSRoby:TYA:PHA
1260 .reset LDA#22:JSRowr:LDA#7:JSRowr
1270 TSX:STXsp
1280 LDA&106,X:CLC:ADC#1
1290 STAretadr:STAwrkps
1300 LDY&107,X:BCCokreset:INY
1310 .okreset STYretadr+1:STYwrkps+1
1320 :
1330 .bg LDA#12:LDX#2:JSRoby:JSRowr
1340 LDX#(t0-tbs):JSRprtxt
1350 TSX:LDA&104,X:JSRprnm
1360 TSX:LDA&103,X:JSRprnm
1370 TSX:LDA&102,X:JSRprnm
1380 TSX:LDA&105,X:JSRprnm:JSRbitwise
1390 LDAretadr+1:STAmemps+1
1400 LDAretadr:STAmemps:JSRprwrd
1410 TSX:TXA:CLC:ADC#7:JSRprnm
1420 LDAwrkps:STAmemps
1430 LDAwrkps+1:STAmemps+1:JSRprwrd
1440 LDA#retmemDIV256:STAmemps1+1
1450 LDA#retmem MOD256:STAmemps1:JSRonw
1460 LDA#&70:STAvar:.lp1 JSRonw
1470 LDAvar:JSRprnm:JSRow:JSRascrow
1480 LDAmemps1:CLC:ADC#8:STAmemps1
1490 BCCccl:INCmemps1+1:.ccl
1500 LDAvar:CLC:ADC#8:STAvar
1510 CMP#&90:BElp1
1520 LDAmemps:SEC:SBC#6:STAmemps
1530 BCScse:DECmemps+1:.cse
1540 LDA#9:STAlim:JSRdis1
1550 LDX#(noteline-tbs):JSRprtxt
1560 .gtchr JSRrd:JSRowr:TAY:JSRonw:TYA
1570 CMP#ASC"V":BNEntview
1580 JSRview:.ntview
1590 CMP#ASC"D":BNEntdis
1600 JSRdis:.ntdis
1610 CMP#ASC"E":BNEntedit
1620 JSRedit:.ntedit
1630 CMP#ASC"U":BNEnteditusr
1640 JSReditusr:.nteditusr
1650 CMP#ASC"A":BNEntadjust
1660 JSRadjust:.ntadjust
1670 CMP#ASC"B":BNEntbysrch
1680 JSRbysrch:.ntbysrch
1690 CMP#ASC"S":BNEntstsrch
1700 JSRstsrch:.ntstsrch
1710 CMP#ASC"*":BNEntstar
1720 JSRstar:.ntstar
1730 CMP#ASC"M":BNEntmove
1740 JSRmove:.ntmove
1750 CMP#ASC"P":BNEntprompt
1760 JSRprompt:.ntprompt
1770 CMP#ASC"N":BNEntnote
1780 JSRnote:.ntnote
1790 CMP#ASC"J":BNEntjsr
```

```
1800 JSRjsr:.ntjsr
1810 CMP#ASC"W":BEQchwrkps
1820 CMP#ASC"Q":BEQquit
1830 JMPbg
1840 .quit LDA#22:JSRowr
1850 PLA:JSRowr
1860 .quit1
1870 LDX#&1F:.lp3
1880 LDAretmem,X:STA&70,X
1890 DEX:BPLlp3
1900 PLA:TAY:PLA:TAX:PLA:PLP:RTS
1910 :
1920 :
1930 :
1940 .chwrkps LDX#(t1-tbs):JSRprtxt
1950 LDA#wrkps:JSRslctwrd:JMPbg
1960 :
1970 :
1980 .view LDX#(t1-tbs):JSRprtxt
1990 LDA#memps:JSRslctwrd
2000 LDAmemps:AND#&F8
2010 CLC:ADC#200:STAmemps
2020 BCCviewccl:INCmemps+1:.viewccl
2030 LDA#0:STAcnt:LDA#12:JSRowr
2040 .bgview LDAcnt:CMP#25:BCSbgview1
2050 INCcnt:BNEdownview
2060 .bgview1:LDX#&9E:JSRkeys:BCSupview
2070 LDX#&BE:JSRkeys:BCSdownview
2080 LDX#&8F:JSRkeys:BCCbgview:JMPrts
2090 .downview LDX#(t2-tbs):JSRprtxt
2100 LDAmemps:SEC:SBC#8:STAmemps
2110 BCScseview:DECmemps+1:.cseview
2120 LDYmemps+1:SEC:SBC#&60
2130 BCScseview1:DEY:.cseview1 JMPgo
2140 .upview LDX#(t3-tbs):JSRprtxt
2150 LDAmemps:CLC:ADC#8:STAmemps
2160 BCCcclview:INCmemps+1:.cclview
2170 LDYmemps+1:CLC:ADC#&60
2180 BCCcclview1:INY:.cclview1
2190 .go STAmemps1:TYA:STAmemps1+1
2200 JSRprnm:LDX#&FF:JSRkeys:BCCntspd
2210 JMPbgview:.ntspd
2220 JSRback:LDAmemps1:JSRprnm:JSRrow
2230 JSRascrow:JMPbgview
2240 :
2250 :
2260 .adjust LDX#(t4-tbs):JSRprtxt
2270 .gtchar1 JSRrd:LDXsp
2280 CMP#ASC"Y":BEQmtadj:INX
2290 CMP#ASC"X":BEQmtadj:INX
2300 CMP#ASC"A":BEQmtadj:INX
2310 CMP#ASC"P":BNEgtchar1
2320 .mtadj JSRowr:JSRspc
2330 STXmemps:LDX#(t5-tbs):JSRprtxt
2340 LDA#var2:JSRslct
2350 LDXmemps:STA&102,X:JMPrts
2360 :
2370 :
2380 .star JSRowr
```

```
2390 LDA#18:STAlim:JSRrdline
2400 LDA#&0D:STAinline,Y:JSRonw
2410 LDX#inline MOD256
2420 LDY#inline DIV256
2430 JSRocl:JMPkeyrts
2440 :
2450 :
2460 .note LDX#(t6-tbs):JSRprtxt
2470 LDA#14:STAlim:JSRrdline
2480 LDY#13:.lpnote
2490 LDAinline,Y:STAnoteline+3,Y
2500 DEY:BPLlpnote:JMPrts
2510 :
2520 :
3000 .edit
3010 .editusr
3020 .bysrch
3030 .stsrch
3040 .move
3050 .prompt
3060 .jsr
3070 .dis
3080 JMPkeyrts
3090 .dis1
3100 RTS
3110 :
3120 :
3130 :
6000 .slctwrd LDX#0
6010 STXslctvec+1:STAslctvec
6020 LDA#4:STAlim:JSRrdline:JSRalign
6030 LDY#0:LDAinline,Y:BNEmtslctwrd
6040 INY:LDAwrkps+1:STA(slctvec),Y
6050 JSRprnm:JSRback:DEY
6060 LDAwrkps:STA(slctvec),Y
6070 JMPprnm
6080 .mtslctwrd JSRsbslct0
6090 INY:STA(slctvec),Y
6100 JSRback:INY:JSRsbslct0
6110 LDY#0:STA(slctvec),Y:RTS
6120 :
6130 .slct LDX#0
6140 .slctntzp STXslctvec+1:STAslctvec
6150 LDA#2:STAlim:JSRrdline:JSRalign
6160 LDY#0:JSRsbslct0
6170 STA(slctvec),Y:RTS
6180 :
6190 .sbslct0 LDAinline,Y:JSRsbslct1
6200 ASLA:ASLA:ASLA:ASLA:STAvar
6210 LDAinline+1,Y:JSRsbslct1
6220 ORAvar:JSRprnm:RTS
6230 .unrec LDA#48
6240 .sbslct1
6250 CMP#48:BCCunrec:CMP#58:BCCnonlet
6260 CMP#65:BCCunrec:CMP#71:BCSunrec
6270 SEC:SBC#7:.nonlet SEC:SBC#48:RTS
6280 .align LDXlgth:BEQenalign
6290 .lpalign JSRback
6300 DEX:BNElpalign:.enalign:RTS
6310 :
6320 :
6330 .rdline LDY#0:TYA:INClim:LDXlim
6340 .wipe STAinline,X:DEX:BPLwipe
6350 .lprdline STYvar1:JSRrd2
6360 CMP#127:BNEntdelt
6370 .delt DEY:BMInochar
6380 JSRowr:LDA#0:STAinline,Y
6390 JMPlprdline
6400 .nochar INY:JMPlprdline
6410 .ntdelt CMP#&0D:BEQenrdline
6420 JSRowr:LDYvar1:STAinline,Y
6430 INY:CPYlim:BNElprdline
6440 JSRbeep:LDA#127:JMPdelt
6450 .enrdline STYlgth:RTS
6460 :
6470 :
6480 .row JSRspc:LDY#0:.lprow
6490 LDA(memps1),Y:JSRprnm
6500 INY:CPY#8:BNElprow:RTS
6510 .ascrow JSRspc:LDY#0:.lpascrow
6520 LDA(memps1),Y:JSRasc
6530 INY:CPY#8:BNElpascrow:RTS
6540 .asc CMP#32:BCSokasc
6550 LDA#ASC".":.okasc
6560 CMP#127:BCCokasc1
6570 LDA#ASC".":.okasc1 JSRowr:RTS
6580 :
6590 :
6600 .prnm TAX:LSRA:LSRA:LSRA:LSRA
6610 JSRprnm1:TXA:AND#&0F:JSRprnm1
6620 JSRspc:TXA:RTS
6630 .prnm1 CMP#10:BCSlttrs
6640 ADC#48:JMPmtprnm1:.lttrs
6650 CLC:ADC#55:.mtprnm1 JMPowr
6660 :
6670 :
6680 .bitwise TAX
6690 LDA#&80:STAmask:LDY#8:.lpbit
6700 TXA:ANDmask:BEQntone
6710 LDA#49:BNEmtbit
6720 .ntone LDA#48:.mtbit JSRowr
6730 LSRmask:DEY:BNElpbit
6740 JSR spc:TXA:RTS
6750 :
6760 :
6770 .rd LDA#15:LDX#1:JSRoby
6780 .rd2 JSRord:CMP#&1B:BEQesc:RTS
6790 .esc LDA#&7E:JSRoby
6800 LDXsp:TXS:JMPbg
6810 :
6820 :
6830 .prtxt LDAtbs,X:CMP#43
6840 BEQenprtxt:JSRoas
6850 INX:BNEprtxt:.enprtxt RTS
6860 :
6870 .keyrts LDX#(t7-tbs):JSRprtxt
6880 JSRrd:.rts JMPesc
```

Continued on page 38

Games Review

by Peter Rochford

The Archimedes with its amazing sound and graphics capability steals most of the limelight these days, yet there are still quality games being produced for the good old Beeb. In fact, software houses are now looking to capitalise on successful Arc releases by bringing out versions of them for the Beeb where possible.

E-Type

E-TYPE

Such is the case with one of the latest releases from the 4th Dimension, *E-Type* (£14.20). This originally appeared for the Archimedes with great success, and justifiably so. E-Type is a road racing game where you take control of a Jaguar E-Type sports car to tear off across roads and deserts at speeds of over 150mph.

There are several different tracks to choose from and varying levels of skill. Control can be either by joystick, or the popular 'Snapper' key combination in conjunction with Return and Space bar.

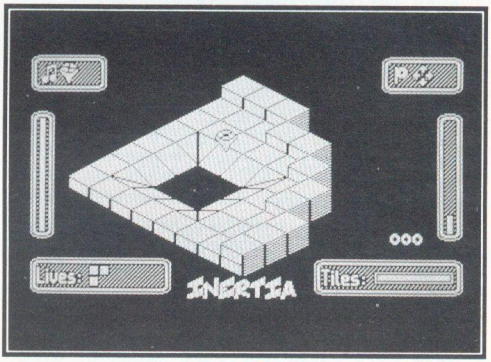
Once you have moved off, you accelerate away, stepping through the gears as you go. Along the roads are various obstacles to avoid such as boulders and trees. Some of these just slow you down a bit when hit, but others bring you to an abrupt halt, with both occupants of your car being unceremoniously catapulted out of their seats. Having set the skill level beforehand, you are allowed to crash several times before your car is a final write-off.

The object of the game is to complete the track in a certain qualifying time in order to progress on to the next. To this end, time bonuses are awarded for hitting certain objects in the road, which include suicidal policemen who stand in your way. Great fun!

Naturally, you are not alone on the highway. There are scores of other cars hurtling around like maniacs and these do their best to obstruct you or force you off the road.

I have regularly played the Arc version of E-Type and I suppose that does tend to spoil you a bit. However, this version for the Beeb is every bit as good in terms of fun, and shares most of the features of the Arc game. Graphics are well-detailed and the 3D scrolling landscape is flicker free and colourful. Control of the car is straightforward and this is a game that you can quickly take to.

E-Type is one of the best releases now around for the Beeb and should appeal to many. Definitely recommended.



Inertia

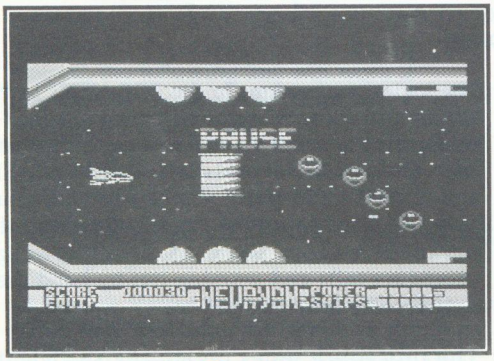
INERTIA

Another new release from the 4th Dimension is *Inertia* (£12.95). This is a rather unusual game that takes place on a 3D tiled landscape. The object is to guide your free-floating craft around this landscape, negotiating the bridges and ramps, to collect the special tiles that are scattered around. Some tiles you pass over have strange properties,

such as ice, jump and transformation. The worst hazards to avoid are the holes that seem to suck you down, and the tendency to drift too easily off the edge of the landscape.

Inertia is a massive game with screens that seem to go on just about forever. Control of the craft is tricky at first, but you quickly find that you can get to grips with it. Once you have, beware. This game is time-consuming and very addictive!

Graphics in Inertia are not what you might call outstanding, but that doesn't really matter. For all their simplicity, it is the game itself that makes up for it. As I said at the start, this is an unusual and novel game that makes a change from shoot-'em-ups and "ladders and levels". Highly recommended.



Nevryon

NEVRYON

Yet another 4th Dimension release is *Nevryon* (£14.20). I can't pronounce it either! This is a sideways scrolling shoot-'em-up arcade-type game where you must thrash everything in sight. There are eight levels and the game consists of 100K of graphics and code.

The object of the game, so the story goes, is to free the planet of Nevryon from the aliens that have landed to steal the titanium ore in the planet's caves. Down in the caves is where most of the action takes place. You fly your craft by either keyboard or joystick control, and shoot everything in your path.

The action is fast and furious with plenty of detailed colourful sprites and lots of raucous noisy sound. You can choose either a large, low-res screen or a smaller, hi-res screen by pressing a function key. I must say that I am not impressed by the scrolling of this game; it is very flickery. I appreciate that there is a considerable amount of object detail including background stars, but I have certainly seen better sideways scrolling on a Beeb.

Generally I was disappointed with Nevryon. I found that the flickering of the screen and the sometimes messy graphics spoilt my enjoyment. I personally would not buy it. Some old favourites of this genre, such as Planetoid and Rocket Raid, provide a far more satisfying game of death and destruction to my mind.

HOSTAGES

Hostages (£16.96), from Superior Software, is an arcade-type action adventure that has already appeared on machines such as the C64, ST and Amiga. There is also a version available for the Arc, and now too for the Beeb.

The object of the game is to make an assault on an Embassy that has been taken over by terrorists, in order to release the hostages they are holding captive.

You first must place your marksmen around the building and then drop your paratroops onto the roof, where they descend by rope into the building to work their way through the rooms, avoiding being killed by the terrorists. This is a rather simple statement of the game. It is, in fact, rather more involved than this, being a mixture of strategy and shoot-'em-up.

This is not a game for an occasional five minute bash. You really have to sit down and plan what you are doing to achieve any degree of success. I found it intriguing and fun to play. You can easily get quite sucked into the scenario and become really involved.

The graphics for an eight bit micro like the Beeb are very good indeed and full marks to Peter Scott for another fine conversion job. The sound is equally good too.

Hostages is a game that I find very appealing and I have already become quite addicted to it. This is one of Superior's best releases to date and definitely recommended.

All the games reviewed here are available from BEEBUG. All prices quoted include VAT.

1st course

Scrolling Text Routines

by S.Sexton

In the First Course articles for Vol.7 Nos.4 & 5 we presented a number of routines by Lindsey Cullen for putting scrolling text on to your screens. This month S.Sexton contributes some more ideas on the same subject.

A routine for putting a scrolling text message onto the screen can often provide a useful 'front end' for many programs. In this article we shall look at a number of variants of a procedure, PROCScroll, which allows controlled scrolling of a text message in any mode. The width of the scroll, the scroll starting point on the screen, and the speed and direction of scrolling can all be defined by the user. These parameters are passed to the procedure with the following syntax:

