

Vol.8 No.10 April 1990

49 53 55

REVIEWS

DEC		
EATURES	6	Re
Veaving Patterns Upgrading to an Archimedes (Part 2) ADFS Desktop Curve Fitting (Part 1) A Datafile Search Utility	8 13 17 25	F
Music Programming III / Map (Part 1)	28	•
First Course - Improving Basic Programs, Sideways and Shadow RAM EdiKit (Part 4) 512 Forum	eeb!)	33 37 43 46
Improve Your Master (and Workshop - Writing a Compiler (Part 5 Elegant Entry to Wordwise Order Out Of Chaos (Part 2) Plus 2)	49 5

1

Receiving Information by Radio 22 Ausic To Your Ears AD ITEMS	
REGULANTIE	4
Editor's Jottings	5
News	57
Hints and Tips	58
RISC User	59
Postbag	60
Personal Ads	62
Subscriptions & Back Issues	63
B Magazine Disc/Casselle	

HINTS & TIPS tions For 1000

Deno Zero Memory Locations For
Page Zelo Basic IV
Control III Date DES Catalogues
Invisible Dro ection
Program Flotoono
Memory Wipe
Autobooting Woldweer Of The Screen
Printing In The Corner of

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 1.

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



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 cassettebased system.



Program needs at least one bank of sideways RAM.



Program is for Master 128 and Compact only.

Editor's Jottings

NEW BEEBUG OPEN DAY

A third BEEBUG Open Day is being organised for Sunday 22nd April with displays and demonstrations from a variety of software houses, plus BEEBUG staff to help and advise you on any purchases you may wish to make. Magazine staff will also be there to hear your views and talk about the Acorn world. Full details are being circulated with this issue of BEEBUG. Note the date now we very much hope to see you then. With the changing Acorn scene, it seems reasonable to consider at this time, whether the Workshops should continue, and if so in what form and with what aim. We have no immediate plans to terminate this series at present, but your views and suggestions with regard to the future of the Workshop series are most welcome.

VOLUME 8 INDEX AND BIBLIOGRAPHY

As is our custom at this time of year we shall shortly be

SHOWS

As reported on this month's News page, this year's BBC Acorn User Show is scheduled for September, and the Computer Shopper Show for December. Acorn has confirmed that it will exhibit at both shows, and BEEBUG will also have a stand at both.

Acorn's attendance is certainly good news for all Acorn fans. Our view has always been that Acorn should make a point of attending all shows which are specific to its machines. It encourages other third party suppliers to attend as well, and customers feel that they have a

right to expect the support of Acorn at such events, and have the opportunity to meet the company and express their views, if they wish, face to face. Let's hope that Acorn will also have some exciting new releases to show off as well.

BEEBUG WORKSHOPS

The BEEBUG magazine Workshops have been running for six years, and in that time have covered a variety of topics aimed principally at the more experienced (mainly Basic) programmer. This issue marks the conclusion of the current series of Workshops on compilers, and the end of two year stint at the helm of this series by David Spencer, our former Technical Editor.

Editorial Extra

New System from Acorn

As we go to press we learn that Acorn will be releasing a packaged A3000 system called The Learning Curve on 2nd April aimed at the home education market. The package comprises an A3000 plus a new version of the PC emulator, Software Solutions' Genesis information handling application, 1st Word Plus, parents' guide to the National Curriculum, and a supporting video, all for £695 plus VAT (that's just £50 more than a standard A3000). Details are available from BEEBUG. compiling and printing a complete index to the whole of volume 8 of BEEBUG magazine. This will be distributed free to all current subscribers with the May issue of the magazine, the start of volume 9.

We shall also be updating our Magscan discs so that they too are complete to the end of volume 8. For convenience the Magscan update always contains a complete set of bibliography files from volume 1 to the latest complete volume.

Magscan is the computerised bibliography to BEEBUG magazine, allowing quick and easy reference to any previously published item by use of keywords. All Magscan files are also compatible with ArcScan, our Archimedesbased bibliography system (and likewise ArcScan files will work with Magscan).

The full Magscan package, including indexes to the end of volume 8, costs £12.50 plus £1.00 p&p; the update costs £4.75 plus 60p p&p. Please state whether you require 40T or 80T versions when ordering, and be prepared to quote the date of original purchase if you just require the update.

News News News News News News

NEW MD FOR ACORN

Harvey Coleman has left his position as Managing Director of Acorn Computers Ltd, to become Chief Executive Olivetti Systems and Networks Canada Ltd. His position as MD will be taken by Sam Wauchope, previously Acorn's Sales and Marketing Director. Mr Wauchope will become Group Managing Director after the company's AGM in April. In turn, Michael O'Riordan, until recently Group Marketing Director with ITL, has been appointed as Sales and Marketing Director of Acorn.

SLOGGER RE-EMERGES

Just as we all thought that Slogger Computers had disappeared without trace, the company has reappeared, and with a new product for the Master 128. This is a 'smart' cartridge which will do a lot more than just accommodate ROMs. No further details are yet available but we hope to have a review in the May issue of BEEBUG. For those who wish to contact Slogger the company can be located at Sancreed Business Centre, Sancreed, Newbridge, Penzance, Cornwall TR20 8QP, tel. (0736) 810920.

BBC SOFT MERGES WITH LONGMAN/LOGOTRON

It became known last year that BBC Soft, the software publishing unit within BBC Enterprises, was being disbanded. Now comes news that the BBC list of titles will be taken over by the Longman/Logotron stable to form a major software publishing house in the UK specialising in educational software. The new joint venture will be based in Cambridge and managed by Longman/Logotron. Longman/Logotron are at Dales Brewery, Cambridge CB1 2LJ, tel. (0223) 323656.

SUPERIOR MAY NOW BECOME SUPERIEUR

Latest games release from the indefatigable games software house, Superior Software is *Perplexity*, claimed to combine the best puzzling features of Repton with superb 3D graphics. The end result looks rather like a 3D Pacman, but addicted games players will no doubt find this as challenging as ever. The game costs £9.95 on cassette, £11.25 on 5.25" disc, and £14.95 on 3.5" disc for the Master Compact.

It is also reported that Superior has signed a deal with French publisher Infogrammes and is negotiating for the rights to *Hostages* and and *Sin City*. First releases from this venture are expected about Easter time.

Superior Software is at P.O.Box 6, Brigg, South Humberside DN20 9NH, tel. (0652) 57807.

SHOWS UPDATE

Latest news on the shows front is that Acorn will have a significant presence at the *Which? Computer Show* to be held at the NEC, Birmingham from 24th to 27th April. The Acorn stand will encompass a range of third party suppliers offering Archimedes based products.

This year's *BBC Acorn User Show* will be at the New Horticultural Hall, Westminster from 7th to 9th September. Again, Acorn will have a major stand at this show, and BEEBUG will also be there on stand 74. This is the only show dedicated to Acorn computers, and thus the one show that no BBC micro user will want to miss.

Finally, the *Computer Shopper Show*, on which we reported in Vol.8 No.8 will be from 6th to 9th December (one day longer), at the Wembley Conference Centre. Acorn has confirmed that it has already booked a stand for this show, and BEEBUG will be there as well on stand J25.

COMPILING TECHNIQUES

If our recent Workshop series on compiler writing has taken your fancy, then a new book from McGraw-Hill may be of interest. *Introduction to Compiling Techniques* by Dr.J.Bennett is priced at £12.95 and should be available through all good bookshops (ISBN:0 07 707215 4).

YES MARKETING

This is the name of a young enterprise company set up by Queen Elizabeth's Hospital School of Bristol. The first product is a game called *Balls!* which retails for just £5 inc. p&p. It is available from YES Marketing, QEH, Berkeley Place, Bristol BS8 1JX.

Weaving Patterns

David Gill introduces a fascinating application for your micro with this short program.

The program WEAVE was designed for use in schools to simulate weaving patterns, but has been found to appeal to all ages. Each pattern involves different ways of interleaving vertical and horizontal threads as would occur on a loom.



Figure 1

Type the program in, and save it away before trying it out. When run, the program asks the

Escape will return you to the initial dialogue screen, while pressing Shift-Escape will terminate the program completely.

With a little experimenting a wide variety of interesting patterns can be achieved.

PROGRAM NOTES

The high resolution graphics of mode 1 are used to make a lifelike representation of weaving by using two procedures. One of these represents a vertical thread over the horizontal (PROCv), and the other a horizontal thread over the vertical (PROCh). Each procedure uses 8 pixels in a mixture of two colours with the top left-hand pixel in background black to make a block of 9 (see Figure 1). A check pattern of four

vertical threads and six horizontal threads is shown in Figure 2.



user to decide how many vertical and horizontal threads make up a pattern repeat. The simplest *plain* weave requires 2 vertical and 2 horizontal threads. A herringbone pattern needs 10 vertical and 4 horizontal threads.

The number of horizontal threads has to be even and the program allows a maximum of 20 vertical and 10 horizontal threads. A plain weave uses a single colour warp, but a check pattern alternates warp colours determined by the horizontal repeat.

The colours of the threads can be changed whilst weaving proceeds, by pressing 'V' or 'H', and the weaving process can be temporarily halted by pressing 'S', and any key subsequently to continue. At any time, pressing

Figure 2

Two further procedures (PROCI and PROCr) give the left and right selvedges to turn the horizontal *weft* thread at the edges of the sample cloth. PROCwarp sets the vertical *warp* threads at the beginning of a run, and these threads are overwritten by PROCI and PROCr.

Although no screen dump (for hard copy) has been included in this listing, a suitable call could be be inserted into PROCend with little difficulty. Using a black and white printer, you will get the best result by changing the colours

Weaving Patterns

back to give red and white threads on a black background.