```
PROCScroll(mssg$,X,Y,wdth,dir,spd)
```

where:

mssg$ is the string to be scrolled,

X and *Y* define the starting position for the left hand end of the scroll banner,

wdth is the width of scroll banner,

dir is the direction of scroll. TRUE will give a right to left scroll, and FALSE the opposite,

spd is the length of a FOR-NEXT loop, slowing down the scroll. A value of 250 is a good starting point.

Note that this method of creating a pause in a program is dependent upon the speed of the machine on which it is used. For a more constant approach it is better to use the pseudo variable TIME in the form:

```
t=TIME:REPEAT:UNTIL TIME-t>wait
```

where *wait* is the number of centi-seconds delay required. However, the FOR-NEXT loop is commonly used, and does give finer control than is possible with TIME. On a Master 128, the value of 250 suggested above is reduced to just 5 if the alternative method with TIME is used.

Here is the first and most basic form of our scroll procedure:

```
1000 DEF PROCscrollA(mssg$,X%,Y%,wdth%,dir%,wait%)
1010 len%=LEN(mssg$)+wdth%+1
1020 FORstep%=len%*-(dir%=FALSE)TO len%*-(dir%=TRUE)
     STEP -(dir%*2+1)
1030 PRINTTAB(X%,Y%);MID$(STRING$(wdth%," ")+mssg$+
     " ",step%,wdth%)
1040 FOR T%=1 TO wait%:NEXT
1050 NEXT step%
1060 ENDPROC
```

The logical expressions in the FOR-NEXT loop at line 1020 allow the same statement to be self-configuring to count up or down depending on the values of the expressions (dir%=FALSE) and (dir%=TRUE) which will be 0 or -1. If you find this hard to follow try substituting the parameters from some of the following procedure calls and working out by hand the format of the resulting FOR-NEXT expression.

The procedure relies on adding sufficient spaces at each end of the string to cover up the results of previous PRINT statements.

In use, for example:

```
PROCscrollA(A$,0,10,39,TRUE,250)
```

will scroll the text in A$ from right to left, while:

```
PROCscrollA(A$,0,10,39,FALSE,250)
```

will scroll the text from left to right. Both of these examples specify a width of 39, so with a 40 column mode 7 screen, the text scrolls the full width of the screen. On the other hand:

```
PROCscrollA(A$,15,10,10,TRUE,250)
```

will just scroll the text within a 10 character window, and this can again be selected to scroll in either direction. You will probably find it

easier to appreciate what is going on if you type in the procedures and try out the examples as we progress.

However, this is only the start of the fun that is possible with this routine. The text can have *resistance* to being scrolled (imagine somebody giving the text "jerks" opposing the scroll). The effect (in a random form) can be added with the following line:

```
1045 IF RND(10)<4 THEN step%=step%-2
```

Or, if controlled "jerks" are wanted, (e.g. to re-read a section of text if missed the first time), the following can be used:

```
1045 IF INKEY-99 THEN step%=step%-2
```

which assumes that the space bar is pressed to re-read text. Or, if scrolls are wanted to "converge", one being *reversed*, the following line will work if the value of "scroll width" is 19, and a 40 column mode is used:

```
1045 PRINTTAB(X%+19,Y%);MID$(STRING$(wdt
h%," ")+mssg$+" ",LEN(mssg$)-step%,wdth%)
```

However, a better solution to split string scrolling is given in PROCscrollB.

```
1100 DEF PROCscrollB(mssg$,X%,Y%,wdth%,dir%,wait%)
1110 len%=LEN(mssg$)+wdth%+1
1120 FORstep%=len%*-(dir%=FALSE)TO len%*-(dir%=TRUE)
     STEP -(dir%*2+1)
1130 PRINTTAB(X%,Y%);MID$(STRING$(wdth%," ")+mssg$+
     " ",step%,wdth%)
1140 PRINTTAB(X%+19,Y%);MID$(STRING$(wdth%," ")+mssg$+
     " ",LEN(mssg$)-step%+wdth%+1,wdth%)
1150 FOR T%=1 TO wait%:NEXT
1160 NEXT step%
1170 ENDPROC
```

The same text string is scrolled from both left and right margins at the same time, or depending on the direction of scroll, from the centre of the screen outwards towards the margins. Using the call:

```
PROCscrollB(A$,0,10,19,TRUE,250)
```

will scroll the text from the sides to the centre, while:

```
PROCscrollB(A$,0,10,19,FALSE,250)
```

will work from the centre to the sides. Care is needed to ensure that the two sub-strings reach the middle, or the sides together. Some trial and error may be needed here to get the most satisfying effect.

The possibilities *really* get interesting when the procedure is adapted to handle TWO messages, one at the top of the screen and one at the bottom. It gets even more fun if one of these is running backwards!

```
1200 DEF PROCscrollC(mssg$,X%,Y%,wdth%,dir%,wait%)
1210 len%=LEN(mssg$)+wdth%+1
1220 FORstep%=len%*-(dir%=FALSE)TO len%*-(dir%=TRUE)
     STEP -(dir%*2+1)
1230 FOR a=1 TO 4
1240 PRINTTAB(X%,Y%+a*2);MID$(STRING$(wdth%," ")+
     mssg$+" ",step%,wdth%)
1250 PRINTTAB(X%,Y%+a*2+1);MID$(STRING$(wdth%," ")+
     mssg$+" ",LEN(mssg$)-step%+wdth%+1,wdth%)
1260 NEXT
1270 FOR T%=1 TO wait%:NEXT
1280 NEXT step%
1290 ENDPROC
```

This gives a ripple tank effect, with the scroll multiplied down the screen. For example:

```
PROCscrollC(A$,0,6,39,TRUE,0)
```

or the same thing backwards with:

```
PROCscrollC(A$,0,6,39,FALSE,0)
```

Another option (in mode 7) is double height text, but a problem occurs here when text is printed *behind* the scroll window. A "turn-off" VDU code would be needed. But if this is not important, the PRINT statement could be doubled.

```
1300 DEF PROCscrollD(mssg$,X%,Y%,wdth%,dir%,wait%)
1310 len%=LEN(mssg$)+wdth%+1
1320 FORstep%=len%*-(dir%=FALSE)TO len%*-(dir%=TRUE)
     STEP -(dir%*2+1)
1330 PRINTTAB(X%-1,Y%);CHR$141;MID$(STRING$(wdth%," ")+
     mssg$+" ",step%,wdth%)
1340 PRINTTAB(X%-1,Y%+1);CHR$141;MID$(STRING$(wdth%,"
")+
     mssg$+" ",step%,wdth%)
1350 FOR T%=1 TO wait%:NEXT,:ENDPROC
```

For example, this call will give an effect similar to the first routine but now in double height:

```
PROCscrollD(A$,1,10,39,TRUE,250)
```

and this one will scroll the text the other way:

```
PROCscrollD(A$,1,10,39,FALSE,250)
```

Combinations of "reversed scrolls", "multiple scrolls" and "block scrolls" (same message repeatedly scrolled down the screen) tend to slow the program down because of the sheer amount of work that needs to be done, but the value of *spd* can be lowered to suit.

Lastly, here is version of the basic procedure which combines double messages and reverses all at once. To cope with two messages, all the parameters except for direction and speed which apply to both messages are doubled.

```
1400 DEF PROCscrollE(mssg1$,mssg2$,X1%,Y1%,X2%,
     Y2%,wdth1%,dir%,wdth2%,wait%)
1410 FOR step%=0 TO LEN(mssg1$)+wdth2%+1
1420 PRINTTAB(X1%,Y1%);MID$(STRING$(wdth1%," ")+
     mssg1$+" ",step%,wdth1%)
1430 PRINTTAB(X2%,Y2%);MID$(STRING$(wdth2%," ")+
     mssg2$+" ",LEN(mssg2$)-step%+wdth2%+1,wdth2%)
1440 FOR T%=1 TO wait%:NEXT
1450 NEXT step%
1460 ENDPROC
```

There is one drawback to the system. Because of the MID$ process in the procedure, there is a limit to the maximum length of string that can be handled. This varies with the value of "scroll width" and can be found, for a given scroll width by:

```
max. length = 255 - (scroll width + 1)
```

An example of the use of this procedure for two strings A$ and B$ might be:

```
PROCscrollE(A$,B$,10,5,0,20,20,TRUE,39,200)
```

All these ideas are incorporated in a comprehensive working demo on this month's magazine disc. B

ADFS Directory Examiner and Command File Creator (continued from page 17)

```
 3410 :
 3420 DEF PROCdisc
 3430 REPEAT
 3440 PROCprompt("  Insert source disc a
nd press any key"):G%=GET
 3450 PROCprompt("  (Paged Mode-Press Sh
ift to scroll)       Press Break to leav
e the program")
 3460 CLS:OSCLI("DIR"):OSCLI"."
 3470 back%=0:*FX21,0
 3480 PROCprompt(" Press Return to accep
t current disc,    or change source disc
 and press a key.")
 3490 G%=GET
 3500 UNTIL G%=13
 3510 ENDPROC
 3520 :
 3530 DEF PROCinit
 3540 VDU23,245,145,210,180,144,6,41,79,
137
 3550 VDU23,246,32,80,70,105,75,77,73,134
 3560 VDU23,247,32,80,66,102,66,66,66,135
 3570 VDU23,248,32,80,70,105,66,68,72,143
 3580 VDU23,249,32,80,70,105,66,65,73,134
 3590 VDU23,250,32,80,66,100,76,79,66,130
 3600 VDU23,251,32,80,71,100,98,65,69,130
 3610 VDU23,252,32,80,66,100,72,78,73,134
 3620 VDU23,253,32,80,71,97,65,66,66,130
 3630 VDU23,254,32,80,70,105,70,73,73,134
 3640 VDU23,255,32,80,70,105,71,65,66,132
 3650 DIM dir$(46),level$(127),control%(6)
 3660 DIM control% 12,output% 16,block% 4
 3670 DIM buffer% 10,string% 11,cntrl% 18
 3680 *FX21,0
 3690 *FX4,1
 3700 level%=0:level$(0)="$"
 3710 back%=0:cfp%=0:key$=""
 3720 OSCLI"KEY7 "+CHR$(253)
 3730 OSCLI"KEY8 "+CHR$(254)
 3740 OSCLI"KEY9 "+CHR$(255)
 3750 ENDPROC
 3760 :
 3770 DEF PROCend
 3780 CLOSE#0:OSCLI"DIR":VDU26,12,15,20
 3790 *FX18
 3800 *FX4,0
 3810 PRINT CHR$(12);"Finished:"'"Key f0
 to *EXEC Command File"'"Key f1 to *EDIT
 Command File"
 3820 *KEY0 *EXEC !Command_F|M
 3830 *KEY1 *EDIT !Command_F|M
 3840 ENDPROC
```

B

512 Forum

by Robin Burton

This month I'll carry on with the topics that were to follow the article of two months ago. The delay has proved beneficial, because since then I've had a letter from a Forum reader which included a very useful tip.

CD AGAIN

Towards the end of the June Forum on the subject of directory and path names, I had mentioned the change directory command, 'CHDIR', or 'CD' for short. That there's more to say about this subject will be obvious to most of you, but there's a good reason why I didn't go further at the time. At this point perhaps I should explain how the content of each issue of the Forum evolves.

Usually (though I must admit, not always) as I sit down to write each Forum I have a pretty clear idea of the topics for the month. At the same time there is rarely a detailed plan, all I have in mind are the main points I want to raise. The truth is, therefore, that more often than not I "make it up as I go along". The result of this technique (and I use that word in its loosest possible sense) is that I begin with the thought that I'm going to run out of ideas before I reach the end of the article, but the reverse is more often the case and I run out of space before I've covered everything I want to include, like this month, hence the 'extra bit' (thanks Mike).

So it was two months ago. I'd like to have developed the use of 'CD' further and intended to do so, but last month's digression delayed this. In the time since then I've learned of another use of 'CD', one which I freely admit I'd never tried, so all credit to the author. Before I give you this tip I'd better cover the points I had intended to go on to, otherwise it might not be understood by some of our newer 512 converts.

PATH

Some of you will be familiar with the floating drives, N:, O: and P:, though these are much more likely to be used in hard disc systems where the number and structure of sub-directories is unavoidably more complex than on floppy disc. I've mentioned floating drives before in the Forum, but even so I know that some of you aren't sure how they work, particularly as DOS reference books won't mention them because they're actually a CP/M facility.

The 'floating' drives are not physical disc drives, but are notional drive identifiers which can be allocated by the user to real drives or directories. For example you might have a long-winded directory path to which you frequently need to refer. The format of the allocation, using drive N: as our example is:

```
CD N:=DIRECTORY PATHNAME
```

where the directory pathname can be any you wish to access.

By using the floating drives you can therefore allocate a single 'shorthand' name to any directory, which the system will recognise and translate accordingly when the short name is given. Practical problems will perhaps show the benefits of this facility better.

Suppose that you have several long directory names that you use frequently and therefore need to type fairly often. Sometimes such names can be included in a 'PATH' command, but the major problem of 'PATH' (as I explained in Forum Vol.8 No.3) is that all the separate paths you want to specify must be given together in a single command.

This is because every new 'PATH' setting always entirely replaces any paths set up by the previous 'PATH' command. The obvious implication is that the total length of all the pathnames you wish to specify must all fit within the length of a single DOS command-line command, 127 characters. Given that each

directory name can be up to eight characters and the directory names in any path must be separated by a '\' character and are usually preceded by a drive identifier, you can see that it's quite easy to run out of space if you're setting multiple paths.

Are you limited therefore to very short directory names, or constrained to set only a very few paths?

Well, not necessarily, but to overcome these space limitations you might try attacking the problem from a slightly different angle and this is where 'CD' and floating drives become extremely useful (you see, I got there in the end).

CD PLUS PATH

In my 512 I have a number of pathnames and floating drive settings in my 'AUTOEXEC.BAT' file. Some of them, of course, are short and present no problems. For example, I permanently run a RAMdisc to which I copy all the most frequently needed standard DOS utilities such as COMMAND, FSET, SDIR, CHKDSK, DISK, SHOW, FORMAT, PRINT and so on. All these need loading from disc every time they are called, but from the RAMdisc they operate so fast that they operate just like permanent DOS commands. Also, because access is so rapid, Drive D: is always the first entry in my PATH setup. If any file isn't in drive D: the delay of checking it is so short it's undetectable.

I also run some files from the root of drive C:, mostly with the system attribute set so that they won't clutter the root directory display. These are also general utilities, the ones which aren't used very frequently, but which, even so, might be needed at any time. Therefore C:\ is the next path setting.

The next path I set is the directory holding my word processor applications software, a directory called 'PCWRITE' which is in my main applications directory, 'SOFTWARE', also in drive C:. The full pathname specification is therefore:

```
C:\SOFTWARE\PCWRITE\
```

I also use PCTOOLS from Central Point Software as enough of it works correctly in the 512 to justify the megabyte or so it occupies on disc. This too is in 'SOFTWARE', and its path is called:

```
C:\SOFTWARE\PCTOOLS\
```

The next directory I include contains my own machine code programs, and its full path is:

```
C:\ES\PROGRAMS\
```

I also regularly use an 80186 assembler/debugger which also has its own directory, called:

```
C:\SOFTWARE\A86\
```

I have several other regularly used paths, so by now you must be getting the idea. If I tried to specify all these pathnames together in one PATH command I'd probably run out of room. The important point here is that 'PATH' won't tell you there's a problem, you find out later when something doesn't work.

If the convenience of setting paths can't be done by the direct approach, the answer is to use floating drives with the 'CD' command.

The most sensible way is to substitute the floating drive names for the longest pathnames that you manually refer to most frequently, thus saving the most space and effort. The pathnames for the word processor and PCTOOLS are the longest, but I rarely need to explicitly 'CD' to them. There are others though that I often use, but which, without a shortcut need a lot of typing. One example is my source directory, the path of which is 'C:\ES\SOURCES\'. However, by using the command:

```
CD N:=C:\ES\SOURCES\
```

in my batch file, when I want to specify this path I can simply call it N:. Sometimes this facility can also be used in 'PATH', so as an example, for the PCWRITE pathname, using drive O: I can likewise shorten that path to O:, and P: for PCTOOLS. Now, using my first four pathnames for illustration, in my AUTOEXEC.BAT file, instead of:

```
PATH D:;C:;C:\SOFTWARE\PCWRITE\
      :C:\SOFTWARE\PCTOOLS\
```

I can simply use:

```
PATH D:;C:;O:;P:
```

which as you can see is considerably shorter, and leaves plenty of room for more paths.

The main advantage of floating drives though, is using them when you will need to regularly change directories manually. Although there are only three floating drives and I have used two of them in the batch file, I always reserve one, N:, to point to my working directory, whatever that is at the time. Let me explain.

In the 512 I perform one of two main jobs. One is writing text for various purposes, the other, and the most frequent is writing machine code programs. Taking the latter as the example, all my source code is in a directory called '\ES\SOURCES\' in drive C:. By allocating drive N: to this path when I'm doing this job, I can simply type 'P:' and I have virtually automatically performed:

```
CD \ES\SOURCES
```

To edit a source program I just type:

```
ED PROGRAM.SRC
```

ED being the main word processor module, and PROGRAM.SRC being the name of the program I'm working on at the time. After editing I can then exit to a DOS shell, type:

```
ASSEMBLE PROGRAM.SRC
```

and the COM file is assembled into \ES\PROGRAMS\ by my batch file, ASSEMBLE, with an automatic change of directory to the program directory at the end so that I can immediately test the program. As frequently happens I then need to go back to the source directory to correct or change something I've found during de-bugging, so all I need enter is:

```
N:
```

and I'm back in the source directory. This is followed by:

```
EXIT
```

to leave the DOS shell and return to my word processor, which still has the source code loaded ready for instant edit.

Not everyone will have such a repetitive operation as this of course, but if you do it's well worth the effort of setting it up for maximum convenience like this. With this set-up I can temporarily exit my word processor to a DOS shell, assemble a program, check a specific point in the debugger and return to the source code in the word processor in under half a minute, with the absolute minimum of typing and hence little chance of error.

PROBLEM AREAS

Unfortunately it's not a perfect world and there are a couple of potential problems lurking in these techniques which may become obvious only when you've tried them. The first concerns the way you specify pathnames, and applies whether you use floating drives or not.

You'll have noticed that I specified all the pathnames in full, starting with the drive and the root directory. The reason is that each path will be searched in isolation by DOS, so you can't for example specify '\SOFTWARE\' as one path and have, for example '\PCWRITE\' and '\PCTOOLS\' as alternatives or 'sub-paths' within the first. 'PATH' just isn't that clever, so unless you're positive that a pathname will always be within your current directory in the current drive when it's needed you must supply full path specifications.

The second possible problem concerns the use of floating drives only, but it is even less obvious and potentially harder to diagnose. Whether it's a problem or not depends on the software you're using at the time. Some applications will work, others might not so some investment in trial and error may be needed before you can decide when to use floating drives.

If an application uses separate definition files for its configuration options, as many commercial packages do, depending on how it reads path settings it might or might not be able to find these files if you're not in its own directory at the time.

The problem is that the application will load wherever it's called from, because DOS Plus controls that process and of course DOS Plus correctly translates floating drive specifications. However, the reading of configuration or definition files will be done by the program itself, after the main module has begun execution. Since floating drives are a CP/M facility (i.e. not DOS) there's a good chance that

many applications will complain they can't find a file, or may alternatively resort to built-in defaults at this point, because they can't translate the floating drive path correctly.

ASSIGNING DRIVES

You can see that with floating drives and the less publicised features of 'CD', there's a lot of scope to customise your 512 for specific operations. Even so there's one variation I've never used.

Colin Price wrote to me from Reading as a result of the June Forum (Vol.9 No.2) to point out another very useful facility of 'CD'. I'd said in that issue that there's no 'ASSIGN' command in DOS Plus, but there are other ways round the problem. Colin wrote to remind me of one of them. CD is capable of providing such a facility.

The general format of the command is:

```
CD target-drive=substitute-drive
```

where 'target-drive' is the drive a package may want to access, but 'substitute-drive' is the real drive you want it to use. This looks reasonable given the other uses of CD, that is, to assign directory paths to floating drives, but even so I've never used it quite like this.

To run a package which insists on drive C: when you only have floppies you can therefore type:

```
CD C:=A:
```

after which accesses to drive C: will be redirected by DOS Plus to drive A:.

A LAST WORD

As you know I wrote in the March issue about the progress (or lack of) of the 512 Technical Guide, but that things didn't go according to plan is obvious by now. In the event I didn't get the final page proofs until the end of May.

The reasons are academic at this stage, but one fact that isn't is that I really can promise you that the book will be appearing within a couple of weeks of this issue unless the printer has a fire (perhaps I shouldn't have said that)!

Monix: A Machine Code Monitor (continued from page 29)

```
6890 :
6900 .prwrd LDAmemps+1:JSRprnm:JSRback
6910 LDAmemps:JMPprnm
6920 :
6930 .keys LDA#&81:LDY#&FF:JMPoby
6940 :
6950 .back LDA#8:JMPowr
6960 :
6970 .beep LDA#7:JMPowr
6980 :
6990 .spc LDA#32:JMPowr
7000 :
7010 :
7020 .init SEC:JSRsetup
7030 LDA#&4C:STA&287
7040 LDA#setup MOD256:STA&288
7050 LDA#setup DIV256:STA&289
7060 LDX#(t50-tbs):JMPprtxt
7070 .setup BCCcclsetup
7080 SEI:LDA#monitor MOD256:STA&200
7090 LDA#monitor DIV256:STA&201:CLI
7100 LDX#(t51-tbs):JSRprtxt
7110 .cclsetup LDA#180
7120 .setpage LDX#((end-1)DIV256)+1
7130 LDY#0:JMPoby
7140 :
7150 :
7160 :
```

```
7170 .prmvar EQUB&00
7180 .retmem EQUS STRING$(32,"-")
7190 .inline EQUS STRING$(10,"-")
7200 .srchline EQUS STRING$(10,"-")
7210 .tbs
7220  EQUB&1E:EQUS" Mnx YN +"
7230 .t0 EQUB&86:EQUS"A  X  Y  P NV-BDI
ZC PC   SP Wk":EQUW&2B0D
7240 .t1 EQUS"Address":EQUB&86:EQUS"&+"
7250 .t2 EQUW&0B1E:EQUB&2B
7260 .t3 EQUD&0A18001F:EQUB&2B
7270 .t4 EQUD&0D7F7F7F:EQUS"AXYP":EQUW&
2B86
7280 .t5 EQUS"= &+"
7290 .t6 EQUS"String":EQUW&2B86
7300 .t7 EQUB&0D:EQUS"Any key+"
7310 .noteline EQUW&0D0D:EQUB&84:EQUS"M
onix B1.8C   "
7320 EQUW&0D0D:EQUB&83:EQUS"W V D A E *
J N S B M U P Q ":EQUW&2B84
7330 .t50 EQUS"Press Break":EQUW&2B0D
7340 .t51 EQUS"Monix B1.8C":EQUW&0D0D:E
QUB&2B
9000 .end
9010 ]
9020 PRINT;~P%
9030 NEXTA%
9040 ENDPROC
```

Thanks for the Memory - Bas128 (Part 2)

by Andrew Rowland

This month we continue our exploration of Bas128 with a look at assembler. But first, two useful programming utilities.

LISTIF

Bas128 is based on Basic II, and as a Master owner, I found I greatly missed the LISTIF command. Listing 1 is the utility I wrote to replace it, and it can be adapted for Basic I and II as well. Listing 1 is not a program as such but an EXEC file. Carefully *BUILD LISTIF or type it into a text editor like Wordwise or the Master's Edit, and save normally as text with the name LISTIF. View is not suitable because it does not allow long enough lines. As it is, abbreviations are still necessary. Only press Return where indicated by <CR>.

Listing 1

```
V.21 <CR>
Z%=@%:@%=5:C%=PA.+4 <CR>
T%=C%+LEN$C% <CR>
V.6:REP.S%=T%+4:L%=LEN$S%-LEN$C%+1:F.I%=1TOL%:F%=MI.$S%,I%,LEN$C%)=$C%:I%=I%
-(F%*L%):N.:IFF% P.T%?1*256+T%?2" ";:F.I%=1TO LEN$S%:A%=ASCMI.$S%,I%):CA.&6F
B3:N.:P.:T%=T%+LEN$S%+4:U.T%?1>127 EL.T%=T%+LEN$S%+4:U.T%?1>127:V.21 <CR>
0 <CR>
@%=Z%:V.6 <CR>
```

LISTIF assumes your program does not have a line 0. To use it, enter the string you are searching for as line 0 - this ensures that keywords are tokenised. Leading and trailing spaces are significant. Then type *EXEC LISTIF. For example, if you want to list every occurrence of PRINT, enter 0PRINT then *E.LISTIF. Line 0 will be removed afterwards. To prevent it doing this, enter it as line 1 instead: the number isn't important so long as it's the first line in the program.

The utility makes use of the fact that the lines of Basic stored in memory are terminated by a Carriage Return (ASCII 13) and so can be treated as static strings. It sets up a REPEAT loop to go through each line of the program, seeing if the first line is contained in any subsequent line using an equivalent of INSTR. If it does, the whole line is displayed using the routine in Bas128 at &6FB3 which expands tokens to keywords if necessary. Owing to the limits of a one line loop, it does not display tokenised line numbers (following GOTO etc.) properly, but will still identify lines correctly.

To use it for Basic II, replace the long line in the middle with:

```
V.6:REP.S%=T%+4:IFINSTR($S%,$C%) P.T%?1*256
+T%?2" ";:F.I%=1TO LEN$S%:A%=ASCMI.$S%,I%):
CA.&B50E:N.:P.:T%=T%+LEN$S%+4:U.T%?1>127 EL
.T%=T%+LEN$S%+4:U.T%?1>127:V.21
```

For Basic I, the call address &B50E should be changed to &B53A.

EDIT

Master owners will also miss the EDIT command. Listing 2 comes to the rescue here, creating a utility called BEDIT. When you want to use the Master's editor, simply type *BEDIT and the result is the same as EDIT in Basic IV, except that the format produced by LISTO is followed (EDIT never puts spaces after the line number) and you can't specify line numbers - you always get the whole program. Returning to Bas128 is not so easy: you must save the program as text, return to Bas128 and EXEC it in - do *not* use 'Return to language'.

The program works by redirecting the OSWRCH vector, which handles writing characters to the screen. By forcing Bas128 to list the current program (lines 310-450), it is able to store the

output in a buffer, producing a pure ASCII version. The Master's Editor has a feature that Basic uses to implement the EDIT command, whereby you can inform it of the presence of text in memory. In this case, the start address is stored at &72, the end address at &70 and the Editor called with *EDIT 72,70 (line 720).

The length of program it can cope with is limited by the size of buffer available - from OSHWM to &2F00 - but is still useful. It is not possible to use this utility with other word processors or with Basic II, and this is a topic I may return to in a future article.

Listing 2

```
 10 REM Program BEDITbas
 20 REM Version B1.00
 30 REM Author  Andrew Rowland
 40 REM BEEBUG  Aug/Sept 1990
 50 REM Program subject to copyright
 60 :
100 wrchv=&20E
110 osbyte=&FFF4:oscli=&FFF7
120 pointer=&70:start=&72
130 oldv=&74:flag=&76:temp=&77
140 FOR pass=0 TO 3 STEP 3
150 P%=&A00
160 [OPT pass
170 .install \ alter WRCHV
180 LDA wrchv  :STA oldv
190 LDA wrchv+1:STA oldv+1
200 SEI:LDA #entry MOD &100
210 STA wrchv
220 LDA #entry DIV &100
230 STA wrchv+1
240 CLI
250 \ set pointers
260 LDA #2
270 STA pointer:STA start
280 LDA #180:LDX #0:LDY #&FF
290 JSR osbyte
300 STX pointer+1:STX start+1
310 \ put "L." in k/b pointer
320 LDA #21:LDX #0:JSR osbyte
330 \ X is preserved
340 .firsttime
350 LDA auto,X:JSR poke
360 INX
370 CPX #4:BNE firsttime
380 RTS \ END
390 .auto EQUS "L."+CHR$13+CHR$0
400 \ perform *FX138,0,n
410 .poke
420 TAY:TXA:PHA
430 LDA #138:LDX #0:JSR osbyte
440 PLA:TAX
450 RTS
460 \ ******************************
470 .entry
480 STY temp
490 LDY flag:BNE secondtime
500 CMP #13 :BNE out
510 INC flag:BNE out
520 .secondtime
530 CMP #0 :BEQ finished
540 CMP #10:BEQ out
550 LDY #0:STA (pointer),Y
560 INC pointer:BNE out
570 INC pointer+1
580 LDA pointer+1:CMP #&2F
590 BNE out
600 JSR restore
610 BRK:BRK
620 EQUS "Program too long":BRK
630 .out
640 LDY temp:RTS
650 :
660 .finished
670 DEC pointer:JSR restore
680 LDX #string MOD &100
690 LDY #string DIV &100
700 JMP oscli
710 .string
720 EQUS "EDIT "+STR$~start+","+STR$~p
ointer:EQUB 13
730 :
740 .restore
750 SEI
760 LDA oldv  :STA wrchv
770 LDA oldv+1:STA wrchv+1
780 CLI:RTS
790 ]NEXT
800 a$="SAVE BEDIT "+STR$~install+" "+
STR$~P%
810 PRINTa$:OSCLIa$
820 END
```

ASSEMBLER

This is an area where Bas128 really comes into its own, particularly when assembling long ROMs. Normally, space must be found in main memory for both the source code and the machine code, but Bas128's built in assembler allows you to assemble to main memory or sideways RAM using 17 bit values for P% and O%. When above &10000, the assembler automatically adjusts the addresses to &8000 - &BFFF. Any machine code

so produced can be CALLed in the usual way, ignoring the fact that it is actually in sideways RAM, as the addresses are adjusted automatically. However, it will not allow you to jump to or refer to one RAM bank from another one; and if you do, a "Bank" error is generated (error 99). If you assemble code past the end of a bank, you get a "Wrap" error (error 98).

The easiest way to assemble a ROM is to use bank Z (7 on the Master). The bank starts at &1C000, so P% should be set to this and HIMEM lowered accordingly (alternatively, use bank W (4) and raise PAGE to &14000). It is not necessary to use offset assembly. As an example, listing 3 assembles a ROM header - that is, sufficient code for it to be recognised as a ROM without performing any other function. Enter, save and run the program. You will see a ROM installed in bank 7. Press Ctrl-Break to initialise the image and try *HELP - a short message should appear. Remember that machine code located above &FFFF should be saved using *SRSAVE or the procedure called PROCsr that I presented last month.

Listing 3

```
  10 REM Program ROMhdr
  20 REM Version B1.00
  30 REM Author  Andrew Rowland
  40 REM BEEBUG  Aug/Sept 1990
  50 REM Program subject to copyright
  60 :
 100 title$="Example ROM"
 110 vs$="Version 1.00"
 120 help$="This ROM performs no
functi
ons"
 130 oswrch=&FFEE:osnewl=&FFE7
 140 :
 150 HIMEM=&1C000
 160 FOR pass%=0 TO 3 STEP 3
 170 P%=HIMEM
 180 [OPT pass%
 190 .langentry
 200 BRK:BRK:BRK
 210 JMP serventry
 220 EQUB &82:EQUB (copywr-langentry)
 230 EQUB VALRIGHT$(vs$,2) \ vs number
 240 .title EQUS title$
 250 EQUB 0:EQUS vs$
 260 .copywr
 270 EQUB 0:EQUS"(C) BEEBUG 1990"
 280 EQUB 0
 290 .serventry
 300 CMP #9:BEQ givehelp
 310 .noserv RTS
 320 :
 330 .givehelp
 340 PHA:TYA:PHA:TXA:PHA
 350 LDA (&F2),Y:CMP #&D
 360 BNE serveo:JSR osnewl
 370 LDX #0
 380 .titleloop
 390 LDA title,X:BNE over
 400 LDA #32
 410 .over
 420 JSR oswrch:INX
 430 CPX #LENtitle$+LENvs$+1
 440 BNE titleloop
 450 JSR osnewl:LDX #0
 460 .helploop
 470 LDA help,X:JSR oswrch
 480 INX:CPX #LEN(help$)
 490 BNE helploop
 500 JSR osnewl
 510 .serveo
 520 PLA:TAX:PLA:TAY:PLA
 530 RTS
 540 :
 550 .help EQUS help$
 560 ]NEXT
 570 *ROMS
 580 PRINT"Now press Ctrl Break and
try
 *HELP"
 590 END
```

ASSEMBLER AND THE MASTER

For Master owners there is the problem that Bas128 does not support 65C12 op codes, and assembling these requires some ingenuity. Listing 4 is my answer. To use it, delete any line numbers below 1000, renumber so the line numbers are higher than any used in your own assembler source programs, and append to any program you are developing. When you need a 65C12 instruction that Bas128 can't assemble, call FNass from assembler like this:

```
OPT FNass("STZ flag:BIT &70,X")
```

There are three things to keep in mind: only include the 65C12 mnemonics (including the additional addressing modes for some 6502 mnemonics) shown in table 1; the OPT directive must be initialised with the variable 'pass' and

you should normally perform a three pass assembly. In addition, mnemonics must be in capital letters, comments may not be included and labels may not be assigned, i.e. use:

```
.loop OPT FNass("JMP (address,X)")
```

NOT:

```
OPT FNass(".loop JMP (address,X)")
```

Note that where BRA is used, 'BRA loop' will assemble correctly but branches of the form 'P%-4' or 'P%+8" must be one less than in normal assembler, thus 'P%-5' and 'P%+7'.

New mnemonics

BRA	DECA/DEA	INC A/INA
CLR	STZ	PHX
PHY	PLX	PLY
TRB	TSB	

New addressing mode - zero page indirect
e.g. LDA (&70) - may be used with:

ADC	CMP	ORA
AND	EOR	SBC
LDA	STA	

Pre-indexed absolute indirect addressing
e.g. JMP (&FE00,X) - JMP only

More addressing modes with BIT:
BIT # BIT absolute,X BIT zero page,X

Table 1. Instructions recognised by FNass

EVAL is used to evaluate addresses etc., so on the first pass, it can cause "No such variable" errors. This mechanism cannot be switched off as it can in true assembly. My compromise solution is to insist on 'pass' being used to set OPT and not to evaluate at all whenever assembly errors are suppressed. Instead, the program guesses as to whether zero page addressing is intended or not, and as a result, labels can be initialised wrongly. The only solution is to perform an extra pass, with errors suppressed only on the first one. A neat way of accomplishing this is demonstrated in listing 4. See also this month's article in the series *Practical Assembler* for further information on 65C12 mnemonics.

CONCLUSION

That concludes our look at Bas128. I hope it has inspired you to experiment further. The language certainly deserves far wider use than it currently enjoys - why not send us your Bas128 hints and tips?

Listing 4

```
  10 REM Program Assembler
  20 REM Version B1.00
  30 REM Author  Andrew Rowland
  40 REM BEEBUG  Aug/Sept 1990
  50 REM Program subject to copyright
  60 :
 100 REM DEMO - do not run the code
 110 DIM code 50
 120 PROCmc(0):PROCmc(2):PROCmc(3)
 130 END
 140 :
1000 DEF PROCmc(pass)
1010 P%=code
1020 [OPT pass
1030 .loop
1040 OPT FNass("STZ loop:BIT#128:JMP (&
6,X)")
1050 ]ENDPROC
1060 :
1070 DEF FNass(A$)
1080 LOCAL B$,X,zp,immed,bracket,mnemon
ic$,arg$,addr$
1090 REPEAT B$=A$
1100 IFINSTR(B$,":") B$=LEFT$(B$,INSTR(
B$,":")-1):A$=MID$(A$,INSTR(A$,":")+1) E
LSE A$=""
1110 PROCacode(B$)
1120 UNTILA$=""
1130 =pass
1140 :
1150 DEF PROCacode(A$)
1160 A$=FNspace(A$)
1170 mnemonic$=LEFT$(A$,3):arg$=FNspace
(MID$(A$,4))
1180 A$=mnemonic$+" "+arg$
1190 IF (pass AND 1) PRINTRIGHT$("000"+
STR$~(FNtranslate(P%)),4);" ";
1200 IF mnemonic$="CLR" mnemonic$="STZ"
1210 IF mnemonic$="BRA" PROCb(&80):PROC
b(FNrelative(arg$)):PROClisting(A$):ENDP
ROC
1220 IF arg$="" OR arg$="A" PROCb(FNimp
lied(mnemonic$)):PROClisting(A$):ENDPROC
1230 X=RIGHT$(arg$,2)=",X"
1240 bracket=ASCarg$=ASC"(" .
1250 IF bracket addr$=FNspace(MID$(arg$
,2,INSTR(arg$,")")-2)):X=(X OR RIGHT$(ad
dr$,2)=",X") ELSEaddr$=arg$
1260 IF X addr$=LEFT$(addr$,LENaddr$-2)
```

```
 1270 immed=ASCarg$=ASC"#"
 1280 IF NOT(immed) ELSEIF mnemonic$="BI
T" PROCb(&89) ELSEPROCe(0,"BIT # expecte
d")
 1290 IF immed addr$=FNspace(MID$(arg$,2
)):zp=TRUE ELSE zp=(FNeval(addr$) < &100
)
 1300 IF mnemonic$="JMP" zp=FALSE
 1310 IF zp AND bracket PROCb(FNzpindire
ct(mnemonic$)) ELSE IF NOT immed PROCb(F
Nother)
 1320 IF zp PROCb(FNeval(addr$)) ELSE PR
OCw(FNeval(addr$))
 1330 PROClisting(A$)
 1340 ENDPROC
 1350 :
 1360 DEF FNrelative(B$)
 1370 IF (pass AND 2)=0 =0
 1380 LOCAL A%:A%=EVALB$-P%
 1390 IFA%>129 OR A%<-126 PROCe(1,"Out o
f range")
 1400 =A%-1
 1410 :
 1420 DEF FNimplied(A$)
 1430 LOCAL A%:A%=&1A
 1440 IF LEFT$(A$,2)="IN"=A%
 1450 IF ASCA$=ASC"P" A%=A% OR &40 ELSE=
&3A
 1460 IF MID$(A$,2,1)="L" A%=A% OR &20
 1470 IF MID$(A$,3,1)="X" A%=A% OR &80
 1480 =A%
 1490 :
 1500 DEF FNzpindirect(A$)
 1510 LOCAL A%:A%=&12
 1520 IF ASCA$=ASC"S" OR ASCA$=ASC"C" OR
 ASCA$=ASC"L" A%=A% OR &80
 1530 IF ASCA$=ASC"E" OR INSTR(mnemonic$
,"C") A%=A% OR &40
 1540 IF INSTR(mnemonic$,"D") OR RIGHT$(
A$,1)="C" A%=A% OR &20
 1550 =A%
 1560 :
 1570 DEF FNother
 1580 LOCAL A%
 1590 IF zp A%=&14 ELSE A%=&1C
 1600 IF X A%=A% OR &20
 1610 IF mnemonic$="JMP" A%=A% OR &40
 1620 IF mnemonic$="STZ" AND (NOT zp) A%
=A% OR &80:IF X THEN=&9E
 1630 IF mnemonic$="STZ" AND zp A%=A% OR
 &40:IF NOT X THEN=&64
 1640 IF mnemonic$="TSB" A%=A% AND &F
 1650 =A%
 1660 :
 1670 DEF PROCb(code%)
 1680 IF(pass AND 2)>0 IF code%>&FF PROC
e(2,"Byte")
 1690 [OPT (pass AND 4):EQUB code%
 1700 ]PROChex(code%)
 1710 ENDPROC
 1720 :
 1730 DEF PROCw(code%)
 1740 code%=FNtranslate(code%)
 1750 PROCb(code% MOD &100)
 1760 PROCb(code% DIV &100)
 1770 ENDPROC
 1780 :
 1790 DEF FNtranslate(A%)
 1800 IF A%<&10000 =A%
 1810 IFFNbank(P%)<>FNbank(A%) PROCe(98,
"Wrap")
 1820 =A% MOD &4000+&8000
 1830 :
 1840 DEF FNbank(A%)
 1850 =A% DIV &4000
 1860 :
 1870 DEF FNeval(A$)
 1880 IF (pass AND 2)=0 =&FFFF
 1890 =EVAL(A$)
 1900 :
 1910 DEF PROCe(err,err$)
 1920 $&900=CHR$0+CHR$err+err$+" in: "+A
$+CHR$0:CALL&900
 1930 ENDPROC
 1940 :
 1950 DEF PROChex(A%)
 1960 IF (pass AND 1) PRINTRIGHT$("0"+ST
R$~A%,2)+" ";
 1970 ENDPROC
 1980 :
 1990 DEF PROClisting(A$)
 2000 IF (pass AND 1) PRINTTAB(24)A$
 2010 ENDPROC
 2020 :
 2030 DEF FNspace(A$)
 2040 IFASCA$=32 REPEATA$=MID$(A$,2):UNT
ILASCA$<>32
 2050 IFRIGHT$(A$,1)=" "REPEATA$=LEFT$(A
$,LENA$-1):UNTILRIGHT$(A$,1)<>" "
 2060 =A$
```

Music 5000 Synthesiser Universal

Reviewed by Ian Waugh

Product	**Music 5000 Synthesiser Universal**
Supplier	**Hybrid Technology Ltd., 273 The Science Park, Cambridge CB4 4WE. Tel. (0223) 420360**
Price	**£113.85 inc. VAT. Software only £33.35 inc. VAT**

Although the Hybrid Music System is very popular with home-based musicians, it is even more popular in education where it is the de facto standard computer-based music system across the music and IT curriculum. This is not without good cause as it suits the needs of education particularly well and is very cost effective. There is also a growing range of music discs available, for example from Hybrid itself and from Panda Discs (see the review in BEEBUG Vol.8 No.10).

But it isn't the only music system for the BBC micro. There are now well over 50 programs which use the Beeb's sound chip, the majority of which were designed with education in mind. If you've dabbled with any of these you'll have realised that, no matter how excellent the program may be in concept and execution, the BBC's sound chip represents a very real limitation both in terms of sound quality and pitch control. And since the sounds produced by the chip do not adhere to the standard tonal range, this is a positive disadvantage when it comes to aural training programs and instrument tutors.

The Music 5000 Synthesiser Universal means an end to so-so sounds. It intercepts the commands directed at the sound chip and plays them through the Music 5000. The complete package consists of a Music 5000 Synthesiser, the Ample Nucleus ROM and software which loads into sideways RAM. The system was designed for the Master, however, and will not run on a BBC B, even one fitted with sideways RAM. The Master's numeric keypad acts as a set of hot keys to control the program and a Key Card fits over them to remind you of their functions.

USING THE PACKAGE

After generating a Start-Up disc from the Hybrid Issue disc (in usual Hybrid foolproof fashion), the software loads into sideways RAM and you are presented with a graphic control screen. Sixteen instruments are available including piano, glock, flute, organ and strings along with four drum sounds. You can assign any sound to any of the BBC's four voices using the numeric keypad and you can test the sounds by playing the QWERTY keyboard.

You can raise and lower the pitch of a voice in octave steps which is useful as many programs actually sound an octave higher than they should. There are fine tune controls, too, which let you tune the Universal to other instruments. An indicator on the screen displays the pitch graphically.

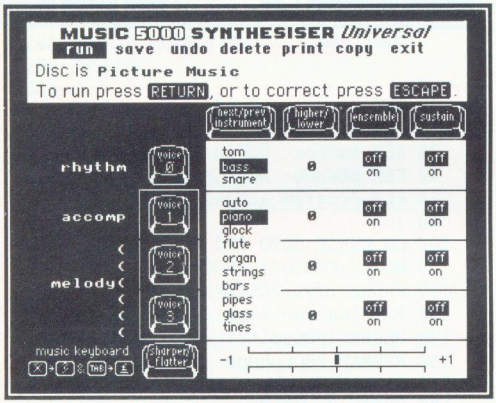
There are two other controls: *Ensemble* adds a second voice, slightly detuned, to the first, thus producing a fuller sound. *Sustain* also adds a second voice but with a delay to produce a reverberant effect. These extra voices come from the Music 5000 (which normally can play eight voices at once) so no voices are lost from the program.

The effects can be set individually for each voice by holding down the corresponding key on the numeric keypad (for example, key 1 for voice 1) and tapping the Sustain or Ensemble key. The screen shows the instruments and effects selected for each voice and you also get aural confirmation as you make a change. This is important as it means you can effectively control the Universal blind, allowing you to change the settings within a piece of software while it is running.

Once you're used to the system - which shouldn't take long - you can try it with some software. Select *Run* on the main screen and insert a music program disc. The Universal reads the disc for a fingerprint and asks for the Start-Up disc again. The fingerprint reading is quite clever as you can write to and delete from the disc without altering its fingerprint.

A list of around 50 music programs appears - which covers virtually all the commercial music

programs currently available - and you select the name of the one whose disc has just been read. If the program name is not there or if you are using a program of your own, you supply the name and it will be stored on the Start-Up disc. If the name is there, the program loads a set of default settings which will complement the program. These can still be altered using the hot keys but their provision at the start allows you to plug in and go. Many of the program settings also produce helpful information about the programs, for example how many voices it uses or how they are arranged. You can save a group of settings to disc, or delete them. You can also print the current setting on an Epson-compatible printer. This produces a small dump of the control portion of the screen.



The Music 5000 Synthesiser Universal control screen

Some programs use just one or two voices; to enhance the sound further with such programs, a *Link* facility links unused voices so that they play alongside each other. You can alter the octave and instrument of linked voices.

One of the options which appears on the instrument list is *Auto*. If you select this, the program automatically substitutes the 16 Music 5000 voices for the 16 BBC envelopes. This is especially useful if a program lets you select 'instruments' (i.e. alternate envelopes) within itself.

USE IT WITH YOUR OWN SOFTWARE

You've probably realised that the Universal will work equally well with your own Basic programs. Although it only supports four BBC voices - as this is all the BBC has! - with Sustain switched on it can actually sound eight Music 5000 voices at the same time. This can be accomplished in Basic using the Sustain option by playing two-note chords on the same channel. Obviously, none of the existing commercial programs uses this, but it opens the door to some heavyweight music software. The system is even clever enough to analyse the envelope's amplitude values and convert them into dynamics.

The Music 5000 supplied with the Universal is fully compatible with the Music 5000 supplied with the Hybrid Music System except for the addition of a special audio plug. When inserted, it re-channels the Music 5000's output to the BBC's internal speaker. How? Well, there is an external sound input on the BBC's 1 MHz bus. Not a lot of people know that.

In fact, the whole operation of the Universal is based around hooks which were designed into the BBC's sound operating system - that's what you call forward planning. The Universal works with all legally-written software; however, commands which poke the sound chip directly cannot be intercepted. The Island Logic Music System, Peter Beater's Music Games and Duette are the three main (and possibly only) programs which aren't compatible.

I initially had a problem running the software and traced this to the SpellMaster ROM, which is known to cause problems with other software sometimes (though it doesn't appear to interfere with the normal Hybrid Music System). A *UNPLUG command solved this.

The Universal makes a tremendous difference to all music software and it seems almost a crime to go back to the BBC's beep. It will, of course, work with all correctly written software which uses sound, not just music programs. As the package includes the Music 5000, you can upgrade to the full Hybrid Music System at any time.

The Universal will undoubtedly be of greatest benefit in education, for which it was primarily designed. Its ease of use coupled with its ability to play the excellent Music 5000 sounds without resource to any external amplification or music centre must make it an extremely attractive proposition. It's altogether an original, imaginative and fascinating product.

Quad

Alan Wrigley describes a game by David Pooley which will tax your ability to think quickly.

Quad is a fairly simple but extremely fascinating game. It is similar to the arcade game *Tetris*, which many gamesters will have played.

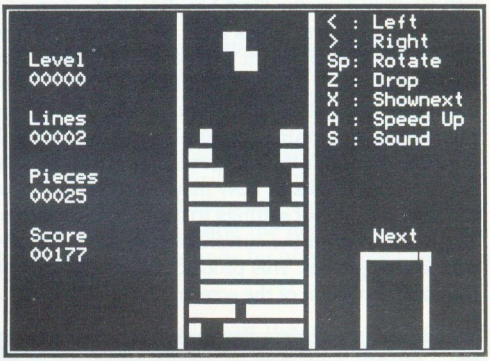
The object of the game is to fill a rectangle with shapes which drop from the top of the screen. This is done by rotating the shape and moving it laterally so that hopefully it drops into the space you have chosen for it. The aim is to fill the rectangle with each shape interlocking and no spaces left between. This is more difficult than it sounds, since the shapes appear in a random order, and you have to decide very quickly where they are to be placed.

Type in Listing 1 and save it as Quad. Assembler is used in places to speed up the screen handling, so take particular care with these lines. You can now play the game without further ado. It will run on any BBC micro, including a model B.

PLAYING THE GAME

After the title screen has appeared, you will see a screen describing the keys which can be used with the game. The < and > keys move the shape left or right as appropriate, while the space bar rotates it. If you are satisfied the alignment is correct you can press Z and the shape will drop into place. Pressing X will show the next shape due to appear; this may help you to decide where to place the current one. Finally, S turns the sound on or off, and A speeds up the game.

Before playing, you must choose the difficulty level, from 0 to 9, which sets the speed at which the shapes drop. If you have not played the game before, then 0 is probably a good choice! The game itself is quite simple to play but not at all easy. However, if you complete a line across the rectangle, that line is removed, thus allowing you to fill gaps underneath that may previously have been inaccessible. The score is kept on the left-hand side of the screen - see if you can get your name into the high-score table!



The game in progress

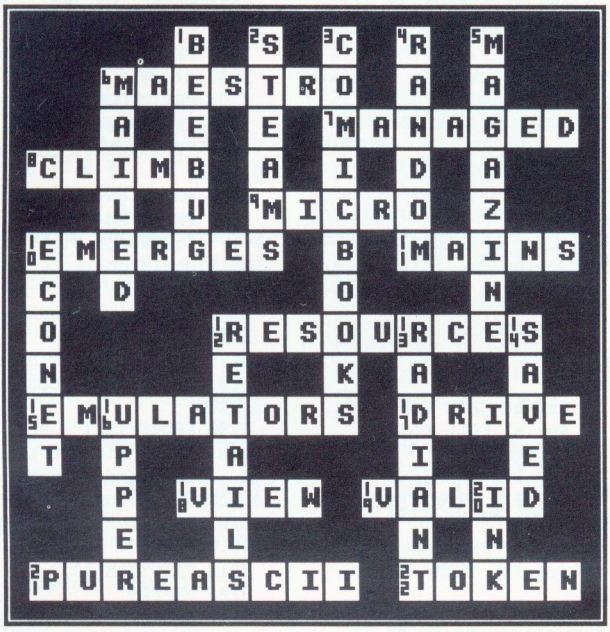
```
   10 REM          >Quad
   20 REM Version B1.0
   30 REM Author  David Pooley
   40 REM BEEBUG  Aug/Sept 1990
   50 REM Program subject to copyright
   60 :
  100 ON ERROR GOTO 230
  110 MODE7:PROCoff:PROCtitle
  120 PROCarrays:PROCassemble
  130 PROCinstructions
  140 REPEAT:PROCdifficulty:PROCscreen
  150 REPEAT:PROCshape
  160 REPEAT:PROCkeys:PROCdown
  170 UNTIL dropped
  180 PROCscore:UNTIL full
  190 PROCgame_over:PROChigh_scores
  200 UNTIL FNfinished:PROCend
  210 END
  220 :
  230 ON ERROR OFF
  240 VDU26,12
  250 IF ERR<>17 REPORT:PRINT" at line "
;ERL ELSE PROCend
  260 END
  270 :
 1000 DEF PROCoff
 1010 ENVELOPE 1,1,0,0,0,0,0,0,100,-1,-1
,-1,100,80
 1020 VDU23;8202;0;0;0;:*FX 210,0
 1030 ENDPROC
 1040 :
 1050 DEF PROCtitle
 1060 FOR Y%=0 TO 24
 1070 VDU 31,0,Y%,145+Y% MOD 7:NEXT Y%
 1080 FOR Letter%=0 TO 3
```