A typical 'plain' weave

10 REM Program Weave 20 REM Version B1.2 30 REM Author David Gill 40 REM BEEBUG April 1990 50 REM Program subject to copyright 60 : 100 MODE1:DIM P\$(20,10) 110 VDU23, 1, 0, 0; 0; 0; 0; 0; 120 ON ERROR GOTO380 130 REPEAT 140 REPEAT: PROCsetup: UNTIL Q\$<>"R" 150 PLOT4, 100, 50: PROCwarp 160 PLOT4, 100, 100 170 REPEAT: Y%=Y%+0%*12 180 FOR Q%=0 TO S% 190 r%=1 200 FOR T%=1 TO U% 210 FOR P%=0 TO R% 220 IF P\$(P%,Q%)="H" PROCh ELSE PROCV 230 PLOT0, 12, 0:PROCcol 240 NEXT P%:IF Q\$="C" r%=(r%+2)MOD4 250 NEXT T%:Q%=Q%+1:PROCsr 260 IF Q\$="C" THEN r%=(r%+2)MOD4 270 FOR T%=1 TO U% 280 FOR P%=R% TO 0 STEP-1 290 IF P\$(P%,O%)="H" PROCh ELSE PROCV 300 PLOTO, -12, 0: PROCcol 310 NEXT P%:IF Q\$="C" r%=(r%+2)MOD4 320 NEXT T%:PROCsl 330 NEXT Q%:IF Q\$="C" w%=(w%+2) MOD4 340 UNTIL Y%>800 350 UNTIL FNend: MODE7 360 END

380 IF ERR=17 AND NOT INKEY-1 GOTO 120 390 MODE7: IF ERR<>17 REPORT: PRINT" at line ";ERL 400 END 410 : 1000 DEF PROCV 1010 GCOL0, 0:PLOT1, 0, -4 1020 GCOL0, w%: PLOT1, 0, -4: PLOT0, 4, 8 1030 GCOL0, r%: PLOT1, 0, -4: PLOT1, 0, -4 1040 PLOT1, 4, 0: PLOT1, 0, 4: PLOT1, 0, 4 1050 PLOT0, -8,0 1060 ENDPROC 1070 : 1080 DEF PROCh 1090 GCOL0, 0:PLOT1, 4,0 1100 GCOL0, r%: PLOT1, 4, 0: PLOT0, -8, -4 1110 GCOLO, w%: PLOT1, 4, 0: PLOT1, 4, 0 1120 PLOT1, -8, -4: PLOT1, 4, 0: PLOT1, 4, 0 1130 PLOTO, -8,8 1140 ENDPROC 1160 DEF PROCwarp 1170 PLOT0, 4, 0 1180 FOR T%=1 TO U% 1190 GCOL0, r% 1200 FOR P%=1 TO R%+1 1210 PLOT1, 0, 850: PLOT0, 4, 0 1220 PLOT1, 0, -850: PLOT0, 8,0 1230 NEXT P% 1240 IF O\$="C" THEN r%=(r%+2)MOD4 1250 NEXT T% 1260 PRINTTAB(3,2)"S) top weave. H or V Change colours" 1270 ENDPROC 1280 : 1290 DEF PROCsr 1300 GCOL0, 0: PLOT0, 0, -4 1310 GCOLO, w%: PLOT1, 0, -4: PLOT1, 4, 4 1320 PLOT1, 4, 4: PLOT1, -4, 0: PLOT1, 0, 4 1330 PLOT1, -4, 4:PLOT1, 0, -4:PLOT0, 0, -4 1340 PLOT0, -12, 12 1360 : 1370 DEF PROCsl 1380 PLOT0, 4, 0 1390 GCOL0, w%: PLOT1, 4, -4: PLOT1, 4, -4 1400 PLOT1, 0, 4: PLOT1, -4, 4 1410 PLOT1, 4, 4: PLOT1, 0, 4 1420 PLOT1, -4, -4: PLOT0, -4, -4 1430 PLOT0, 8, 12 1440 ENDPROC 1450 : 1460 DEF PROCsetup Continued on page 12

Mike Williams concludes our investigation of upgrading to an Archimedes.

In last month's BEEBUG I discussed many of the problems facing a BBC micro user who is contemplating an upgrade to an Archimedes or A3000. Initially, at least, one of the questions uppermost in the minds of many is the likely compatibility between a BBC micro system and an Archimedes.

I have already discussed some of the considerations as far as hardware is concerned, so this month we will look first of all at the thorny problem of software compatibility. I shall also consider a case study based on the configuration of a BEEBUG reader to add a touch or realism. We will then finish with some basic factual information on sources of information and suppliers of hardware and software.

SOFTWARE COMPATIBILITY

The Archimedes is based around the ARM processor, which has its own machine code quite different from that used with the 6502 based BBC micro. This is a fundamental difference which means that no 6502 machine code programs will work directly on an Archimedes.

The other major problem is that memory on an Archimedes is organised quite differently from that of a BBC micro. In particular, memory is allocated dynamically on an 'as needs' basis. The consequence is that no program can rely on having access to specific memory locations. This can cause problems with any direct memory access, peeks and pokes for example, or the direct saving and loading of screen displays using *SAVE and *LOAD.

However, the picture is generally a lot better than these remarks may suggest. Many programs written entirely in Basic will run immediately on an Archimedes. This machine uses BBC Basic V, which is an enhanced version of the BBC Basic with which you are already familiar. It also has many new features, matrix operations for example, and many improvements such as the facility to pass arrays as parameters to procedures and functions.

You will also find that any Basic program which does run on an Archimedes performs a lot faster. This can sometimes be an embarrassment, for example with a game which now becomes unplayable, but by and large the increase in performance is quite welcome. As a result many programs, particularly animations, can take on a new lease of life. However, it is impossible to say with certainty whether any given program will run or not.

The main incompatibilities are any use of embedded 6502 assembler or machine code, and any form of direct memory access. Sometimes a little tweaking is all that is necessary to get a program working. If you have any commercially purchased software written in Basic, the best bet is to contact the software house concerned for information and advice.

EMULATION

If you do have machine code programs or other software which seems unlikely to run given my guidelines so far, then do not despair. There are other means open to you in the form of emulators. The Archimedes is supplied with two 6502 emulators both of which in their own way try to re-create the environment of a BBC micro.

The first and most comprehensive of these is called *65Host*. It tries to emulate a complete model B with OS1.2 and Basic II. Once loaded you will find that this does indeed appear to be the case. HIMEM (the upper limit of user memory) appears as &7C00 (in mode 7), or whatever is appropriate for the mode selected. PAGE, however, is set at &1B00, higher than normal because of the ADFS filing system used by the Archimedes. Basic II contains a 6502 assembler, so embedded 6502 machine code should now cause no problems. Neither should

screen saves or loads, or peeks and pokes into the user area of memory. Even star commands specific to the model B or Master will in most cases be interpreted correctly.

This sounds too good to be true, and to some extent it is. Not all programs will work. The most likely culprits are machine code programs which try to access non-existent devices or components of the machine, or call memory locations which have different meanings (for example, direct calls to the Basic ROM). The worst offenders are often games, which frequently do some very nasty (or nifty) things, depending on your point of view, and utilities which try to be too clever by half (BEEBUG's own Sleuth ROM for example).

The other emulator is called *65Tube* which gives a clue to its format. It emulates a 6502 second processor and includes a copy of HiBasic, the version of BBC Basic supplied with the 6502 second processor. This is designed to work with a much larger memory space, and on an Archimedes provides some 44K of user memory. That means that many programs which will not work using 65Host through lack of memory will work with 65Tube.

This emulator essentially translates 6502 machine code into equivalent ARM code. Anything else it just passes on to RISC OS to deal with if it can. This means that RISC OS star commands are recognised and correctly executed. It also means that all the screen modes of the Archimedes are available in a BBC micro environment. In addition, because 65Tube is not trying to emulate a whole micro (software and hardware) it is generally quicker than 65Host. You must remember that any emulator adds an extra layer to the computer with a noticeable slowing down of functions despite the Archimedes very powerful processor.

ROM-BASED SOFTWARE

Many BBC micro owners are likely to have software in ROM: for example, Wordwise, View, Printmaster, Dumpmaster and many others. A common question which arises is whether this software can still be used. As with other software the answer is more a 'maybe' than a definite 'yes' or 'no'.

Beebug April 1990

First of all the Archimedes as supplied has no ROM sockets, so there is nowhere to put any ROMs anyway. However, a ROM podule is available from either Acorn or Computer Concepts, and these will take BBC ROMs. Such ROMs should be used either of the two emulators, but preferably 65Tube, though there is no guarantee that you will meet with success even then. Again the best advice is to contact the originating software house. BEEBUG's Technical Department may be able to offer some assistance, but cannot know every possible ROM.

However, again all is not lost. Much of the most popular ROM software will work: Wordwise, View, and the Inter series for example. The best approach is to obtain a copy of the ROM image on disc and then load this under 65Tube. All of the software referred to above, and other members of the View family, is available in this format for an upgrade price. Master Compact owners are particularly lucky as the View word processor was always supplied as a ROM image on disc for that machine. This can be loaded quite simply on an Archimedes using 65Tube, and if you wish you can then have the luxury of using View in 132 column screen modes.

SOFTWARE SUMMARY

The foregoing may have seemed somewhat rambling, but it is in the nature of software, particularly given the huge variety produced for the BBC micro, that in an article of this nature it is impossible to give precise advice. I have tried to indicate what are the most likely sources of incompatibility for you to assess your own programs. For commercial software I strongly recommend contacting the original software producer.

On the other hand, there seems little point in spending a large sum of money on an Archimedes, and then using it as a faster BBC micro. That would seem a waste. Be prepared to allocate a part of your budget to new software for the Archimedes, and give some thought to how you would spend it. However, the ability to continue using existing software does tide you over as you become more and more oriented to the Archimedes way of life.

Bear in mind, that there are no problems in transferring any text or data files across to the new machine (provided you have the physical means to do this as outlined last month), and such files can often be read into Archimedes based software. For example, spool out the text (unjustified) from Wordwise or Interword, and read the resulting file directly into 1st Word Plus or Pipedream, the two principal competing word processors for the Archimedes. Pipedream even has a special input mode for handling View (and ViewSheet) files.

CASE STUDY

BBC System

BBC model B with Watford DFS, ATPL ROM board with 16K sideways RAM, Watford File-Plus, Wordwise Plus, PMS NTQ, BEEBUG Toolkit, Morley Teletext Adaptor with ATS ROM, Kaga Taxan 810 printer, dual 5.25" drives with own power supply, and Microvitec Cub 1431 colour monitor.



BEEBUG 5.25" Disc Buffer

As far as hardware is concerned both the printer and the dual disc drives described should be usable with any Archimedes, but note that 40 track drives are not supported by the Arc's ADFS, and can only be used with the PC emulator, or DFS emulation. In addition, disc drives which use the Beeb's own power supply pose a problem - one solution may be to buy a BBC micro power supply on its own (about £46). The printer will require a new cable (cost about £9), and the disc drive unit will require a disc buffer (cost about £33) if it is to be connected to an Archimedes.

The colour monitor is definitely not usable, and the Morley Teletext adaptor is also unlikely to function - it would require an I/O podule to provide a bus connection and a ROM podule to take the controlling ROM, hardly a worthwhile investment, particularly now that the BBC has abandoned downloadable software for micros. As you will have worked out for yourself, the sideways ROM/RAM board has no place at all on an Archimedes.

In this instance the reader uses the DFS filing system on his model B. To access 5.25" discs on an Archimedes he will also need either BEEBUG's DFS reader (cost about £10) or Dabs Press' ArcDFS (cost about £30).

The biggest stumbling block may be in the use of that Watford DFS. Discs formatted single density using the standard Acorn 31 file catalogue should pose no problems for transfer, but the use of double density or 62 file extended catalogue can be troublesome. One solution would be to copy any files required to standard format discs before transfer. However, ArcDFS will handle these non-standard formats without difficulty, and the BEEBUG DFS Reader will also handle the 62 file format.

As far as software is concerned Wordwise can be upgraded to a ROM image to work under 65Tube (and now on disc as Wordwise A-Plus to work in native mode), but it would seem a shame to settle permanently for a word processor which makes such little use of the Archimedes capabilities. Neither the PMS NTQ ROM nor the BEEBUG Toolkit is likely to work, and would be better sold with the old system (maybe through the small ads in BEEBUG).

FURTHER CONSIDERATIONS

My comments here are likely to be of a more subjective nature, and you must decide what is important for yourself. I also hope to give a few pointers to some of the better software for the Archimedes.

The Archimedes is a huge step forward from the original BBC micro. It would therefore be a waste, as I have said before, to treat it as no more than a fast BBC micro, but using as far as possible all existing software and data. Despite your misgivings, I would recommend transferring as much as you are likely to want onto your new Archimedes, and then selling the old system as a complete going concern. A good secondhand price for BBC micros can still be obtained, but that won't last for ever. The small ads in BEEBUG do seem to be a very successful way of disposing of unwanted equipment of this kind, or you can trade in your own system with BEEBUG when upgrading.

If you use View (or its associated packages), Wordwise, or any of the Inter family from Computer Concepts, then you may well find the relatively small cost of upgrading to an Archimedes version worthwhile. If you are looking for pure Archimedes alternatives both Pipedream (from Colton Software) and 1st Word Plus (from Acorn) have proved popular, and Pipedream has the added advantage of being a spreadsheet as well. Other spreadsheets include Logistix (from Acorn) and SigmaSheet (from Minerva). Conventional databases are fewer on the ground, but look at System Delta Plus and Multistore from Minerva, and Flying Start II from Mitre Software. New products from Acorn are expected this year as well.



Pipedream showing text, spreadsheet and imported graphics

With its range of colours and graphics capabilities, there is quite a lot of choice in the area of art and graphics (and the Archimedes is supplied with two quite good applications anyway in the form of Draw and Paint). Look

Beebug April 1990

particularly at Artisan and Pro-Artisan from Clares Micro Supplies, Atelier from Minerva, and Art Nouveau from Computer Assisted Learning. For CAD work there are Real Time Solids Modeller from Silicon Vision and Euclid from Ace Computing.

For musical applications consider the offerings of Electromusic Research, and the forthcoming Inspirations from Pandora Software.

This should give a brief indication of what software is available, but there is a whole lot more, and for those who are looking for good entertainment there are already several classic games for the Arc (Zarch, E-Type or Interdictor for example), and plenty more on the way.

There are few books as yet published about the Archimedes, but Archimedes First Steps (£9.95 from Dabs Press) is a good introduction for beginners, and the same publisher's Basic V (also at £9.95) covers all the additional features of Basic V compared with earlier BBC Basics. Note that Acorn's own BBC Basic Guide is not supplied as standard with the A3000 (£19.95 if bought separately). More competent programmers will want to invest in the four volume Programmer's Reference Manual (£79.00) from Acorn and other books from Dabs.

If you are considering an upgrade, then I strongly recommend that you visit a good Acorn dealer (not just somewhere that sells the Archimedes alongside a whole range of other machines). The staff will be able to give you the benefit of their knowledge and experience, and you should also be able to see and try out any software before you buy it. A few hours spent in this way and the cost of a some petrol are worth their weight in gold.

I hope that in these two articles I have managed to cover and answer many of the questions which are likely to arise when contemplating an upgrade. If you need further help then contact the various suppliers and of course BEEBUG. And if you've already made the jump and have any advice which you think will help others then let us know and we will pass it on.

Suppliers referred to in the text:

Ace Computing 27 Victoria Road, Cambridge CB4 3BW. Tel. (0223) 322559.

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

Clares Micro Supplies 98 Middlewich Rd, Northwich, Cheshire CW9 7DA. Tel. (0606) 48511.

Colton Software Broadway House, 149-151 St Neots Road, Hardwick, Cambridge CB3 7QJ. Tel. (0954) 211472.

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

Dabs Press 5 Victoria Lane, Whitefield, Manchester M25 6AL. Tel. 061-766 8423. Electromusic Research Ltd 14 Mount Close, Wickford, Essex SS11 8HG. Tel. (0702) 335747.

Minerva Software 69 Sidwell Street, Exeter EX4 6PH. Tel. (0392) 437756.

Mitre Software Ltd International House, 26 Creechurch Lane, London EC3A 5BA. Tel. 01-283 4646.

Pandora Technology Ltd 9 St Marks Place, London W11 1NS. Tel. 01-221 9653.

Silicon Vision Ltd Signal House, Lyon Road, Harrow, Middx HA1 2AG. Tel. 01-422 2274.

Watford Electronics Jessa House, 250 Lower High Street, Watford WD1 2AN. Tel. (0923) 37774.

Most products for the Archimedes are obtainable through BEEBUG - see the Retail Catalogue.

Weaving Patterns (continued from page 7)

```
1470 ch%=1:cv%=2:r%=1:w%=3:Y%=100:Q%=0
1480 CLS:PRINTTAB(4,2) "WEAVING ON A LOO
M"
1490 PRINTTAB(4,4) "Number of threads in
repeat"
1500 REPEAT: PRINTTAB(4,6) "Vertical:"; TA
B(16, 6); SPC3: INPUTTAB(16, 6) "" R%: UNTIL R
%<=20
1510 U%=720DIV(R%*8):R%=R%-1
1520 REPEAT: PRINTTAB(4,8) "Horizontal:";
TAB (16,8) SPC20: INPUTTAB (16,8) "" S%: UNTIL
1530 S%=S%-1:IF S%MOD2=0 THEN PRINTTAB(
20,8) "Must be even": C%=INKEY(100): GOT015
1540 PRINTTAB(24,6) "V) or H) to":PRINTT
AB(24,8) "set pattern"
 1550 FOR Q%=0 TO S%
 1560 FOR P%=0 TO R%
 1570 PRINTTAB(20-R%+2*P%, 19-S%+2*Q%)"."
 1580 NEXT P%,Q%
 1590 FOR Q%=0 TO S%
 1600 FOR P%=0 TO R%
 1610 REPEAT:*FX21,0
```

1620 T\$=CHR\$(GET AND &DF) 1630 UNTIL T\$="H" OR T\$="V" 1640 PRINTTAB(20-R%+2*P%, 19-S%+2*Q%)T\$ 1650 P\$(P%,Q%)=T\$ 1660 NEXT P%,Q% 1670 PRINTTAB(4,29)"P)lain, C)heck or R)eset Pattern."; 1680 Q\$=CHR\$ (GET AND &DF) :CLS 1690 ENDPROC 1700 : 1710 DEF PROCcol 1720 C%=(INKEY(0) AND &DF) 1730 IF C%=ASC("H") THEN VDU19,3,ch%,0, 0,0:ch%=ch%+1:IF ch%>7 THEN ch%=1 1740 IF C%=ASC("V") THEN VDU19,1, cv%,0, 0,0:cv%=cv%+1:IF cv%>7 THEN cv%=1 1750 IF C%=ASC("S") THEN X\$=GET\$ 1760 ENDPROC 1770 : 1780 DEF FNend 1790 PRINTTAB(3,2)"R)eset or Q)uit"SPC2 2; 1800 REPEAT:Q\$=CHR\$(GET AND &DF):UNTIL Q\$="R" OR Q\$="Q" B

Run your programs from an Archimedes-style desktop with this utility from Jonathan Ribbens.

The purpose of this utility is to create a simple desktop for BBC computers equipped with the ADFS. When it is run, files in the current directory are displayed graphically on the screen as icons, with the type of file included where this can be deduced from the file information. Thus Basic programs, ROM images (which *must* have a load address of &8000 to be recognised as such) and directories are all indicated as such on the screen.

A pointer is displayed, which can be moved around the desktop with the cursor keys. Function key f0 simulates a mouse button, and is used to select items on the desktop as required.

THE LISTINGS

The desktop consists of two programs. The first, *Desktop*, sets up a number of user-defined characters and assembles some machine code to move the pointer. The second program, *Desk2*, handles the operation of the desktop. Type in the two listings carefully and save them with the names specified. The desktop can then be entered by typing CHAIN"DESKTOP".

DETAILED DESCRIPTION

When the program is run, a "desktop" is drawn on the screen. On the top line are shown a number of options, which will be described later. Below this is a line containing the name of the current directory, and the rest of the screen displays the file icons. Files are shown as square boxes, containing the legend "BAS" for Basic programs, "ROM" for ROM images, or nothing for unknown file types. Directories are shown as folders, with the legend "DIR". An ADFS directory can contain 47 entries, but there is only space on the screen for 36 items. If the current directory contains more than this number, pressing the Tab key will toggle the display between the first 36 and the remainder. This is indicated by a prompt on the second line to the right of the current directory name.

To the left of the second line is a large "X" symbol. Selecting this symbol (i.e. by moving the pointer over it and pressing f0) moves back up one directory level (unless the root directory is already the current one) and displays its contents.

Beebug April 1990

Selecting any item within the current directory will highlight its icon by reversing the colours. Selecting an icon already highlighted will activate that selection, in a similar way to double clicking on the Archimedes and other Wimp systems, except that you will find you need to pause between the two key presses. If it is a Basic program it will be chained, if a directory it will be selected as the current directory and its contents displayed, and if of unknown type it will be assumed to be machine code and a *RUN operation will be performed. If it is a ROM image you will be prompted for the slot number into which to load it. The SRLOAD command in line 1430 in listing 2 is for the Master; model B owners may need to change this to suit their sideways RAM.





The options at the top of the screen are Show, Dump, Info, New Disc and Quit. If a file is highlighted, all the printable ASCII codes in it may be viewed by selecting *Show*. Selecting *Dump* will display the file in hex and ASCII. *Info* displays the attributes of each entry in the directory, in just the same way as typing *INFO. *New Disc* performs a *MOUNT and re-runs the desktop, which is useful if you wish to swap discs in the drive. *Quit*, unsurprisingly, leaves the desktop.

When using the desktop, bear in mind that it is simply a program, not an operating environment in the way that a proper Wimp system is. In other words, once you have selected a file to be run, the

desktop will simply disappear. If you are writing your own Basic programs, however, you can always arrange for your program to chain the desktop on exit (making sure that the correct directory has been selected). You could perhaps put copies of the desktop programs in the root directory of each of your program discs, and chain *Desktop* directly from the !Boot file.