```
 1090 VDU 28,8*Letter%+4,24,39,0
 1100 Letter$=MID$("QUAD",Letter%+1,1)
 1110 PROCget_pattern(Letter$)
 1120 FOR Row%=&78 TO &71 STEP -1
 1130 FOR Bit%=0 TO 7
 1140 IF (?Row% AND 128) VDU255 ELSE VDU
32
 1150 ?Row%=?Row%*2:NEXT Bit%
 1160 PROCwait(4):VDU 11,13:NEXT Row%
 1170 FOR Scroll%=1 TO 9
 1180 PROCwait(5):VDU 30,11
 1190 NEXT Scroll%:NEXT Letter%
 1200 ENDPROC
 1210 :
 1220 DEF PROCget_pattern(Letter$)
 1230 ?&70=ASC(Letter$)
 1240 A%=10:X%=&70:Y%=0:CALL &FFF1
 1250 ENDPROC
 1260 :
 1270 DEF PROCwait(Time%)
 1280 TIME=0:REPEAT UNTIL TIME>Time%
 1290 ENDPROC
 1300 :
 1310 DEF PROCarrays
 1320 DIM High%(7),High$(7)
 1330 FOR High%=0 TO 7
 1340 High%(High%)=5000-High%*500
 1350 High$(High%)="Quaddie "+STR$(High%
+1)
 1360 NEXT High%
 1370 ENDPROC
 1380 :
 1390 DEF PROCassemble
 1400 code=&900:shapes=&B00:setchar=175
 1410 shape=&70:rot=&71:start=&72
 1420 char=&74:count=&75:flag=&76
 1430 down1=&77:down2=&79
 1440 FOR pass%=0 TO 2 STEP 2
 1450 P%=code
 1460 [OPT pass%
 1470 .print:JSR offset:LDA #4:STA count
:LDA char:.printloop:LDY shapes,X:STA (s
tart),Y:INX:DEC count:BNE printloop:RTS
 1480 .offset:LDA shape:ASL A:ASL A:ADC
rot:ASL A:ASL A:TAX:RTS
 1490 .check:JSR offset:LDA #4:STA count
:LDA #0:STA flag:.checkloop:LDY shapes,X
:LDA (start),Y:CMP #32:BEQ ok:CMP #255:B
EQ ok:DEC flag:RTS
 1500 .ok:INX:DEC count:BNE checkloop:RT
S
 1510 .right:JSR add1:JSR check:JSR take
1:LDA flag:BEQ moveright:RTS:.moveright:
LDA #32:STA char:JSR print:JSR add1:LDA
#255:STA char:JSR print:RTS
 1520 .left:JSR take1:JSR check:JSR add1
:LDA flag:BEQ moveleft:RTS:.moveleft:LDA
 #32:STA char:JSR print:JSR take1:LDA #2
55:STA char:JSR print:RTS
 1530 .down:JSR add40:JSR check:JSR take
40:LDA flag:BEQ movedown:RTS:.movedown:L
DA #32:STA char:JSR print:JSR add40:LDA
#255:STA char:JSR print:RTS
 1540 .rotate:INC rot:LDA rot:AND #3:STA
 rot:JSR check:DEC rot:LDA rot:CLC:ADC #
4:AND #3:STA rot:LDA flag:BEQ makerot:RT
S
 1550 .makerot:LDA #32:STA char:JSR prin
t:INC rot:LDA rot:AND #3:STA rot:LDA #25
5:STA char:JSR print:RTS
 1560 .add1:INC start:BNE over:INC start
+1:.over:RTS
 1570 .take1:SEC:LDA start:SBC #1:STA st
art:LDA start+1:SBC #0:STA start+1:RTS
 1580 .add40:CLC:LDA start:ADC #40:STA s
tart:LDA start+1:ADC #0:STA start+1:RTS
 1590 .take40:SEC:LDA start:SBC #40:STA
start:LDA start+1:SBC #0:STA start+1:RTS
 1600 .lines:LDA #0:STA count:LDA #&5F:S
TA start:LDA #&7C:STA start+1:.nextline:
LDY #9:.lineloop:LDA (start),Y:CMP #setc
har:BNE nodown:DEY:BPL lineloop:INC coun
t:LDA start:STA down1:LDA start+1:STA do
wn1+1
 1610 JSR take40:LDA start:STA down2:LDA
 start+1:STA down2+1:JSR add40:.scroll:L
DY #9:.downline:LDA (down2),Y:STA (down1
),Y:DEY:BPL downline:LDA down2:STA down1
:LDA down2+1:STA down1+1
 1620 SEC:LDA down2:SBC #40:STA down2:LD
A down2+1:SBC #0:STA down2+1:LDA down2+1
:CMP #&7B:BNE scroll:.nodown:JSR add40:L
DA start+1:CMP #&80:BNE nextline:RTS
 1630 .slide:LDA #19:JSR &FFF4:LDA #&90:
STA start:LDA #&7D:STA start+1:.slideloo
p1:LDY #38:.slideloop2:LDA (start),Y:INY
:STA (start),Y:DEY:DEY:BPL slideloop2:JS
R add40
 1640 LDA start+1:CMP #&7F:BNE slideloop
1:RTS
 1650 ]
 1660 NEXT pass%
 1670 FOR data%=shapes TO shapes+111
 1680 READ ?data%:NEXT data%
 1690 DATA 40,41,81,82,2,41,42,81,41,42,
82,83,42,81,82,121
 1700 DATA 40,41,42,43,2,42,82,122,80,81
,82,83,1,41,81,121
 1710 DATA 41,42,80,81,1,41,42,82,42,43,
81,82,41,81,82,122
 1720 DATA 40,41,42,82,2,42,81,82,41,81,
82,83,41,42,81,121
 1730 DATA 41,42,43,81,41,42,82,122,42,8
0,81,82,1,41,81,82
 1740 DATA 40,41,42,81,2,41,42,82,42,81,
82,83,41,81,82,121
```

```
 1750 DATA 41,42,81,82,41,42,81,82,41,42
,81,82,41,42,81,82
 1760 PROCwait(100):FOR Slide%=1 TO 36
 1770 CALL slide:NEXT Slide%
 1780 VDU26:ENDPROC
 1790 :
 1800 DEF PROCinstructions
 1810 PROCdouble("Quad",4,7,0,TRUE)
 1820 PROCwrite("  Try to fit the fallin
g shapes into the already existing patte
rn on the screen by manipulating them wi
th the following keys :",6,3)
 1830 PROCpress("<","move the shape left
",3,8)
 1840 PROCpress(">","move the shape righ
t",3,9)
 1850 PROCpress("SPACE","rotate the shap
e",3,10)
 1860 PROCpress("Z","drop the piece into
 place",3,11)
 1870 PROCwrite("  The following keys ca
n be used during the game for other effe
cts :",6,13)
 1880 PROCpress("X","show the next shape
",1,16)
 1890 PROCpress("A","speed the game up",
1,17)
 1900 PROCpress("S","turn the sound on/o
ff",1,18)
 1910 PROCwrite(" Press the space bar to
 start the game",7,21)
 1920 PROCspace:ENDPROC
 1930 :
 1940 DEF PROCdouble(Double$,Back%,Fore%
,Start%,Clear%)
 1950 IF Clear% THEN CLS
 1960 FOR Print%=0 TO 1
 1970 PRINTTAB(16-LEN(Double$)/2,Start%+
Print%);CHR$(128+Back%);CHR$141;CHR$157;
CHR$(128+Fore%);Double$;SPC2;CHR$156
 1980 NEXT Print%
 1990 ENDPROC
 2000 :
 2010 DEF PROCwrite(Write$,Col%,Start%)
 2020 REPEAT:IF LEN(Write$)<40 THEN PRIN
TTAB(0,Start%);CHR$(128+Col%);Write$:Wri
te$=""
 2030 IF LEN(Write$)>39 Pos%=41:REPEAT:P
os%=Pos%-1:UNTIL MID$(Write$,Pos%,1)=" "
:PRINTTAB(0,Start%);CHR$(128+Col%);LEFT$
(Write$,Pos%-1):Start%=Start%+1:Write$=M
ID$(Write$,Pos%+1)
 2040 UNTIL Write$="":ENDPROC
 2050 :
 2060 DEF PROCpress(Key$,Action$,Col%,St
art%)
 2070 PRINTTAB(4,Start%);CHR$(128+Col%);
Key$;STRING$(30-LEN(Key$+Action$),".");A
ction$
 2080 ENDPROC
 2090 :
 2100 DEF PROCspace
 2110 *FX 15,0
 2120 REPEAT:Key=GET:UNTIL Key=32
 2130 ENDPROC
 2140 :
 2150 DEF PROCdifficulty
 2160 PROCdouble("Input Difficulty",4,7,
0,TRUE)
 2170 PROCwrite("  Please enter the diff
iculty level at which you want the game
to start. You may choose any number betw
een 0 (easy) and 9 (very hard)",6,4)
 2180 PRINT''TAB(7)CHR$&83"Starting Diff
iculty ? ";
 2190 REPEAT:D%=GET-48
 2200 UNTIL D%>=0 AND D%<=9
 2210 VDUD%+48:PROCwait(10)
 2220 L%=0:P%=0:S%=0:N%=RND(7)-1
 2230 W%=FALSE:ENDPROC
 2240 :
 2250 DEF PROCscreen
 2260 CLS:FOR Y%=0 TO 22
 2270 VDU31,13,Y%,145+Y% MOD 7
 2280 IF Y%>1 AND Y%<22 PRINT"5"SPC10"j"
CHR$&87
 2290 IF Y%=22 PRINT"-"STRING$(10,",")".
"
 2300 NEXT Y%
 2310 PRINTTAB(29,17)CHR$&93"<,,,,l"
 2320 FOR Y%=18 TO 21
 2330 PRINTTAB(29,Y%)CHR$&93"5    j"
 2340 NEXT Y%
 2350 PRINTTAB(29,22)CHR$&93"-,,,,."
 2360 PRINTTAB(30,16)CHR$&86"Next"
 2370 PRINTTAB(27,5)"< : Left"TAB(27,6)"
> : Right"TAB(27,7)"Sp: Rotate"TAB(27,8)
"Z : Drop"TAB(27,9)"X : Shownext"TAB(27,
10)"A : Speed Up"TAB(27,11)"S : Sound"
 2380 PRINTTAB(0,7)CHR$&86"Level"TAB(0,1
0)CHR$&86"Lines"TAB(0,13)CHR$&86"Pieces"
TAB(0,16)CHR$&86"Score"
 2390 PROCnumbers
 2400 ENDPROC
 2410 :
 2420 DEF PROCnumbers
 2430 PROCnum(0,8,1,D%)
 2440 PROCnum(0,11,1,L%)
 2450 PROCnum(0,14,1,P%)
 2460 PROCnum(0,17,1,S%)
 2470 B%=20-D%*3:IF D%>3 B%=17-D%*2:IF D
%>7 B%=10-D%
 2480 ENDPROC
 2490 :
 2500 DEF PROCnum(X%,Y%,C%,N%)
 2510 PRINTTAB(X%,Y%)CHR$(128+C%);RIGHT$
```