The program will also work with Econet without any modifications.

Listing 1

```
10 REM Program Desktop
   20 REM Version B1.0
   30 REM Author Jonathan Ribbens
40 REM BEEBUG April 1990
   50 REM Program subject to copyright
   60 :
  100 *FX 20,2
  110 A%=131:B%=(USR(&FFF4)AND&FFFF00)/2
56
  120 IF PAGE<B%:PAGE=B%:CHAIN"Desktop"
  130 VDU6:FOR p=0 TO 2 STEP 2:P%=&A00
  140 [OPTp
  150 LDA#&40:STA&70:LDA#&59:STA&71
  160 LDA#&55
  170 .lp1
  180 LDY#0
  190 .lp2
  200 STA(&70), Y:CMP#&55
  210 BEQ skip:LDA#&55:BNEcont
  220 .skip
  230 LDA#&AA
  240 .cont
  250 INY: BNE1p2: INC&71: LDX&71
  260 CPX#&80:BNElp1:RTS
  270 ]NEXT
  280 VDU23, 128, 0, 248, 224, 192, 128, 128, 0,
  290 VDU23, 129, 0, 31, 7, 3, 1, 1, 0, 0
  300 VDU23, 130, 0, 66, 36, 24, 24, 36, 66, 0
  310 VDU23, 131, 0, 126, 66, 90, 90, 66, 126, 0
  320 VDU23, 132, 0, 42, 106, 10, 122, 2, 126, 0
  330 VDU23, 133, 0, 63, 64, 128, 128, 179, 185,
169
  340 VDU23, 134, 0, 0, 128, 126, 2, 178, 58, 42
  350 VDU23, 135, 169, 169, 169, 185, 179, 128,
255,0
  360 VDU23, 136, 42, 58, 50, 42, 170, 2, 254, 0
  370 VDU23, 137, 255, 128, 130, 128, 174, 186,
128,190
  380 VDU23,138,255,1,1,1,1,1,1,1,1
  390 VDU23, 139, 168, 190, 128, 182, 170, 190,
128,255
  400 VDU23, 140, 1, 1, 1, 1, 1, 1, 1, 1, 255
  410 VDU23, 141, 255, 128, 128, 190, 144, 190,
128,190
  420 VDU23, 142, 255, 1, 1, 1, 1, 1, 1, 1, 1
```

430 VDU23, 143, 162, 190, 128, 186, 172, 190, 128,255 440 VDU23, 144, 1, 1, 1, 1, 1, 1, 1, 1, 255 450 VDU23, 145, 255, 128, 128, 128, 128, 128, 128,128 460 VDU23, 146, 255, 1, 1, 1, 1, 1, 1, 1, 1 470 VDU23, 147, 128, 128, 128, 128, 128, 128, 128, 128.255 480 VDU23, 148, 1, 1, 1, 1, 1, 1, 1, 1, 255 490 VDU23, 149, 0, 0, 112, 80, 112, 80, 80, 0 500 VDU23, 150, 0, 0, 96, 80, 96, 80, 96, 0 510 VDU23, 151, 0, 0, 112, 64, 64, 64, 112, 0 520 VDU23, 152, 0, 0, 96, 80, 80, 80, 96, 0 530 VDU23, 153, 0, 0, 112, 64, 96, 64, 112, 0 540 VDU23, 154, 0, 0, 112, 64, 96, 64, 64, 0 550 VDU23, 155, 0, 0, 112, 64, 80, 80, 112, 0 560 VDU23, 156, 0, 0, 80, 80, 112, 80, 80, 0 570 VDU23, 157, 0, 0, 112, 32, 32, 32, 112, 0 580 VDU23, 158, 0, 0, 112, 32, 32, 32, 96, 0 590 VDU23,159,0,0,80,80,96,80,80,0 600 VDU23,160,0,0,64,64,64,64,112,0 610 VDU23, 161, 0, 0, 80, 112, 80, 80, 80, 0 620 VDU23, 162, 0, 0, 80, 112, 112, 112, 80, 0 630 VDU23,163,0,0,112,80,80,80,112,0 640 VDU23, 164, 0, 0, 112, 80, 112, 64, 64, 0 650 VDU23, 165, 0, 0, 112, 80, 80, 112, 112, 0 660 VDU23, 166, 0, 0, 112, 80, 112, 96, 80, 0 670 VDU23, 167, 0, 0, 112, 64, 112, 16, 112, 0 680 VDU23, 168, 0, 0, 112, 32, 32, 32, 32, 0 690 VDU23,169,0,0,80,80,80,80,112,0 700 VDU23, 170, 0, 0, 80, 80, 80, 80, 32, 0 710 VDU23, 171, 0, 0, 80, 80, 112, 112, 32, 0 720 VDU23, 172, 0, 0, 80, 80, 32, 80, 80, 0 730 VDU23, 173, 0, 0, 80, 80, 80, 32, 32, 0 740 VDU23, 174, 0, 0, 112, 16, 32, 64, 112, 0 750 VDU23, 175, 0, 0, 64, 64, 64, 0, 64, 0 760 VDU23, 176, 0, 0, 32, 80, 80, 80, 32, 0 770 VDU23, 177, 0, 0, 96, 32, 32, 32, 32, 0 780 VDU23, 178, 0, 0, 112, 16, 112, 64, 112, 0 790 VDU23, 179, 0, 0, 112, 16, 48, 16, 112, 0 800 VDU23, 180, 0, 0, 64, 64, 112, 32, 32, 0 810 VDU23, 181, 0, 0, 112, 64, 112, 16, 112, 0 820 VDU23, 182, 0, 0, 112, 64, 112, 80, 112, 0 830 VDU23,183,0,0,112,16,16,32,32,0 840 VDU23,184,0,0,112,80,112,80,112,0 850 VDU23, 185, 0, 0, 112, 80, 112, 16, 16, 0 860 VDU23, 186, 0, 0, 0, 0, 0, 0, 112, 0 870 VDU23, 187, 0, 0, 0, 0, 112, 0, 0, 0 880 VDU23,188,0,0,0,112,0,112,0,0 890 VDU23, 189, 0, 0, 0, 16, 32, 32, 64, 0 900 VDU23, 190, 0, 0, 0, 96, 64, 64, 96, 0 910 VDU23, 191, 0, 0, 0, 96, 32, 32, 96, 0 920 VDU23, 192, 0, 0, 32, 64, 64, 64, 32, 0 930 VDU23, 193, 0, 0, 64, 32, 32, 32, 64, 0 940 VDU23, 194, &55AA; &55AA; &55AA; &55AA; 950 FOR p=0 TO 2 STEP 2:P%=&900 960 [OPTp 970 JMP showpointer 980 JMP deletepointer 990 .showpointer

1000 JSR calcaddr 1010 LDY#0 1020 .lp1 1030 LDA(&70), Y:STAstordat, Y 1040 ANDmaskdat, Y: ORApoindat, Y 1050 STA(&70), Y: INY: CPY#8: BNE1p1 1060 JSRnewline:LDY#0 1070 .lp2 1080 LDA(&70), Y:STAstordat+8, Y 1090 ANDmaskdat+8, Y: ORApoindat+8, Y 1100 STA(&70), Y: INY: CPY#8: BNE1p2: RTS 1110 .poindat 1120 EQUD &60400000:EQUD &7E7C7870 1130 EOUD & 0C0C1818: EOUD & 00000606 1140 .maskdat 1150 EQUD &070F1FFF:EQUD &00000103 1160 EOUD & EOC1C100: EOUD & FFF0F0E0 1170 .stordat 1180 EQUD &00000000:EQUD &00000000 1190 EQUD &00000000:EQUD &0000000 1200 .deletepointer 1210 JSR calcaddr:LDY#0 1220 .lp3 1230 LDAstordat, Y:STA(&70), Y 1240 INY:CPY#8:BNE1p3:JSRnewline 1250 LDY#0 1260 .lp4 1270 LDAstordat+8,Y 1280 STA(&70), Y: INY: CPY#8: BNE1p4: RTS 1290 .calcaddr 1300 LDA#&5800 MOD256:STA&70 1310 LDA#&5800 DIV256:STA&71 1320 CPX#0:BEO doy 1330 .lp5 1340 LDA&70:CLC:ADC#8:STA&70 1350 LDA&71:ADC#0:STA&71:DEX:BNE1p5 1360 .doy 1370 CPY#0:BEQ endcalc 1380 .1p6 1390 JSRnewline: DEY: BNE1p6 1400 .endcalc 1410 RTS 1420 .newline 1430 LDA&70:CLC:ADC#&40:STA&70 1440 LDA&71:ADC#&01:STA&71:RTS 1450 INEXT 1460 CHAIN"Desk2"

Listing 2

```
10 REM Program Desk2
20 REM Version B1.0
30 REM Author Jonathan Ribbens
40 REM BEEBUG April 1990
50 REM Program subject to copyright
60 :
100 ON ERROR PROCtidy:CLS:REPORT:PRINT
ERL:END
110 MODE4:VDU19,0,7;0;19,1,0;0;30,128,
31,39,0,129,23;8202;0;0;; :MOVE 0,1023:D
RAW 1279,1023:CALL&A00
```

120 PROCtext ("Show"+STRING\$ (4, " ")+"Du mp"+STRING\$(4," ")+"Info"+STRING\$(6," ") +"New Disc"+STRING\$(12," ")+"Ouit"+STRIN G\$(14, " ")+"DeskTop", 96, 1020) 130 OSCLI"FX200, 1": PROCinit: PROCmain 140 VDU26:CLS:PROCtidy:END 150 : 1000 DEF PROCtext (A\$, a%, b%) :LOCAL 1%:VD U5:FOR I%=1 TO LENA\$:MOVE a%, b%:VDUFNfon t (ASCMID\$ (A\$, I%)) :a%=a%+16:NEXT:VDU4:END 1010 : 1020 DEF PROCbox(X1%, Y1%, X2%, Y2%):MOVE X1%, Y1%: MOVE X2%, Y1%: PLOT 85, X1%, Y2%: PLO T 85, X2%, Y2%: ENDPROC 1030 : 1040 DEF PROCinit:dx1%=1:dx2%=38:dy1%=2 8:dy2%=2:S%=255:scr%=0:DIMf\$(46),t%(46): 1050 : 1060 DEF PROCclear:X%=1:Y%=0:PROCread:V DU17, 128, 17, 0, 12, 17, 128, 17, 1, 88: GCOL0, 1: MOVE x1%, y2%-32:DRAW x2%, y2%-32:IF T\$<>" " PROCstring(T\$,2,0) 1070 PROCfiles: IF F% FOR i%=FNstart TO FNend: PROChighlt (i%): NEXT: IF F%>35 PROCs tring("Press TAB for more files", 16,0) 1080 swap%=FALSE:ENDPROC 1090 : 1100 DEF PROCmove: IF INKEY-26 IF X%>1 C ALL&903:X%=X%-1:CALL&900 1110 IF INKEY-122 IF X%<39 CALL&903:X%= X%+1:CALL&900 1120 IF INKEY-58 IF Y% CALL&903:Y%=Y%-1 :CALL&900 1130 IF INKEY-42 IF Y%<29 CALL&903:Y%=Y %+1:CALL&900 1140 IF INKEY-97 IF F%>35 swap%=TRUE 1150 A%=19:CALL&FFF4:CALL&FFF4:ENDPROC 1160 : 1170 DEF PROCstring(A\$,a%,b%):LOCAL x%, y%:GCOL0,1:x%=a%*32+dx1%*32:y%=(1024-(b% *32+dy2%*32)):PROCtext(A\$,x%,y%):ENDPROC 1180 : 1190 DEF PROCread: LOCAL A%, X%, Y%, I%:?&7 0=0:!&71=&A40:!&75=1:!&79=0:X%=&70:Y%=0: A%=5:CALL&FFD1:A%=6:CALL&FFD1:T\$="":B%=& A41+?&A40:FOR I%=1 TO ?B%:T\$=T\$+CHR\$(I%? B%):NEXT:IF ASCT\$=36 OR ASCT\$=38 T\$="ROO T DIRECTORY": root%=TRUE ELSE root%=FALSE 1200 ENDPROC 1210 : 1220 DEF PROCfiles:LOCAL A%, X%, Y%, I%, J% :F%=0:!&A49=0:X%=&40:Y%=&A:A%=8:REPEAT:? &A40=0:!&A41=&70:!&A45=1:?&71=13:CALL&FF D1:IF ?&71<>13 f\$="":FOR J%=1 TO ?&70:f\$ =f\$+CHR\$(J%?&70):NEXT:f\$(F%)=LEFT\$(f\$,IN STR(f\$+" "," ")-1):F%=F%+1 1230 UNTIL ?&71=13:IF F%=0:ENDPROC

1240 F%=F%-1:A%=5:!&A40=&A60:FOR J%=0 T

$0 = 9 \cdot (c_A (0 - f(19) \cdot 19 - (IICD(FFDD)) AND (FF))$	1480 DEF PROCEDOW: IF + & (S&)=0 ENDPROC
0 F 8. 3 & A00-13 (08) . 18- (03 Karrbb) AND arr.	1400 DEF INCOSIOW.IF (8(58)-0 ENDINCE
$E_{3}=!\&A46$ AND &FFFF:IF $I_{3}=2$ $C_{3}(J_{3})=0$ ELSE	1490 CLS:28=OPENINI\$(S8):VD014:REPEAT:B
IF E%>&8000 AND E%<&8030 t%(J%)=1 ELSE	%=BGET#Z%:IF B%>31 AND B%<127 VDUB% ELSE
IF (!&A42 AND &FFFF) = &8000 t%(J%) = 2 ELSE	IF B%=10 OR B%=13 PRINT
$+ \Re (T_{\Re}) = 3$	1500 UNTIL EOF#Z% OR INKEY(-113):CLOSE#
	7% · UDII 5 · OSCI I "FYI 5" · DROCLOU · FNDDROC
1250 NEAT; ENDEROC	1510
1260 :	1510 :
1270 DEF PROChighlt(1%):IF 1%=255 OR 1%	1520 DEF PROCdump: IF t% (S%)=0 ENDPROC
<fnstart 1%="" or="">FNend ENDPROC ELSE VDU31,</fnstart>	1530 CLS: Z%=OPENINf\$ (S%): P%=0: VDU14: REP
6*((I%-FNstart)MOD6)+1,2+((I%-FNstart)DI	EAT:PRINTRIGHT\$("00000"+STR\$~P%,5);" ";:
V61*4	FOR 18=0 TO 7:182&70=32:1F NOT EOF#28:18
1000 TE +8/TE1-0 IDII 22 124 10 0 0 125	2c70 - pCpr #79, ppTNrpTCurs("0.0"+Crps, (192))
1200 IF La(1a) -0 VD0133,134,10,0,0,133,	20) 0) II II. ELCE DELIMIN II.
136 ELSE IF C8(18)=1 VD0137,138,10,8,8,1	70),2); "; ELSE PRINI ;
39,140 ELSE IF t%(I%)=2 VDU141,142,10,8,	1540 NEXT: FOR 18=0 TO /: B8=18?&/0:1F B8
8,143,144 ELSE IF t%(I%)=3 VDU145,146,10	>31 AND B%<127 VDUB% ELSE VDU46
.8.8.147.148	1550 NEXT:P%=P%+8:UNTIL EOF#Z% OR INKEY
1290 VDU10, 8, 8, PROCString(f\$(T%), POS, VP	(-113):CLOSE#Z%:VDU15:PROCkey:ENDPROC
OS) + FNDDDOC	1560 •
1200	1570 DEE DROCKAN OCCITIEVISI DRINTIND
1300 :	1570 DEF PROCKEY: USCLI FAIS :PRINT Pre
1310 DEF PROCmain: quit%=FALSE:x1%=dx1%*	ss any key";:2%=GET:PROCCIear:COLOUR129:
32:x2%=dx2%*32+32:y1%=992-dy1%*32:y2%=10	COLOUR0:PROChighlt(S%):COLOUR128:COLOUR1
24-dy2%*32:GCOL0,1	: ENDPROC
1320 PROChox $(x1\%-4, y1\%-4, x2\%+4, y2\%+4)$:P	1580 :
$POChov(v19=16, v19=16, v29=16, v29=16) \cdot CCOL$	1590 DEE PROCipfo.CLS.VDU114.PRINT"Name"
	TODO DEL TROCINIO. CLO. VDOIA. INTRIA AGAIN
0, 0: PROCDOX (X1%, Y1%, X2%, Y2%): VD026, 28, dx	; TAB(15); "LOAD EXEC LENGTH" : LOCAL AS, AS
1%, dy1%, dx2%, dy2%	, Y%, 1%: FOR 1%=0 TO F%: X%=& /0: 1%=0: A%=5:!
1330 PROCclear:REPEAT:CALL&900:REPEAT:P	&70=&A00:\$&A00=f\$(1%):CALL&FFDD:COLOUR12
ROCmove:UNTIL INKEY-33 OR swap%:CALL&903	8-(I%=S%):COLOUR1+(I%=S%)
1340 IF swap% scr%=scr% EOR 1:PROCclear	1600 IF t%(I%)=0 PRINTf\$(I%); TAB(13); "D
ELSE IF Y%=0 PROCOptions ELSE IF X%=dx1	": ELSE PRINTf\$(1%):TAB(15):FNhex(!&72);
& AND Ve-dure AND roote-FAICE OCCITUDIDA	"". $FNbox(1576)$." ". $FNbox(157A)$.
AND IS=dy28 AND FOOLS-FALSE OSCLI DIR	1 (10, COLOUD100, COLOUD1, DDINE, NEVE, DDOCK
":PROCClear:S%=255 ELSE IF Y%>2 C%=(X%-d	1610 COLOURI28:COLOURI:PRINT:NEXT:PROCK
x1%)DIV6+(Y%-dy2%)DIV4*6+FNstart:IFC%<=F	ey:VDU15:ENDPROC
Nend PROCwindow	1620 :
1350 UNTIL quit%:ENDPROC	1630 DEF PROCnew:X%=&70:Y%=0:A%=0:IF (U
1360 :	SR&FFDA AND 255) <>8 CLS:PRINT"Filing sys
1370 DEEPROCWIDDOW: IF S%=C% PROCESSOR EL	tem is not ADES" PROCKEY ELSE OSCUT"MOUN
CE COLOUD120, COLOUD0, PROChigh1+ (C%), COLO	T", DDOColoar, S%-0
SE COLOURIZ . COLOURO . FROCHIghic (Co) . COLO	1 .FROCCIERI. 5 0-0
UR128:COLOURI:PROChighit (S%):S%=C%	1640 ENDPROC
1380 ENDPROC	1650 :
1390 :	1660 DEFPROCtidy:OSCLI"FX229,0":VDU22,4
1400 DEF PROCexec: IF t%(S%)=0 OSCLI"DIR	,23;10,103,0;0;0;
"+f\$(S%):PROCclear:S%=255 ELSE IF t%(S%	1670 guit%=TRUE:OSCLI"FX15":ENDPROC
1-1 ppoctidy: CUNTNFS (S9) FIGE IF +8(S8) =	1680 .
2 PROCEEDY, CHAINES (5%) ELSE IF Co(5%)-	1000 . 1000 DEE ENand: LE $E^{0} < 26$ OB $aar^{0} = 1 \cdot - E^{0}$
S PROCEDAY: OSCIT.,+IS(2%) FISE PROCLOWI	1700 25
oad	1700 =35
1410 ENDPROC	1710 :
1420 :	1720 DEF FNstart:=36*scr%
1430 DEF PROCromload:CLS:OSCLI"FX15":IN	1730 :
PUT"Load ROM image into which"!!"sideway	1740 DEF FNhex $(A\%) = RIGHTS ("0000"+STRS~A$
a DAM alot "ralot% OCCLITEDIOAD "+f¢/6%)	2 A)
S RAM SIOL ; SIOLO: USCLI SRLUAD +19(5%)	1750
+" 8000 "+STR\$SIOL%:PROCCLEAR:ENDPROC	1/50 :
1440 :	1/60 DEF FNiont (A%):1F A%>64 AND A%<91:
1450 DEF PROCoptions:x%=X%-2:IF x%DIV4=	=A%+84 ELSE IF A%>96 AND A%<123:=A%+52 E
0 AND S%<255 PROCshow ELSE IF x%DIV4=1 A	LSE IF A%=33:=175 ELSE IF A%>47 AND A%<5
ND S%<255 PROCOUMP ELSE IF x%DIV4=2 PROC	8:=A%+128 ELSE IF FNfont1:=W%+185-(W%>3)
info FLSE IF x&DIVA=3 OF x&DIVA=A PPOCDO	1770 =0
THE BLOD IF ADDIVI-5 ON ADDIVI-4 PROCHE	1780 .
1460 PRODUCE OK X6DIV4=0 PROCEIDY	1700 . 1700 DEE ENfort1.129 THOURD / 1 _ [] / 1 OUD
1460 ENDPROC	1/90 DEF FNIONCI:W%=INSTR("=[]()", CHR
1470 :	\$A%):=W%

Beebug April 1990

0

Curve Fitting (Part 1)

In this two part series, Sheridan Williams shows how programmers can make use of the technique of curve fitting.