```
("00000"+STR$(N%),5)
 2520 ENDPROC
 2530 :
 2540 DEF PROCshape
 2550 ?shape=N%:N%=RND(7)-1:C%=0
 2560 ?rot=0:?start=&3A:?(start+1)=&7C
 2570 ?char=255:CALL print
 2580 IF W% PROCshow_next(N%,255)
 2590 dropped=FALSE:TIME=0
 2600 ENDPROC
 2610 :
 2620 DEF PROCshow_next(S%,C%)
 2630 !&80=!&70:!&84=!74
 2640 ?shape=S%:?rot=0
 2650 ?start=&EF:?(start+1)=&7E
 2660 ?char=C%:CALL print
 2670 !&70=!&80:!&74=!&84
 2680 ENDPROC
 2690 :
 2700 DEF PROCkeys
 2710 Key=INKEY(0)
 2720 IF Key=44 OR Key=60 CALL left
 2730 IF Key=46 OR Key=62 CALL right
 2740 IF Key=32 CALL rotate
 2750 IF Key=90 OR Key=122 PROCdrop
 2760 IF Key=88 OR Key=110 W%=NOT(W%):PR
OCshow_next(N%,32-223*W%)
 2770 IF Key=65 OR Key=97 PROCfaster
 2780 IF Key=83 OR Key=115 THEN *FX 210,
1,255
 2790 ENDPROC
 2800 :
 2810 DEF PROCdrop
 2820 REPEAT:CALL down
 2830 dropped=?flag:UNTIL dropped
 2840 ENDPROC
 2850 :
 2860 DEF PROCfaster
 2870 D%=D%-(D%<10)
 2880 PROCnumbers:ENDPROC
 2890 :
 2900 DEF PROCdown
 2910 C%=C%+1:IF C%>B% CALL down:dropped
=?flag:C%=0
 2920 ENDPROC
 2930 :
 2940 DEF PROCscore
 2950 S%=S%+(1000-TIME)/40*(D%/40+1)
 2960 SOUND 1,1,255-12*(((!start AND &FF
FF)-&7C00) DIV 40),2
 2970 P%=P%+1:?char=setchar:CALL print
 2980 CALL lines:L%=L%+?count
 2990 IF L% DIV 10>D% D%=L% DIV 10:IF D%
>10 D%=10
 3000 PROCshow_next(N%,32)
 3010 ?start=&3A:?(start+1)=&7C
 3020 ?shape=N%:?rot=0
 3030 CALL check:full=?flag
 3040 PROCnumbers:ENDPROC
 3050 :
 3060 DEF PROCgame_over
 3070 PROCdouble("Game  Over",4,7,19,FAL
SE)
 3080 PROCspace:ENDPROC
 3090 :
 3100 DEF PROChigh_scores
 3110 IF S%>High%(7) PROCenter_name
 3120 PROCdouble("Quad High Scores",4,7,
0,TRUE)
 3130 FOR High%=0 TO 7
 3140 PRINTTAB(4,High%*2+4);CHR$&81;High
%+1;".";CHR$&86;High$(High%);STRING$(20-
LEN(High$(High%)),".");CHR$&83;High%(Hig
h%)
 3150 NEXT High%:ENDPROC
 3160 :
 3170 DEF PROCenter_name
 3180 PROCdouble("Congratulations!",6,4,
3,TRUE)
 3190 PROCwrite("  Your score is high en
ough to entitle you to a place on the hi
gh score board of today's players. Pleas
e enter your name below :",3,7)
 3200 PRINTTAB(8,13);CHR$&82;"|";STRING$
(20,".");"|"
 3210 PRINTTAB(10,13);:Input$=""
 3220 REPEAT
 3230 VDU10,94:Key=GET:VDU127,11
 3240 IF Key>31 AND Key<127 AND LEN(Inpu
t$)<20 THEN VDUKey:Input$=Input$+CHR$(Ke
y)
 3250 IF Key=127 AND LEN(Input$)>0 THEN
VDU127,46,8:Input$=LEFT$(Input$,LEN(Inpu
t$)-1)
 3260 UNTIL Key=13
 3270 High$(7)=Input$:High%(7)=S%
 3280 P%=6:REPEAT
 3290 F%=TRUE:IF High%(P%)<High%(P%+1) T
HEN F%=FALSE:T$=High$(P%):High$(P%)=High
$(P%+1):High$(P%+1)=T$:T%=High%(P%):High
%(P%)=High%(P%+1):High%(P%+1)=T%
 3300 P%=P%-1:UNTIL F% OR P%<0
 3310 ENDPROC
 3320 :
 3330 DEF FNfinished
 3340 PROCdouble("Another Game ? (Y/N)",
2,7,21,FALSE)
 3350 REPEAT:Key=INSTR("YyNn",GET$)
 3360 UNTIL Key>0:=(Key>2)
 3370 :
 3380 DEF PROCend
 3390 PROCdouble("Thanks For Playing",1,
3,12,TRUE)
 3400 ENDPROC
```

Solution to BEEBUG Crossword (Vol.9 No.3).



The first correct solution drawn out of the hat was from S.A. Elwell-Sutton of Ellesmere.

Our congratulations go to Mr Elwell-Sutton, who will be receiving a copy of Panda's Crossword shortly as his prize.

Crossword Review Update

In our review of Crossword by Panda Discs (Vol.9 No.3) we suggested that you could not print out the clues with the crossword grid. In fact, the problem only occurred on a networked printer, and by the time you read this, should in any case have been cured completely. We were also quoted the wrong price for Crossword. It should be £14.95, and not £12.95 as stated. Our apologies to Panda and their customers for any confusion we may have caused. We also understand that two discs of puzzles are available, *Cryptic* for adults, and *Junior* for easier puzzles.

B

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

PHONE CALL COSTING

(BEEBUG Vol.9 No.3)

Two errors affecting this program unfortunately crept into the listings. In lines 1940 and 1960 the accented letter 'E' including the quotes should be replaced by CHR$131 in each case (teletext control codes are unprintable as such). Secondly, ENDPROC in lines 1740 and 1870 should be changed to: =flag

B

Practical Assembler (Part 4)

by Bernard Hill

CALL, USR AND VARIABLE STORAGE

Back in the first article in this current series of assembler articles we looked at ways of interfacing to assembler, and now I want to look in more detail at the method of interfacing to Basic via the keywords USR and CALL.

At the elementary level there is a good deal of similarity between these two ways of calling an assembler routine. Let's look at a simple example to make this clear: a routine to clear all the resident integer variables A% to Z%, each of which occupies four bytes, starting at &404 for the first byte of A% and finishing at &46B for the last byte of Z%. These are of course not cleared by the CLEAR statement, and in fact are preserved over a hard or soft Break, so a routine to wipe them is a common element in many Basic toolkits.

The routine is very simple:

```
10 DIM clear 10
20 P%=clear
30 [OPT 2
40 LDX #&67:LDA #0
50 .loop STA &404,X
60 DEX:BPL loop
70 RTS
80 ]
```

Notes:

1. We don't need to do two passes through the code as we haven't any forward variable references.
2. The reason we more often count down to zero in assembler rather than up to a fixed limit is that the DEX instruction sets the zero and negative flags so that we don't need an explicit CPX instruction to see if we've reached the value we want to stop at. You then use BPL at line 70 if you want to loop back for positive or zero X, and BNE if you want to loop back only for positive X.
3. Any routine which is to be CALLed or USR'd must end with RTS (line 70) to return to Basic.

Now this little routine can be triggered with a simple:

```
CALL clear
```

Or you can use:

```
a%=USR(clear)
```

The only difference is that USR returns an answer and CALL does not. This integer answer consists of four distinct bytes which are most conveniently separated by assigning it to an indirected location such as &70 with:

```
!&70=USR(clear)
```

and accessing the individual bytes, which will be:

&70	The A register on exit
&71	The X register on exit
&72	The Y register on exit
&73	The flags register on exit.

This latter is made up of 8 individual bits representing the 7 processor flags. Of these the most useful to us are the C,Z and N flags which are found in bits 0,1 and 7 of &73 respectively.

In particular, after !&70=USR(clear) above you should find that:

&70=0	(the A-register)
&71=&FF	(X has been decremented from 0),
&72=0	(Y has not been used)

and the N flag is set (by the last DEX operation).

The USR function is most often used for debugging as the values it produces can provide insight into the CPU registers and flags. But the most commonly used machine code call is a simple CALL statement.

THE CALL INSTRUCTION

CALL is, however, much more powerful than USR because it can take parameters. The parameters can be of any type as long as they are a variable name (e.g. A% or *ans$*) rather than a constant value. The CALLed routine can be written to discover the type (string, integer, etc) of the parameters before using them. As an example of this, and as an exercise in building a program from constituent parts, in the next article we shall be working our way towards building a machine code sort routine which can sort string, integer or real arrays. It will

determine for itself the type of parameter it has been given and perform the sort accordingly. This will involve us in writing and testing smaller program sections and is a good indication of how longer programs are built. But before we start this, let's look more carefully at the CALL operation.

When CALL is performed, a parameter information block is set up at &600. This will contain information which the CALLed routine can look at and use. It goes as follows:

&600	contains the number of parameters
&601-2	contain the address of the first parameter, and
&603	contains its type.

The last three bytes are repeated for as many parameters as are used, so that &604-&606 would contain the same information for a second parameter and so on.

As far as BBC Basic is concerned, there are five types of variable, and these have codes in the type byte (&603, &606 etc) as follows:

0	A 1-byte number, e.g. ?x% or a?7
4	A 4-byte integer, e.g. Num% or A%
5	A 5-byte real, e.g. x
128	A static string, e.g. $x
129	A dynamic string, e.g. A$

Note that it is not possible to pass a value parameter to a routine with CALL, such as:

```
CALL routine,55
```

Instead, the 55 must be assigned to a variable first, and then called with the variable as parameter:

```
x%=55
CALL routine,x%
```

Now the 55 would be found by loading the value at the location pointed to by &601-602:

```
LDA &601:STA &70
LDA &602:STA &71
LDY #0:LDA (&70),Y
```

The A register would now contain the lowest byte of x%, i.e. 55, and &600 would contain 1 (1 parameter) while &603 would be 4 for a 4-byte integer.

Listing 1 is a simple machine code routine to find where Basic is storing any named variable (including an element of an array). It contains a number of sample calls to illustrate its use. As an exercise you could easily extend the program to print the parameter type byte from &603 too.

You can use this program to investigate the way in which Basic stores its variables by running Program 1, creating a variable or two, calling the routine and printing the results:

```
>x%=1000
>CALL where,x%
1C61            (or whatever)
>PRINT ~?&1C61,~?&1C62,~?&1C63,~?&1C64
      E8        3        0        0
```

This result is easy to understand because 1000=&3E8 and Basic stores its integer variables in signed format in four bytes, low byte first. *Signed format* means that a negative number has its most significant bit of its last byte set, so that, for instance:

```
                    low.....high
-1 is stored as  FF FF FF FF
-2 is stored as  FE FF FF FF etc
```

See BEEBUG Vol.6 No.6 p.30 for a fuller explanation of signed format.

Real variables have a more complex storage scheme and take five bytes. We will take a much closer look at this in the next article when we also consider the storage of string variables and the problem of finding out (in assembler) whether one Basic variable is larger or smaller than another - rather essential for a sort routine!

HINTS AND TIPS

65C12 MNEMONICS

The 65C12 processor (used on the Master) is pin-compatible with the 6502, but has a number of extra instructions and addressing modes available to it which the standard 6502 CPU hasn't got. It is available for about £9 from Watford Electronics. Basics IV and VI already support these new instructions and they are all most useful additions to the standard 6502 set. Besides PHX,PHY,PLX and PLY which push and pull the X and Y registers directly to the stack there is INA and DEA (or INC A and DEC A) and others.

Sadly, Acorn never released a Basic IV to use in a Model B with 65C12 but those of you with

Basic II can use these instructions in your own programs by means of the macros of Listing 2. These function definitions should be appended (renumbered if necessary) to any program in which you wish to use them. Of course, if you haven't got a 65C12 processor they won't work but you can always substitute the 65C12 for your old 6502 - but be careful taking it out, the 40 pin chips are worse than eproms - use a Bic biro top to lever out gradually from each end.

So if you fancy upgrading your processor and trying the new macros, Listing 2 gives the coding for the commoner new ones. In order to stay compatible with the Model B and Basic II, I shan't make use of them in further articles, but you can use them in your own programs if you wish. To illustrate, the common sequence:

```
PHA:TXA:PHA:TYA:PHA
```

might be replaced in Basic IV with:

```
PHA:PHX:PHY
```

and in Basic II with the macros:

```
PHA:EQUS FNphx:EQUS FNphy
```

Listing 1

```
  10 REM Program 1
  20 REM Version B1.0
  30 REM Author  Bernard Hill
  40 REM Beebug  Aug/Sept 1990
  50 REM Program subject to copyright
  60 :
 100 PROCassemble
 110 DIM x 1,y(2)
 120 s$="BEEBUG"
 130 CALL where,x
 140 CALL where,$x
 150 CALL where,?x
 160 CALL where,x?1
 170 CALL where,y(0)
 180 CALL where,s$
 190 CALL where,?&70
 200 END
 210 :
1000 DEF PROCassemble
1010 DIM where 100
1020 FOR opt=0 TO 2 STEP 2
1030 P%=where
1040 [OPT opt
1050 LDA &600:CMP #1:BEQ over
1060 BRK:EQUB 0:EQUS "Bad call":EQUB 0
1070 .over LDA &602:JSR hex
1080 LDA &601:JSR hex:JSR &FFE7:RTS
1090 \ hex printing routine
1100 .hex PHA
1110 ROR A:ROR A:ROR A:ROR A
1120 JSR hexout:PLA
1130 .hexout AND #&F
1140 TAX:LDA hx,X:JSR &FFE3:RTS
1150 .hx EQUS "0123456789ABCDEF"
1160 ]
1170 NEXT
1180 ENDPROC
```

Listing 2

```
   10 REM LISTING 2: New macros
 1000 New address mode: indirect zero
 1010 page without Y, e.g. LDA (&70)
 1020 DEF FNadcind(z)=FNop2(&72,z)
 1030 DEF FNandind(z)=FNop2(&32,z)
 1040 DEF FNcmpind(z)=FNop2(&D2,z)
 1050 DEF FNeorind(z)=FNop2(&52,z)
 1060 DEF FNldaind(z)=FNop2(&B2,z)
 1070 DEF FNoraind(z)=FNop2(&12,z)
 1080 DEF FNsbcind(z)=FNop2(&F2,z)
 1090 DEF FNstaind(z)=FNop2(&92,z)
 1100 :
 1110 Branch always: BRA address
 1120 DEF FNbra(m)=FNop2(&80,m-P%-2)
 1130 :
 1140 REM INA/DEA/PHX/PHY/PLX/PLY
 1150 DEF FNina=CHR$&1A
 1160 DEF FNdea=CHR$&3A
 1170 DEF FNphx=CHR$&DA
 1180 DEF FNphy=CHR$&5A
 1190 DEF FNplx=CHR$&FA
 1200 DEF FNply=CHR$&7A
 1210 :
 1220 JMP address,X
 1230 DEF FNjmpx(m)=FNop3(&4C,m)
 1240 :
 1250 Store zero in: STZ address and
 1260                STZ address,X
 1270 DEF FNstz(m):IF m<256 THEN =FNop2(
&64,m) ELSE =FNop3(&9C,m)
 1280 DEF FNstzx(m):IF m<256 THEN =FNop2
(&74,m) ELSE =FNop3(&9E,m)
 1290 :
 1300 TRB/TSB address
 1310 ...see Beebug Vol.6 No.7 p.29
 1320 DEF FNtrb(m):IF m<256 THEN =FNop2(
&14,m) ELSE =FNop3(&1C,m)
 1330 DEF FNtsb(m):IF m<256 THEN =FNop2(
4,m) ELSE =FNop3(&C,m)
 1340 :
 1350 DEF FNop2(a,b)=CHR$a+CHR$b
 1360 DEF FNop3(a,b)=CHR$a+CHR$b+CHR$(b
DIV 256)
```

RISC USER

The Archimedes Magazine & Support Group

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

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

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

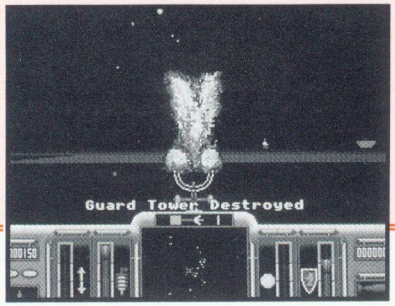
Here are just some of the topics covered in the most recent issues of RISC User:

WIMP MESSAGE MONITOR
This program allows you to monitor the activities of the Wimp as it deals with multi-tasking window-based applications.

BACKGROUND TEXT PRINTER
Use this as a substitute for your normal printer driver, so that all text files may be printed as background tasks while you continue with other work.

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

ARCADE
The latest round-up of new games for the Archimedes, including *The Pawn* and *Guild of Thieves* from Magnetic Scrolls, and *Apocalypse* (one of the very best games yet for the Arc) from Fourth Dimension.



POSTER FROM 4MATION
A review of this most impressive package for manipulating and distorting text displays, art work, posters and the like.

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

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

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

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

INTO THE ARC
A regular series for beginners. The latest article discusses the ins and outs of using printer drivers on the Archimedes for output.

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

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

SUBSCRIPTION DETAILS

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

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

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

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

SplineText Revisited

by Alan Wrigley

One of the more popular programs published in BEEBUG recently was the SplineText utility (Vol.9 No.2), which produced attractive Helvetica lettering in any size on a printer, using a kind of outline font. A few points have arisen regarding the program, which will be dealt with in this article, and I will also describe how to print more than one line of text on a sheet of paper, which was not possible with the original.

Firstly, some readers have had a problem with extra linefeeds occurring in the middle of letters. This was due to an oversight on our part after we had made some modifications to the program, and was not the fault of the author. You may think that typing *FX6,10 will cure this, but in fact it has no effect. The solution is to amend line 2440 of the program *Spline2*. If your printer produces automatic linefeeds, this line should now read:

```
2440 VDU1,27,1,51,1,L%,1,51,1,13
```

If it does not, then no modifications should be necessary.

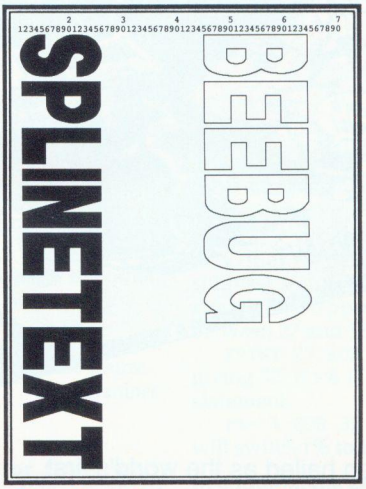
A further potential pitfall spotted by a hawk-eyed reader is concerned with entering the DATA statements at the end of the program. You must make sure that no spaces are typed at the end of any DATA line, i.e. after the "E". If you do so, the program goes on to read the next DATA statement, which it should not do.

Several readers have asked if the program could be modified to print across the paper rather than down; unfortunately this is not possible without major surgery. The program works by first drawing a single character on the screen and then dumping it to the printer. As each character is treated separately, in order to print across the page it would be necessary for the carriage to be reversed after each letter is printed. Although this is possible on some printers there are many which cannot do this. Also, the orientation either of the dump, or of the screen drawing routine, would need to be rotated through 90 degrees.

As published, the program was not capable of printing more than one line of text on a sheet, although the accompanying illustration may have implied that it was. However, it is a simple matter to print more than one line (though still vertically), provided that you are happy to feed the paper through a second time. If you include the following two lines:

```
141  VDU 1,27,1,108,1,margin%
1141 INPUT'"Left margin :"margin%
```

then you will be able to specify the left-hand margin when printing. The prompt will appear after you have chosen whether or not to fill the letters. The text will then be printed vertically with the bottom edge of the characters aligned on the margin you have specified.



Two lines of text on one sheet, with the margin settings shown

A little bit of trial and error will be necessary to get this right. Figure 1 shows an example, with one line of text printed at a margin setting of 1, and another line with a margin of 45.

Finally, I would like to make a plea. If anyone has written the necessary data to complete the ASCII character set, or indeed has compiled the data for different font styles, we will be happy to publish this information at our usual contributors' rates. So why not dig out that graph paper and start designing?

Computing should be fun

...and with this, it is!

Let's Compute! has been hailed as the world's first computer comic. But it's much more – a full-colour, brilliantly illustrated and cleverly written way of tapping into the interest young people have in the world of computing. It's now at your newsagents, but you can get the next 12 issues by sending £12 to: *Let's Compute!* Database Direct, FREEPOST, Ellesmere Port, South Wirral L653EB.

It's on sale NOW!

POSTBAG

LOOKS GOOD, SOUNDS GOOD

Derek Gibbons' Master Display ROM (Vol.8 Nos.8 & 9) makes your Master wake up with tailor-made start-up messages, and even your choice of screen colours. It looks good - why not make it *sound* good? The BBC's BEL character (ASCII 7) is an uncivilised poop. If you add the lines below to his program (either listing 1 or 2), your machine will henceforth make a melodious chime - test it with Ctrl-G. The assembled code performs the same as:

```
ENV. 1,1,0,0,0,0,0,0,126,255,0,255,126,0
*FX212,0
*FX213,200
*FX214,1

1271 LDX #envdata MOD &100
1272 LDY #envdata DIV &100
1273 LDA #8:JSR &FFF1
1274 LDA #212:LDX #0   :LDY #0:JSR &FFF4
1275 LDA #213:LDX #200:LDY #0:JSR &FFF4
1276 LDA #214:LDX #1   :LDY #0:JSR &FFF4
1641 .envdata
1642 EQUD &101:EQUD &0
1643 EQUD &FF00FF7E
1644 EQUW &7E
```

Andrew Rowland

Our thanks to Mr.Rowland for this interesting update to a popular program.

A VIEW ON WORDWISE PLUS-2

BEEBUG Vol.9 No.3 has just arrived. In your review of View you say that *delete line* is not found in Wordwise. Strictly true of course, but it should be mentioned that it IS in Wordwise Plus2, by using Ctrl-L. In the Plus-2 version only Ctrl-H and Ctrl-N have nothing to do. Ctrl and the other 24 letter keys all do something, as well as '@'.

Finally, I was delighted to see the three articles on Ample and hope you will publish more in due course.

Leslie Gardner

PC TO BEEB

There is an increasing trend for firms who commission work to ask for it on disc as well as print-out, invariably in PC compatible format. BEEBUG has published programs to enable a PC compatible to read BBC files and vice versa, but if you don't own a PC compatible (or perhaps an Archimedes) you still have to send your disc away to a bureau to have it transcribed, which costs time and money.

Now my Master Compact (and presumably Master 128) allows you to change from ADFS to DFS and vice versa. Would it not be possible for someone to implement MS-DOS so that it could be selected and used on a BBC micro in the same way. Anyone who devises such a program would earn my undying gratitude, as well as my pennies.

Ruben Hadekel

To emulate MS-DOS in its entirety on a BBC micro is probably not feasible - the machine just doesn't have the power or the capacity to do the job properly. On an Archimedes, Acorn's PC emulator functions very well, but the Archimedes is a much more powerful machine. The only equivalent solution (for a Master 128 but not a Compact) is to install a 512 co-processor. Sufficient power is then available and the result is a very satisfactory MS-DOS machine. The 512 is no longer made, but examples are often advertised in the small ads in BEEBUG.

The other alternative is to use file transfer programs running on a Beeb (which must be fitted with the 1770 (or later) disc controller chip, fitted as standard on a Master 128 and Master Compact (1772 disc upgrade kit £53.53 inc. VAT from BEEBUG, product code 0217C). The programs Mr.Hadekel refers to were published in BEEBUG Vol.6 No.10 & Vol.7 No.1. The only requirement is that the PC format discs used be pre-formatted on a PC - maybe a friendly PC dealer might help.

SIMPLE CYCLES

I refer to the First Course article in Vol.8 No.10 on making a variable cycle through a limited range of values. Where two values are involved the simplest solution is to use the EOR operator. For example, suppose you want r% to oscillate between 27 and 59. Type:

```
PRINT 27 EOR 59
```

giving 32. If r% is initialised to either 27 or 59, the statement:

```
r%=r% EOR 32
```

will switch r% to the other value.

In addition, the example in the article of counting from 1 to 5 can most simply be written as:

```
r%=r% MOD 5 + 1
```

R.C.Smith

The use of EOR in this context is one I had overlooked. I must also thank Mr.Smith, and several other readers, for providing the much better (second) example given above for counting from 1 to 5. B

Personal Ads

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

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

Still nearly 5 pages of BBC hardware, ROMS, books, games, music, utilities and programs left! Includes Morley 2Mb RAM disc and Advanced Teletext adaptor. Most prices between 50p & £5! Must sell - need shelf space. Tel. 091-529 4788 anytime or send SAE for list to; 26 Newark Drive, Whitburn, Sunderland, Tyne & Wear, SR6 7DF.

Master 512/1024 (Solidisk PC+) with dual 40/80 D/S D/D, Zenith mono monitor, View etc. cartridges; Master ROM, WW+, CP, Wordease; MFII; I-Word, I-Sheet, I-Base, DOS+, 2.1; GEM etc.; Dabs M512 Guide, Disc, & Shareware. See working £500 o.n.o. Also BBC B with Torch Z80 disc pack (dual D/S 40/80). BBC MOS: extra RAM and ROM box, Dataplus, Vufile, etc. CPM: Perfect Writer, Speller, Calc, Filer, BBCBASIC Z80 £350 o.n.o. Tel. 021-308 0224.

WANTED: "Introducing CP/M on the BBC Micro Z80 Second processor" by Bruce Smith. Reasonable price paid. Tel. 081-556 5154.

BBC B+ with ATPL RAM/ROM board fitted, Microvitec 1451 colour monitor, Cumana twin 40/80T disc drive, Star dot-matrix printer, AMX mouse, Acorn teletext adaptor, Pace modem and music magazines £450. Tel. (0462) 685238.

Aries B32 Shadow RAM card (with manual) £40 inc. P&P. Tel. (0305) 772817.

WANTED: "Advanced Machine Code Techniques for the BBC Micro". by AP Stephenson and D J Stephenson. Tel. 057- 37 321.

WANTED: Viglen Master console unit with fittings. Tel. (0727) 57045.

For Sale: Juki 6100 daisywheel printer with Centronics interface, complete with five daisywheels and several ss ribbons + BBC B cable if required £170. Tel. 031-449 3869.

M128, twin Pace 40/80 drives, Ref Guides 1&2 £480, Panasonic KXP1081 £75, Comstar + Nightingale Modem £65, Acorn A-Teletext £40, Watford Quest Mouse, Mouse & Conquest £50, Interword £25, Interchart £15, Island Music £20, Printmaster ROM, PMS Multifont plus various ROMs, Viglen cartridge system, 50 blank discs £750 for the lot. Tel. (0457) 852381 after 6pm.

M128 twin 40/80 drive, Cub 452 colour monitor, 2 Quad care packages, Pagemaker, Superart and mouse. EPROM programmer with Interword, Master ROM, Advanced disc toolkit Printmaster ROMs, Reference manuals 1&2, lots of software and magazines £650 o.n.o. Fleet Street Editor plus fonts, graphics £25. Tel. (0475) 2332 anytime.

For Sale: Microvitec colour monitor for BBC and Spectrum £100 o.n.o. Telebox 2, converts colour monitor into colour television set with sound £60 o.n.o. Cumana SSSD, 100k disc drive, own PSU, hardly used £40. BBC compatible remote control tape recorder £10. Software; System Delta Mailshot application, Mini Office, Mini Office Master, Some books. Brother HR15 daisy wheel printer, parallel and serial connections, friction and cut-sheet feeder; several daisy wheels, £350, Star NL10, 9 pin dot matrix printer, friction, tractor and cut-sheet feeder, hardly used £200. Tel. (0707) 50568.

Hegotron Robotics Grafpad II for the Master 128 £30, Archimedes 310, RISC OS, Entry level, Watford 4 way backplane, fan fitted, single floppy, PC Emulator £595 (Upgraded to 440). Tel. (0734) 771230 day, (0734) 784897 eves.

Books; Master 512 User Guide (Dabs), disc & book £10, View: A Dabhand Guide £7, Advanced Machine Code Techniques £6, Assembly Programming made easy £5, Econet Advanced User Guide £5, Advanced Disc User Guide £8. **Discs;** P.I.A.S. 4 £4, Galaforce, Icarus (Master version) £3 each or both for £5, Repton Infinity £7, The Real You (IQ and personality tests) £7, 6 issues of Disc User £10, **Tapes;** Star Seeker, Quest, Life of Repton, Acornsoft Hits II, Rik the Roadie £5, Speed Read course £5. ROMs; View Professional £35. Acorn Teletext adaptor with ATS £50, Care Master Smart cartridge £15, Care Master Quad cartridge £8, Delta 14b joystick £7, Master dust cover £2, Dumb terminal (mono screen, keyboard, RS232) £20, Master 512, Acorn amber monitor, dual 5.25" 3.5" drives £650.

Epson Printer MX-80 F/T III in good condition in original packing £70, Spellcheck III ROM + dictionary disc £10, Watford ROM manager ROM + manual £10, Mini Office II + manual £8, Watford DFS (for 62 files) £10, Prism Modem 1000 + ROM £20, Accoustic Modem for BBC B £5, all in working order and mostly in original packing cases with manuals. Tel. (0993) 771 341 (Sun/Wed) or 071-902 8612 (Thurs/Sat) eves.

M128, Cumana dual drive, Modem, Mouse, PMS Genie, Care Smart Cartridge, Beebug Master ROM, AMX Design ROM, Rom cartridges, joystick, Data recorder, Plinth, Ref. manuals 1&2, 50 discs with box. £500. Tel. 071 639 0487.

BBC B issue 4 fitted Aries B32, Instant Mini Office II, Opus DDOS, Opus Challenger disc drive 512k RAM disc. Twin Voltmace joysticks (unused). Also Kisho computer compatible tape recorder and tapes. 12" mono TV £350. Various leads, User Guides, books, 65 copies "Micro User", 49 copies "BEEBUG" blank 5.25" discs etc. Typing Tutor, Family History £50, will split but higher prices. Tel. (0225) 310083 anytime.

BBC Software/Hardware for sale. For a complete list, send SAE to: David Sayed, 32 Warwick Road, London W5 3XJ.

BBC Master 128, good condition, hardly used, includes View word processor and spreadsheet etc. £310. Tel. (0223) 321128.

M128 software: Stop Press (3 discs) £20, Ref. manual pt.1 for M128 £7,

Quad cartridge £7, Acornsoft Logo ROM £19, Dumpmaster ROM £15, Strykers Run £3.50, Elixir £3.50, Repton Infinity £7, Revs £4, MicroUser Arcade Game Creator disc £3, all items as new. Tel. (0600) 5280.

WANTED: ACP/PRES 256k AQR Cartridge - Write to; J Duncan, 41 Curriehill Castle Drive, Balerno, Midlothian EH4 5TA stating price.

Swap Intersheet, boxed as new, for Interword in similar condition. Tel. (0992) 711788.

BBC B 32K, Torch Graduate (dual disc 8086 processor) c/w Psion software, mono monitor. Offers please. Tel. (0235) 868765.

Arc software: Archway £40, Repton 3 £7, E-Type £8. Compact software: White Knight Chess £4, The Superior collection vol. 2 £4. ACP 1770 DFS for Master or Compact £12. Tel. 081-986 4442.

512 co-processor with mouse and GEM software £80, Interword ROM £30, Viewstore ROM £25, Viewplot £10, Printer Driver Generator, Office Master and Office Mate £5 each. All in excellent condition and complete with manuals. Tel. (06285) 20320 eves and w/e.

M128, Cumana dual drive, Modem, Mouse, PMS Genie, Care Smart Cartridge, BEEBUG Master ROM, AMX Design ROM, ROM Cartridges, joystick, data recorder, Plinth, Ref. manuals 1&2, 50 discs with box £500. Tel. 071-639 0487 (from 1/9/90).

Viewspell ROM, boxed complete with manual, as new £25. Also Hi View, boxed complete with manual £25. Tel. 086-732 8776.

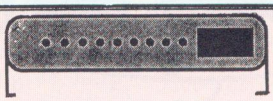
BBC Master 128 + Viglen cases, 512 2nd processor, Microvitec Cub colour monitor, 2 x 80T Watford disc drives, Canon PW1080A printer + spare ribbons, Nidd Valley Mouse, Voltmace joystick, Vine Micros ROM extension board, Morley ROM extension board, BEEBUG 'C', Computer Concepts MegaROM. (Interword /Intersheet/Interchart), Interbase, Wordwise Plus II, Publisher ROM, + many utilities/games software. £650 the lot! Tel. 081-997 0397.

M128, Turbo co-processor, two CS400S Cumana drives, GIS Teletext adaptor, AMX mouse, Superart, Pagemaker, Viewstore, Interword, Wordwise plus, Microprolog, Logotron Logo, and other firmware. BEEBUG bound vols. 2-8, Ref. manuals 1&2 with many other books and discs £500 o.n.o. Tel. (0272) 711733.

Master 512/1024 (Solidisk PC+) with dual 80T D/S D/D, 2 joysticks, view etc., cartridge with Printmaster ROM, Dabs M512 Guide, Master Ref. manuals, BEEBUG magazines Vol. 1 No. 9 to present, TCS 512 Mouse driver, Mosaic Twin advanced PC Lotus compatible spreadsheet package, considerable amount of BBC & PC software £550. Acorn 30Mb Winchester drive for Master £300, also BBC teletext adaptor & TFS, £50 o.n.o. Tel. (0905) 67488 eves.

BBC Master + 512 digitiser & all the best Educational software from 10 year collection (legal copies) will split. Tel. 081-674 5615.

BBC B Watford DFS, 16k Sideways RAM, 40T S/S disc drive, monitor plinth, large amount of software inc. Exile, Elite, Revs, Aviator, Wordwise Plus CC Graphics ROM and Disc Doctor, also external speaker, many magazines (BEEBUG, Micro User, Acorn User) and reference books £300 o.n.o. the lot, may split. Studio 8 with 5 octave keyboard £50 o.n.o. Tel. (0272) 564654.



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

BEEBUG all issues from Vol.2 to date. All mint condition £1 each including p&p, also other bits including viewstore ROM, A.U.G., new A.U.G., lots of other books, ring for lists. Tel. (0737) 556384.

WANTED: Mode 7 Foreign Language Teletext Character Generator integrated circuits SAA5053 (Italian), SAA5054 (Belgian), SAA5057 (Cyrillic). Purchase or hire. Tel. 024024 2423 eves.

Watford DP35 800S double 5.25" and 3.5" disc drive unit £150. Tel. (0707) 54311.

Pace Nightingale modem £40, Commstar II £15, (modem + ROM £50), SPY2 £15, Disc Doctor £10, Dotprint Plus £10, all original ROMs with full documentation. Tel. (0900) 825503.

Opus Challenger 1Mb 5.25" disc drive with RAM disc and very advanced DDOS. Includes REPLAY hardware based tape to disc utility, and bundles of software £119 o.n.o. Or I will do a partial swap for a simpler drive for my Archimedes. Tel. 081-560 7310.

Interword £25, Spellmaster £30, Interbase £35, JUKIT (Juki daisy wheel printer toolkit) £10, all original, boxed with manuals (upgraded to Archimedes). Tel. (0742) 342870.

Back issues of Micro User 1987, 1988, 1989 including binders, back issues of Acorn User 1988, 1989 also with binders, ideal for education, reference. Offers please. Last Ninja (karate game), 3D Pool, both on 5.25" discs £5 each + some odds & sods. Please phone for details (0326) 240734 after 6pm.

M512 co-processor with mouse, DOS plus 2.1, GEM, manual, Shareware collection, Dabhand User Guide and disc, Mastering DOS Plus £130. Advanced Disc Toolkit ROM £10, Hyperdriver ROM £10, Viewstore ROM £15, Viewspell ROM £10, Viewindex £5, Viewchart £2, Acorn ISO Pascal £20, MOS Plus ROM £10 - all with manuals in original packing. BEEBUG - all issues to date £25, M128 Reference manuals 1&2 £5 each, many other books. Tel. (0742) 342321.

Brother HR-15 daisy printer with 3 wheels + ribbons boxed £320, Acorn Master cartridges £6 each, Original double Acorn joysticks £11, Voltmace Datapad 16 £18, Voltmace Delta 14 joystick + keypad £12, Micropulse External 8 socket switchable ROM box £30, ACP Toolkit ROM £20, new Master Ref. manuals 1&2 £10 each. Tel. 081-989 2666.

Chaotic Computing BBS wants new users! Call us on (0943) 817010 V23, 1200/75 baud only, 8N1 scrolling, 24 hours/day. We have many interesting text files and message areas, and are local from Leeds & Bradford.

WANTED: Bridge unit for Master 128 must be one 5.25" drive and one 3.5" drive (disc drives not required). Tel. 071-494 1365 office hours only.

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

Watford DDFS complete, plus Disc Filing System manual £30. Watford 32K Shadow RAM/Printer Buffer board £35. Contact Alan Wrigley at BEEBUG. Tel. (0727) 40303.

ADMISSION: ADULTS £4.00 – UNDER 16's £3.00

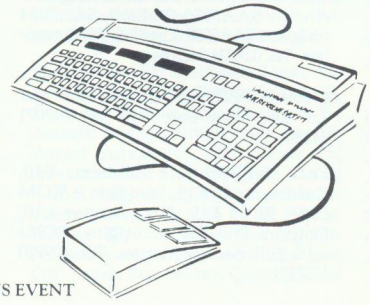
WESTMINSTER EXHIBITION CENTRE
(New Horticultural Hall)
Elverton Street
London, SW1

Friday 7th Sept – 12 noon to 7.00 p.m.
Saturday 8th Sept – 10.00 a.m. to 6.00 p.m.
Sunday 9th Sept – 10.00 a.m. to 6.00 p.m.

Nearest Undergrounds – St James's Park, Pimlico or Victoria & B.R.

THIS UNIQUE SHOW DEDICATED TO THE BBC ACORN FAMILY OF MICROCOMPUTERS FEATURES:

- **In excess of 60 exhibitors representing a vast range of products and services covering all BBC ACORN micro's.**
- Daily Seminar/Workshop sessions covering subjects ranging from Education to Desk Top Publishing. A charge of £1.00 per session is made and all proceeds go to Charity. Tickets are only available at the Show.
- **FREE daily prize draws.**
- Meet the staff of BBC ACORN User Magazine who will be on hand to give you technical advice.
- **Special feature stand by ACORN COMPUTERS.**
- On-Going Schools Desk-Top Publishing and Music project during Show.
- **FREE courtesy coach from Victoria Station. Take the Wilton Road exit and look out for the Guide.**



A SAFESELL EXHIBITIONS EVENT

In association with
BBC ACORN USER MAGAZINE

For more information phone Safesell Exhibitions on 0737 814084

FREE ROMS

Colin Price

The Master 128 Welcome disc currently contains a number of 'free' goodies. These include updated Dircopy and Copyfiles, but there are also two ROM images. The first is DFS2.9 which solves the *CLOSE bug (and other problems) - see also BEEBUG Vol.8 No.6 p.42 & Vol.8 No.10 p.46.

The second is Spriter, the missing part of the GXR ROM containing the Acorn Sprite Editor and plotting routines as well as on-screen help for PLOT codes. Either or both of these ROM images may be loaded into vacant sideways RAM slots by using the *DRLOAD command. Then press Ctrl-Break to re-initialise the system. Using *HELP <name> should give help information on the new ROMs.

ASSEMBLER MACROS

Andrew Rowland

Bernard Hill's article on assembler macros (see Practical Assembler, in BEEBUG Vol.9 No.2) highlights a useful technique. Because a macro repeats similar code each time it is called, it is sometimes better to use a subroutine. The example on printing strings, while entirely suitable for explaining the methods, can be replaced by:

```
JSR prstring
EQUS "any string"
NOP
```

This uses the fact that when a subroutine is called with JSR, a return address is pushed onto the stack. This address points to the byte immediately following the JSR, so can be used to print a string, incrementing the address as you go. This address then becomes the new return address, pointing to the code following the string. The string should be terminated by a suitable byte which isn't needed in the string itself - I use NOP (&EA).

A similar trick can be used to generate errors in service ROMs - JSR error below.

```
 10 REM HintBAS
 20:
100 zp=&70:osasci=&FFE3
110 FOR pass=0 TO 3 STEP 3
120 P%=&900:[OPT pass
130 JSR prstring
140 EQUB 13:EQUS "by A.C.Rowland":NOP
150 JSR error
160 EQUB 17:EQUS "Escape":BRK
170:
180 .prstring PLA:STA zp:PLA:STA zp+1
190           LDY #0
200 .prstrlp  JSR inczp:LDA (zp),Y
210           CMP #&EA \ NOP op code
220           BEQ prstro:JSR osasci
230           JMP prstrlp
240 .prstro   JSR inczp:JMP (zp)
250:
260 .inczp    INC zp:BNE incout
270           INC zp+1
280 .incout   RTS
290:
300 .error    PLA:STA zp:PLA:STA zp+1
310           LDY #0:STY &100
320 .errlp    INY:LDA (zp),Y
330           STA &100,Y:BNE errlp
340 .errout   JMP &100
350 ]NEXT:CALL &900
```

IMPROVED MOVE-DOWN ROUTINE

F.G.Beach

The Move-Down routine by Alan Wrigley (see Hints & Tips, BEEBUG Vol.8 No.8) can be improved further as shown below. Lines 0 to 4 can be used to display any instructions or text about the program to follow. Line 5 keeps the information displayed until Return is pressed. Line 6 is the download routine itself. Line 7 programs function key f1 to delete lines 0 to 9 (or whichever lines are to be deleted) and start the program running, while line 8 simulates the pressing of f1 to put this process into effect.

```
 0 CLS:PRINTTAB(0,4)"Program Name"
 1 PRINT'"Author etc"
 2 PRINT7"More information"
 3 ........................
 4 ........................
 5 PRINT'"Press Return to move down and
   start":INPUT A$;:PRINT'Program now
      being moved down"
 6 *KEY0 *T.|MFORA%=0TO(TOP-PAGE)STEP4:
   A%!&E00=A%!PAGE:NEXT|MPAGE=&E00|
      MOLD|MRUN|M
 7 *KEY1 DELETE 0,9|M*FX138,0,128|M
 8 *FX138,0,129
 9 END:REM Movedown routine to E00
```

BEEBUG MEMBERSHIP

Send applications for membership renewals, membership queries and orders for back issues to the address below. All membership fees, including overseas, should be in pounds sterling drawn (for cheques) on a UK bank. Members may also subscribe to RISC User at a special reduced rate.

BEEBUG SUBSCRIPTION RATES

		BEEBUG & RISC USER
£16.90	1 year (10 issues) UK, BFPO, Ch.I	£25.00
£24.00	Rest of Europe & Eire	£36.00
£29.00	Middle East	£43.00
£31.00	Americas & Africa	£46.00
£34.00	Elsewhere	£51.00

BACK ISSUE PRICES (per issue)

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

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

POST AND PACKING

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

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

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

BEEBUG MAGAZINE is produced by BEEBUG Ltd.

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

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

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

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

In all communication, please quote your membership number.

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

Magazine Disc

August/September 1990
DISC CONTENTS

CONSPICUOUS CONSUMPTION - this is the first of a two part program on monitoring and charting your car's fuel consumption. Both model B and Master versions are provided.

PRACTICAL ASSEMBLER (Part 4) - two demonstration programs from this month's article on 6502 machine code programming.

MONIX: A MACHINE CODE MONITOR - this fully working program (to be augmented further next month) allows you to examine the contents of memory at any desired location.

ADFS DIRECTORY EXAMINER AND COMMAND FILE GENERATOR - this program will readily reveal the structure of any ADFS formatted disc, and if required, will create command files for the automatic loading and merging of procedures and functions.

QUAD GAME - an excellent implementation of Tetris, an addictive yet frustrating game as you try to control the moving shapes.

BEEBUG WORKSHOP: PERMUTATIONS AND THE TRAVELLING SALESMAN PROBLEM - a fresh start to our Workshop series investigates some searching problems, and in turn sets a few problems for the reader.

FIRST COURSE: SCROLLING TEXT DISPLAYS - a complete working demonstration of all the text scrolling techniques described in this month's article.

LORENZ ATTRACTOR - a stimulating program demonstrating this classic phenomenon from the modern science of chaos.

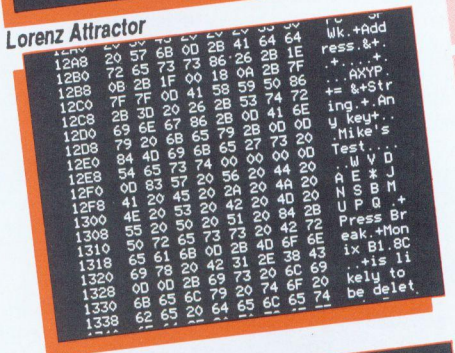
THANKS FOR THE MEMORY: BAS128 - four more programs to help users of Acorn's Bas128 version of Basic, including a LISTIF substitute (as an EXEC file), a link to Edit, a ROM header, and code to enable 65C12 mnemonics.

MAGSCAN DATA - bibliography for this issue (Vol.9 No.4).

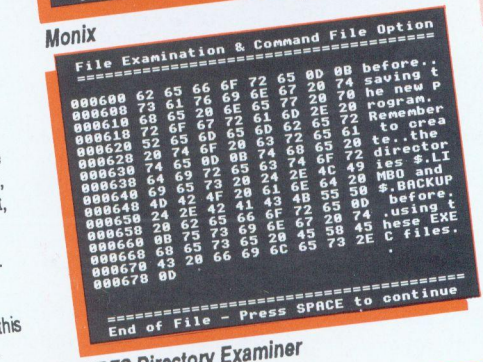
BONUS ITEM

NEWTON'S CRADLE - an impressive implementation of this by now classic animation, but this time for the BBC micro.

Lorenz Attractor



Monix



ADFS Directory Examiner

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

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

DISC (5" or 3.5") SUBSCRIPTION RATES

	UK ONLY	OVERSEAS
6 months (5 issues)	£25.50	£30.00
12 months (10 issues)	£50.00	£56.00

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

All subscriptions and individual orders to:

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

BEEBUG The Archimedes Specialist BEEBUG

SPECIAL SUMMER PROMOTION

A3000 AND LEARNING CURVE

FREE 1 Mb RAM Upgrade

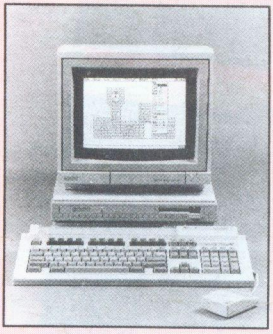
Acorn's new Learning Curve Package consists of an A3000 computer, First Word Plus V2 (the number one Word Processor package for the Arch), the PC Emulator and Genesis (A graphic based database system with a number of sample files). A parent's guide to the national curriculum and a demonstration video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

If you purchase an A3000 or Learning Curve from Beebug, in addition to the backup service for which we are renowned, we will supply free of charge an extra 1Mb RAM board, increasing the computers memory to a powerful 2Mb.

Learning Curve (no monitor)	699.00 (803.85 inc. vat)
Learning Curve (Acorn Monitor)	899.00 (1033.85 inc. vat)
A3000 (no monitor)	599.00 (688.85 inc. vat)
A3000 (Acorn Colour Monitor)	799.00 (918.85 inc. vat)

Colour Monitors

All colour systems sold by BEEBUG use the excellent Acorn colour monitor. When comparing prices we recommend that you insist on this monitor.



THE ARCHIMEDES 400/1 SERIES

For a limited period we are offering a number of unbeatable deals on the 400/1 series computer. We would particularly draw your attention to the offers on the Acorn A440/1 systems. These are genuine A440/1 computers from Acorn on offer for a short period only.

A410/1 With Free 20Mb Hard Drive £1099.00 (£1263.85 inc. VAT)

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

A Genuine A440/1 For Only £1499 (£1723.85 inc. VAT)

For a short period only, we are able to offer the *genuine* Acorn Archimedes A440/1 for only £1499. Normal RRP £2099. Supplied with an Acorn colour monitor for £1599 (£1838.85 inc. VAT). *This is a **saving of £700** from list price.*

A440/1, Multisync & Star XB24-10 Printer £2099 (£2413.85 inc. VAT)

A *genuine* Archimedes 440/1 with a 50Mb Hard drive (28ms access time), 4Mb of RAM, a Samsung CT4581 Colour Multisync Monitor and a STAR XB24-10 printer, for only £2099 (2413.85 inc VAT). *This represents a saving of **£1000** from list price.*

Trade In Your Old System

We are always pleased to accept your existing Acorn computer equipment in part exchange for a new system.

Please call for a quotation.

Please add £230 (inc. vat) for an Acorn colour monitor or £ r a Taxan 775+ Colour Multi-Sy equired..

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

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