A task which can seem initially difficult, but which has many applications, is that of fitting a smooth curve through a set of points. It is not too difficult to do by hand with graph paper, but how do you set about writing a program to achieve the same result? That is what we shall be looking at this month and next.

Far from being of use only to mathematicians, the ability to represent a set of values by an expression representing a smooth curve, has a wide number of applications. For instance, there are many occasions when it would be useful to have a formula to represent a set of points rather than use a table stored in DATA statements, or a series of IF statements.

To give a simple example, a company charges £30 for installing a single widget, £50 for installing 5 widgets, £80 for 10, £120 for 20, and £180 for 50. An invoicing program would now have to charge accordingly, which is all very straightforward if exactly 1, 5, 10, 20 or 50 are being installed, but what if 3, 9, 15 or even 70 are required?

It is not complicated to interpolate mid way between points to find the charge for 15; it would be mid way between £80 and £120 - £100. Similarly to find a charge for 70 widgets would involve a process called extrapolation. Thus 70 widgets would cost:

 $\pm 180 + (\pm 180 - \pm 120) * (70 - 50) / (50 - 20) = \pm 220$ These processes are linear, and are only strictly accurate if we assume the points are joined by straight lines.

However, as you can see from figure 1, the set of values used in this example do not lie on a straight line.

The way round this is to find a formula (function) that provides a best fit to the points

Beebug April 1990

supplied. The program given this month will fit functions involving only powers of x - called polynomials. A function involving x is termed a first order function, x^2 second order (also known as a quadratic), x^3 third order (cubic) etc. The advantage of a polynomial fit, is that you can keep on generating higher orders until you are happy with the accuracy, finally ending up with an (n-1)th order polynomial which is an exact fit to n points. In reality, if you are not happy with the accuracy obtained by order 4, this would tend to suggest that a polynomial may not actually be the best function to fit the data, and next month we will tackle that point.



Figure 1. Plot of no. of widgets against price

The program supplied this month fits a succession of increasing order polynomials to the data. Type in the program and save it. Running our widget data through the program (where x is the number of widgets) produces the following polynomials:

1 (linear)	41.205 + 2.953x
2 (quadratic)	23.448 + 6.017x - 0.058x ²
3 (cubic)	22.897 + 6.243x - 0.073x ² + 0.000215x ³
4 (quartic)	26.500 + 3.091x + 0.433x ² - 0.0237x ³ + .0003x ⁴

The program continues to give polynomials from the first order onwards, and it is up to you to halt the process when you have reached the accuracy required. In most instances increasing the order of the polynomial increases the quality of the fit, and the correlation coefficient given after each set of results, gives statisticians a measure of this quality. Table 1 shows the

results of fitting polynomials up to order 4 (no higher orders are possible with only 5 points). Below the line are the charges calculated using the polynomials corresponding to 15 and 70 widgets.

You are advised to use this technique with great caution, as there could well be no function that will fit all the data. Look at table 1 and see the charge extrapolated by the order 4 polynomial for 70 widgets. A glance at Figure 2



will show why, even though the fit is perfect, it is not appropriate to use it.

Figure 2. Screen dump of widget plot from 1 to 4

In examples where the data is actually sampled from nature, there is often a function that should fit; whereas in the example above the charges were dreamt up by me and therefore is unlikely to be representable by any function.

USING THE PROGRAM

Data is supplied to the program either directly by typing pairs of points separated by commas after the prompt, or more conveniently by supplying a file name from which the data is read. The file should have previously been created using View, Wordwise, Pipedream (TAB format) or any method that produces an ASCII

	Calculated charges				
No. of Widgets	Actual Cost	Order 1 Linear	Order 2 Quadratic	Order 3 Cubic	Order 4 Quartic
1	£30	44.16	29.41	29.07	30.00
5	£50	55.97	52.09	52.31	50.01
10	£80	70.73	77.85	78.24	80.01
20	£120	100.26	120.71	120.27	119.98
50	£180	188.86	180.05	179.37	178.55
15		85.50	100.72	100.84	105.51
70		247.91	161.91	175.81	1448.07
Corr. Coeff	•	0.97	0.9997	0.9997	1.0

Table 1. Calculated results

file. The file uses a free format, provided each value is separated by a space, comma, Return or other non numeric character. A file named "widgets" containing the data used here can be found on this month's magazine disc.

The data and results can be printed, and finally, after the last polynomial has been chosen, you may obtain a plot of the data (the actual data points are marked as triangles) against whichever range of functions you require. When prompted, specify the range of polynomials to be plotted (e.g. as 1,4). You may dump this plot to your printer provided you have the necessary screen dump in line 1420. The program as listed contains the *BDUMP command from BEEBUG's Dumpmaster. Alternatively this could be replace by a *SAVE command to save the screen dump to disc, for example:

*SAVE Screen 5800 8000 5800 5800

UNDERSTANDING THE MATHEMATICS

Suppose we wish to fit a second order polynomial (a quadratic) of the form: y = a + bx+ cx^2 . We need to find values for a, b and c, however three unknowns require three equations. Should you be unfamiliar with the

symbol Σ , it is the Greek letter sigma, and means "the sum of", for example Σ y means the sum of all the y ordinates for each data point.

We can obtain the first equation by summing each of the terms:

 $\Sigma y = an + b\Sigma x + c\Sigma x^2$

A second by multiplying each point by x:

 $\Sigma xy = a\Sigma x + b\Sigma x^2 + c\Sigma x^3$ A third by multiplying by x again:

 $\Sigma x^2 y = a\Sigma x^2 + b\Sigma x^3 + c\Sigma x^4$ Solving simultaneously gives a, b and c.

The mathematics given here are for a quadratic only. The program loops round generating a first order polynomial, then a second order, etc, each time creating a matrix of the coefficients, which is then used to find the unknowns. The correlation coefficient is a measure of the "goodness of fit" and in simple terms is determined by the sum of the squares of all the distances of each point from the curve, this is then scaled to lie between 0 and 1, 0 being no

```
10 REM >polyfit
  20 REM Version B1.2
   30 REM Author Sheridan Williams
  40 REM BEEBUG April 1990
  50 REM Program subject to copyright
  60 REM:
  100 ON ERROR GOTO 960
  101 MODE 4
  110 height=991:width=1279:negligible=1
E-5:np=25
  120 n=50:ply=10
  130 DIM A(ply+1, ply+1), fn$(ply+1):Z1=1
  140 DIM XX(n), YY(n), x(np), y(np)
  150 DIM B(2*n), X(n), SX(2*n), F(n)
  160 max=1E38:x min=max:x max=-max:y mi
n=max:y max=-max
  180 PRINT"BEEBUG POLYNOMIAL CURVE FITT
ING"
  190 PRINT STRING$ (31, "-")
```

200 PRINT" by Sheridan Williams"' 210 print=FNyes ("Do you want printed o utput") 220 PRINT 230 IF FNyes ("Read data from file") PR OCfile ELSE PROCkey 240 IF Z2>ply Z2=ply 250 PRINT"Generating Polynomials in th e range ";Z1;" to ";Z2 260 last=Z2 270 PRINT 280 IF print VDU 2 290 FOR N1=Z1 TO Z2 300 CO=0 310 L=2*N1: M=N1+1 320 FOR I=1 TO L+1:SX(I)=0:B(I)=0:NEXT 330 FOR I=1 TO PA:F(I)=0:NEXT 340 FOR J=1 TO PA:FOR I=0 TO L 350 P=I+1:PW=XX(J)^I 360 SX(P)=SX(P)+PW 370 IF P<=M THEN B(P)=B(P)+YY(J)*PW 380 NEXT:NEXT 390 FOR I=0 TO N1:FOR J=1 TO M 400 A(I+1, J) = SX(N1+2-J+I)410 NEXT:NEXT 420 N=N1+1 430 FOR R=1 TO N-1:FOR K=R+1 TO N 440 M1=R:M2=ABS(A(R,R)) 450 IF ABS(A(K,R))>M2 THEN M1=K:M2=ABS (A(K,R))460 IF M1=R THEN 490 470 FOR J=R TO N:T=A(R,J):A(R,J)=A(K,J): A(K, J) = T: NEXT480 T=B(R):B(R)=B(K):B(K)=T490 M3=A(K, R) / A(R, R)500 A(K,R) = 0510 FOR J=R+1 TO N:A(K,J)=A(K,J)-M3*A(R,J):NEXT 520 B(K) = B(K) - M3 * B(R)530 NEXT K:NEXT R 540 PRINT 550 fn\$(N1) ="" 560 I=N 570 REPEAT: S=B(I): J=I+1 580 IF J<=N S=S-A(I, J) *X(J):J=J+1:GOTO 580 590 IF ABS(A(I,I)) <1E-03 PRINT"EQUATIO N HAS NO RELIABLE SOLUTION": VDU3:END 600 X(I) = S/A(I, I)610 IF X(I)>=0 AND I<>N fn\$(N1)=fn\$(N1) +" +" 620 @%=&01000608 630 fn\$(N1)=fn\$(N1)+STR\$(X(I)) 640 @%=&090A

```
650 fn$(N1)=fn$(N1)+"*X^"+STR$(N1-I+1)
  660 I=I-1
  670 UNTIL I=0
  680 PRINT"Best fit polynomial of order
. ";N1;" is:"'"Y=";fn$(N1)
  690 PRINT
  700 PRINT TAB(7) "X"; TAB(15); "Y"; TAB(23
);"FIT Y";TAB(33);"DIFF"
  710 PRINT STRING$ (39, "-")
  720 @%=&02050A
  730 FOR I=1 TO PA:FOR J=1 TO M
  740 F(I) = F(I) + X(J) * XX(I) ^ (N1 - J + 1)
  750 NEXT
  760 IF YY(I) = 0 D=0 ELSE D=YY(I)-F(I)
  770 PRINT XX(I), YY(I), F(I), D
  780 NEXT
  790 SY=0:S2=0:S3=0:S4=0:S5=0
  800 FOR I=1 TO PA
  810 SY=SY+YY(I):S2=S2+YY(I)*YY(I)
  820 S3=S3+F(I):S4=S4+F(I)*F(I)
  830 S5=S5+YY(I) *F(I)
  840 NEXT
  850 SD=SQR(S2/PA-SY*SY/PA/PA)
  860 S6=SQR(S4/PA-S3*S3/PA/PA)
  870 CO=S5/PA-SY*S3/PA/PA:CR=CO/SD/S6
  880 @%=&050A
  890 PRINT"Correlation coeff between"
  900 PRINT"y and fitted y is "; CR
  910 IF N1<Z2 more=FNyes("Next polynomi
al"): IF NOT more last=N1:N1=Z2
  920 NEXT N1
  930 Z2=last:VDU 3
  940 IF FNyes ("Plot the data") PROCpplot
  950 END
  960 @%=&090A:VDU3:CLOSE#0:IFERR=17 END
  970 REPORT: PRINT" @ line "; ERL: END
  980 :
 1000 DEF FNreadnum:LOCAL num$, ch
 1010 eof=FALSE:REPEAT
 1020 IF EOF#c eof=TRUE:=0
 1030 ch=BGET#c:UNTIL ch>=45
 1040 REPEAT:num$=num$+CHR$ch:ch=BGET#c
 1050 UNTIL ch<45
 1060 IF EOF#c eof=TRUE
1070 =VALnumS
1080 :
1090 DEF FNyes (mess$) : LOCAL Q$
1100 VDU 3:PRINT mess$;" (Y/N)? ";
1110 REPEAT: Q$=CHR$ (GET AND 223)
1120 UNTIL Q$="Y" OR O$="N"
1130 PRINT Q$: IF print VDU 2
1140 = (Q\$ = "Y")
1160 DEF PROCfplot:CLS
```

```
1170 fn$=STR$A+"*"+F1$+"+"+STR$B+"*"+F2
Ŝ
 1180 PROCcalc coords
 1190 PROCdisplay
 1200 PROCplot2
 1210 IF FNdump PRINTTAB(0,0) "Function i
s "; fn$; : PROCdump
 1220 ENDPROC
 1230 :
 1240 DEF PROCpplot:CLS
 1250 REPEAT
 1260 PRINT TAB(0,0) "Input plot range as
 "; Z1; ", "; Z2;
 1270 INPUT Z3,Z4:IF Z3=0 ENDPROC
1280 IF Z3<Z1 OR Z4>Z2 VDU7:PRINT TAB(0
,0) "Range "; Z1; ", "; Z2; " only. Press spac
e when ready. ";:REPEAT UNTIL GET=32:
PRINT TAB(0,0) STRING$(79," ");:GOTO 1260
1290 FOR fn=Z3 TO Z4:fn$=fn$(fn)
1300 PROCcalc coords:NEXT
1310 CLG
1320 FOR fn=Z3 TO Z4:fn$=fn$(fn)
1330 PROCcalc coords: PROCdisplay
1340 NEXT fn
1350 PROCplot2
1360 IF FNdump PRINT TAB(0,0) "Polynomia
1 range from "; Z3; " to "; Z4; : PROCdump
1370 UNTIL FALSE: ENDPROC
1380 :
1390 DEF FNdump:PRINT TAB(0,0); := FNyes(
"Screen dump")
1400 :
1410 DEFPROCdump
1420 *BDUMP
1430 ENDPROC
1440 :
1450 DEF PROCplot2
1460 FOR j%=1 TO PA
1470 x=INT((XX(j%)-x min)/h fact+0.5)
1480 y=INT((YY(j%)-y min)/v fact+0.5)
1490 PROCmark (x, y, 12)
1500 NEXT:ENDPROC
1510 :
1520 DEF PROCcalc coords
1530 from=x min:to=x max
1540 step=(to-from)/(np-1)
1550 point no%=0
1560 FOR X=from TO to+step/2 STEP step
1570 point no%=point no%+1
1580 y=EVAL(fn$):y(point no%)=v
1590 x=X:x(point no%)=x
1600 PROCminmax(x,y)
1610 NEXT
1620 ENDPROC
```

1640 DEF PROCdisplay 1650 REM Calculate scaling factors 1660 IF ABS(x max-x min)<negligible x m ax=x min+2:x min=x min-2 1670 IF ABS(y max-y min) < negligible y m ax=y min+2:y min=y min-2 1680 v fact=(y max-y min)/height 1690 v off=INT(0.5-y min/v fact) 1700 h fact=(x max-x min)/width 1710 h off=INT(0.5-x min/h fact) 1720 PROCaxes 1730 FOR j%=1 TO np 1740 x=INT((x(j%)-x min)/h fact+0.5) 1750 y=INT((y(j%)-y min)/v fact+0.5) 1760 IF j%=1 MOVE x, y ELSE DRAW x, y 1770 NEXT 1780 ENDPROC 1790 : 1800 DEF PROCaxes 1810 @%=&00020305 1820 sx min=x min:sx max=x max 1830 IF h off>=0 AND h off<=width MOVE h off, 0:DRAW h off, height 1840 IF v off>=0 AND v off<=height MOVE 0, v off: DRAW width, v off 1850 VDU5 1860 IF v off>30 xlabel=v off-10 ELSE x label=v off+30 1870 IF v off<0 OR v off>height xlabel= height DIV 2 1880 IF h off<width-200 ylabel=h off+10 ELSE ylabel=h off-200 1890 IF h off<0 OR h off>width ylabel=0 1900 xsize=32*LEN(STR\$(INT(sx max)))+12 1910 IF FNdiff(y min) MOVE ylabel, 30:PR INT y min 1920 IF FNdiff(y max) MOVE ylabel, heigh t:PRINT y max 1930 IF FNdiff(sx min) MOVE 0, xlabel:PR INT sx min 1940 IF FNdiff(sx max) MOVE width-xsize ,xlabel:PRINT sx max 1950 VDU4 1960 @%=2570 1970 ENDPROC 1980 : 1990 DEF FNdiff(x) = ABS(x) > .0052010 DEFPROCSORT (N) : LOCAL I, T, N%, SW 2020 N%=N 2030 REPEAT: N%=N%-1 2040 sw=FALSE

1630 :

2050 FOR I=1 TO N% 2060 IF XX(I) <= XX(I+1) THEN 2100 2070 SW=TRUE 2080 T=XX(I):XX(I)=XX(I+1):XX(I+1)=T 2090 T=YY(I):YY(I)=YY(I+1):YY(I+1)=T2100 NEXT 2110 UNTIL NOT SW OR N%=1 2120 ENDPROC 2130 : 2140 DEF PROCmark(x,y,inc) 2150 REM Currently a triangle 2160 PLOT68, x, y+inc: PLOT68, x-inc, y-inc 2170 PLOT86, x+inc, y-inc: MOVEx, y: ENDPROC 2180 : 2190 DEFPROCminmax(x,y) 2200 IF y<y min y min=y 2210 IF y>y max y max=y 2220 IF x<x min x min=x 2230 IF x>x max x max=x 2240 ENDPROC 2250 : 2260 DEF PROCfile:LOCAL I, c 2270 REPEAT 2280 INPUT"Filename ", name\$ 2290 c=OPENIN name\$ 2300 IF c=0 PRINT"File not found " 2310 UNTIL c>0 2320 IF print VDU2 2330 PRINT"Data being read from file: " :name\$ 2340 I=1 2350 XX(I)=FNreadnum: IF eof VDU7: PRINT" ERROR - File doesn't contain even number of points":CLOSE#c:END 2360 PRINT "x(";I;")=";XX(I); 2370 YY(I)=FNreadnum 2380 PRINT TAB(12); "y("; I; ")="; YY(I) 2390 PROCminmax(XX(I),YY(I)) 2400 IFeof CLOSE#c:PA=I:Z2=PA-1:ENDPROC 2410 I=I+1:GOTO 2350 2420 : 2430 DEF PROCkey:LOCAL I 2440 REPEAT 2450 INPUT"No of pairs of points ", PA 2460 IF PA<2 OR PA>n PRINT"Must be betw een 2 and ";n 247.0 UNTIL PA>2 AND PA<n 2480 Z2=PA-1 2490 PRINT"Number of pairs of points is ";PA 2500 FOR I=1 TO PA:PRINT"Pair "; I;" "; 2510 INPUT XX(I), YY(I) 2520 PROCminmax(XX(I),YY(I)):NEXT 2530 PROCsort (PA) : ENDPROC

Receiving Information by Radio

Mike Wooding reviews a new interface and supporting software for the comprehensive reception of a wide variety of radio data transmissions by computer.

Product	RX-8 Receive System
Supplier	Technical Software,
	Fron, Upper Llandwrog,
	Caernarfon, Gwynedd LL54 7RF. Tel: (0286) 881886.
Price	£259 inc. VAT for the RX-8
	£99 inc. VAT for the GX-2
	(£119 inc. FAX direct printing).

In this review I shall be looking at a newly available software package for the BBC B and Master computers, available from Technical Software. The package is of particular interest to amateur radio stations and short wave listeners, as it supports reception of the eight major transmission modes used for amateur and commercial information exchange, including amateur radio slow-scan television. The package is called the RX-8, RX being short for receive and the 8 referring to the eight modes. Information, as described below, is received by radio, and converted via the interface to text or graphics which is displayed on-screen, and which may be subsequently stored and/or printed.

THE RX-8 MULTI-MODE RECEIVE PACKAGE

The RX-8 is a multi-mode receive only system. The package comprises an EPROM, a User Manual, a Test cassette and an interface with connecting leads. It is well presented and the outward appearance gives a feeling of confidence in the claimed performance.

As the name suggests, there are eight modes supported, these being:

AMTOR/SITOR (ARQ and FEC) - these two modes are very similar, and use two separate frequencies to indicate the high and low signalling tones which, when appropriately coded using the Baudot code (a five character code representing the character set, figures etc.) can be used as an information exchange system. Commonly used by radio amateurs and commercial news services.

ASCII - the American Standard Code for Information Interchange and used as a

computer standard the world over. This system is often used by radio amateurs for exchanging information.

FAX - or facsimile (as it is known in full), is the ever increasingly popular mode of transmitting pictures, drawings, etc., over telephone lines, or in this case, by using radio transmission. Weather pictures are transmitted from Meteosat and other weather satellites by FAX (amongst many other modes) and can be decoded by this package. News services and many commercial enterprises also use FAX to transmit pictures, or whatever, over international distances by direct radio links and by satellite.

MORSE - a telegraphic alphabet using a single frequency, in which the characters are represented by combinations of short and long bursts of the tone. Used in the main by radio amateurs, international shipping and the odd commercial concern.

PACKET RADIO - the system of information exchange whereby the data is sent in short bursts, or packets, between the two stations, with the receiving station error checking the packet and sending a checksum back to the transmitting station. This allows for the same packet of data to be repeatedly sent until correctly received, after which the next packet is sent. The process is repeated until the whole message has been sent and received without error. This system is used extensively by radio amateurs in the U.K. and across the world, with a complex network of repeating stations, mailboxes and satellite gateways (links to other continents).

RTTY - radio teletype is the long-standing commercial and amateur method of using two signalling tones, developed from either teleprinters or computers, for the long-distance radio transmission of text.

SSTV - slow-scan television is a widely used mode of picture transmission used by radio amateurs. Using a system of transmitting frequencies between 1kHz and 2.5kHz, each

.

grey level being represented by a different frequency, pictures can be transmitted using a very much narrower band of frequencies (i.e. 1.5kHz) than with conventional TV pictures (approximately 7.5MHz). However, by the nature of the transmission mode it takes between 8 and 240 seconds to transmit a full picture, depending on the resolution required and whether colour is being transmitted. This means that only still pictures can be sent, the advantage being that these pictures can be sent around the world in one transmission on the short wave bands, whereas conventional television is relatively line-of-site over short distances.

UOSAT 1 and 2 - this final mode allows reception of data and telemetry from the radio amateur satellites launched and controlled by the University of Surrey. These satellites are used by radio amateurs for investigation of long-distance VHF and UHF radio transmission, the satellites having on-board radio repeaters and transponders (frequency changing repeaters). The satellites also transmit other data.



RX-8 Facsimile transmission

THE USER MANUAL

The User Manual is well produced, not one of those photo-copied efforts, but professionally printed. The opening section gives a basic explanation of the 2-tone system used to send information in the various modes. However, as the author states, it is only a very basic introduction to the subject and further information may be required to fully understand the technicalities of the various modes.

The next section explains how to call up the program and the various command keys that are common to all modes. Also given in some detail are explanations of the screen displays, use of the keyboard, the text store and the use of printers and disc/tape storage mediums. At the very beginning of this section the reader is directed to the section at the end of the manual, for instructions on how to install the EPROM in the computer, and also how to connect the interface between the radio and the computer. This I found a little strange, as I expected such instructions to be at the very beginning. However, they are quite comprehensive and I feel sure that even the least adept among us would be more than capable of installing the EPROM and connecting the equipment up correctly. A page dealing with possible problems is provided for those unfortunate enough to experience difficulty in getting the system running correctly.

The remainder of the manual deals with the various modes and how to operate the software for each one. Advice is also given concerning the type of receiver and aerial required, what to

do about computer noise and where on the bands to find the various types of transmission.

CONNECTING THE EQUIPMENT Once the EPROM has been fitted and the computer reassembled, the interface must be connected to the User Port on the BBC using the ribbon cable supplied.

The interface is housed in a neat cream plastic enclosure measuring approximately $15.5 \times 9 \times 4$ cm. Mounted on the front panel is a bank of three push-button switches (ON/OFF, FILTER ON & NARROW) and an LED bargraph display. At the rear are the various interconnection sockets.

The unit is connected to the receiver using the 6-pin DIN to 3.5mm jack lead supplied with the package. If a 3.5mm jack plug does not suit your receiver then inform Technical Software at the time of purchase and the appropriate plug will be fitted to the cable.

THE SYSTEM IN USE

Once all is connected switch on the computer. The usual on-screen prompt line should appear

Receiving Information by Radio

with an extra one above it, announcing the presence of the RX-8 software. If you do not get this prompt message then further investigation inside the computer is required to ascertain the correct insertion of the EPROM.

Having obtained the correct screen message type in *RX8 (or *RX-8, it does not matter which) and you should get the message: INTERFACE NOT OPERATIONAL ?? BASIC

Now switch on the interface and try again!

The software defaults to the radio teletype (RTTY) receive program with a full screen display, featuring the program identification in a bar at the top and the program control bar at the bottom. The control bar indicates which mode the program is in and the various settings selected for receiving that mode. These two bars only occupy about 10% of the screen, leaving the rest for displaying the incoming information.

Switching between the eight receive modes is achieved by a simple 'Shift' and single letter keyboard operation. Within each mode the various facilities available are selected by single key strokes. I found that this simplicity in operation enabled me to learn quickly to use the software effectively. The only requirement was a simple crib sheet until I had become familiar with the operations.

All the modes have some form of fine tuning indicator, either on screen, or by using the LED bargraph display on the Interface. I found this very helpful, as I am not particularly well versed with the nuances of the sounds of the various modes when being received.

Of particular interest to me was the Slow-Scan Television (SSTV) receive mode. I found this system very much simpler to operate than other computer-based receive packages that I have used in the past. Reading the User Manual section dealing with SSTV it appears at first that this mode might be difficult to drive. However, once pictures are being received the simplicity of operation becomes apparent, the results obtained on the screen soon indicate if any of the parameters have been wrongly selected, and these can be changed 'live' and

the results of the changes seen immediately on the screen. Incoming pictures can be frozen on the screen at the end of the incoming frame, and then saved to disc/tape or sent to the printer for hard copy.



RX-8 Receiver System

It would be inappropriate for me in this review to list all the various facilities available in the package. Suffice it to say that any signal that I found on the bands I appeared to be able to decode and read without much difficulty. The software seems able to deal with just about any signal in any of the modes as well as a dedicated unit for that mode.

CONCLUSIONS

All-in-all I found the RX-8 package easy to install and get running. In use, once I had mastered the relatively simple method of operation, the results obtained were quite impressive. For someone who is an avid shortwave listener, or for use by amateurs as the receive half of a transceive unit, I can highly recommend this system. The RX-8 package is highly cost-effective when compared with other units for receiving perhaps only one of these modes. In particular, used as an all mode, colour and black and white, SSTV receive converter, its operation is without fault.

A transmit and receive software package called the GX-2, supporting FAX and SSTV is also available from Technical Software. The facilities available are the same as described above for the reception of FAX and amateur Slow-Scan television. However, to transmit package you MUST be a licensed Radio Amateur B

A Datafile Search Utility

Bernard Hill describes a utility for searching data files for specified text strings.

In BEEBUG Vol.8 No.7 I introduced a utility called *FIND which printed all lines containing a given string within a wordprocessed or other document file. This article extends the idea to a database file, that is one which is formed by the use of PRINT# and BPUT# commands (such as a Masterfile file). Since there is no such concept as a 'line' in a database file (as there is with text), but only saved items, and since most items are short, this utility not only prints the item containing the search string but also the seven on either side (unless it is too close to the beginning or end of the file).

First type in the program, save it as SFINDD and then run it. If it has been correctly entered it will produce afile on disc of length &400 bytes called DFIND. The syntax for running the program is identical to its predecessor, i.e.:

*DFIND <filename> <string>

This will search for a certain string within a file <filename>, and display it with its context on the screen. Pressing any key following this will allow the program to proceed to the next occurrence. Note that (like *FIND) the search is not case sensitive, i.e. "Fred" is matched by "FRED" or "fred". If you wish the search to be case sensitive then add the parameter C to the command, thus:

or:

```
*DFIND <filename> FRED C
*dfind <filename> fred c
```

Should you wish to include spaces in the search string, then place quotation marks around the string, for example:

*DFIND <filename> "BEEBUG MAG"

Unlike the former utility there is no 'N' option as record numbers are always shown on the display.

Beebug April 1990

LIMITATIONS

Note that only values which are saved as strings in the database will be found: this utility will not find values of integer or real variables (Masterfile, however, saves all numeric values as strings). Furthermore, since it accesses a Basic ROM routine to run, it will only work on the same version of Basic with which it was assembled: if you later upgrade from a model B to a Master than you will need to re-run the source code to obtain a version for Basic IV.

The program also uses some of the Basic variables: in particular values of integer and real fields in the database will be displayed using the current value of @%, so the program needs Basic to be the current language before it will run. Should you attempt to run it from Wordwise, for example, a "Not Basic" error is generated.

Line 100 contains an assignment to R%, the target address for the utility to load and execute from, but there is the perennial problem of the best location for machine code utilities such as this. On a Master, using a 1K area at &900 will render sound, RS423, cassette and econet buffers unusable. Model B users have two choices, the first similar to that for a Master:

1. R%=&900. If this area is used then pages 9, A, B and C will be erased - OK provided you don't mind losing your function key definitions, aren't using RS232 or cassette, and have no extended character set.

2. R%=&1300. If you have no (other) files open you can use pages 13-16, or maybe your DFS will let you try pages 15-18, covering its own private workspace. Test this out carefully with R%=&1500.

EXAMPLE APPLICATION

To clarify how the program works, let us consider a datafile formed by the following program:

A Datafile Search Utility

- 10 f=OPENOUT"test"
- 20 BPUT#f,12
- 30 PRINT#f, "Name"
- 40 PRINT#f, "Address"
- 50 PRINT#f, 3.776
- 60 PRINT#f,90
- 70 CLOSE#f

Now the command:

*DFIND test name

produces the following screen report:

Rec#	PTR#	
1	0	B:12
*2	1	S:Name
3	7	S:Address
4	16	R:3.776
5	22	I:90
	any	key

This report indicates the record sought with an asterisk (record 2). The second column indicates the position of the data in the file (the value of PTR#). The column after this shows the type of record saved (Byte, String, Real or Integer) and its value. Pressing any key will then cause the program to finish as in this case there are no more matches for the string "name" in the file.

A NOTE ON FILE STRUCTURES

The co-existence of BPUT# operations with PRINT# operations can cause the program to miss certain search patterns. This is because strings are stored by marking them with an initial zero byte (integers with &40, and reals with &FF - see Figure 1). The program assumes any other value is a byte stored with BPUT. Consequently, a BPUT#0 in the file would cause the program to assume that a string variable follows whose length is stored in the next byte. The program can then carry on and become confused, missing valid string matches as it has become out of step with the actual data, although an even number of zero bytes would be no problem as each pair is then regarded as a string of zero length.

The program works perfectly on files saved without BPUT, but if you have a mixed file then there is a small chance that you may have to resort to your database software if this program does not find your string. This problem is endemic with any database dump program which does not know the structure of the parent database. Fortunately the program works perfectly on Masterfile databases.

0C 00 04 65 6D 61 4E 00 07 73 73 65 72 64 64 41 FF E7 FB A9 71 82	BPUT of 12 String follows: Length of string "Name" stored backwards String follows Length "Address" Real follows Value is 3.776
E7 FB A9 71 82	Value is 3.776
40	Integer follows
00 00 00 3A	Value 13 50

Figure 1. Dump of the file test

```
10 REM Program DFIND
  20 REM Version B1.0
  30 REM Author Bernard Hill
  40 REM BEEBUG April 1990
  50 REM Program subject to copyright
  60 :
 100 R%=&900
 110 O=ASC"""":cntos=0:b=?&8015-48
 120 msg1$="Rec# PTR#"+CHR$13
 130 msg2$=" any key ..."+CHR$13
 140 IF b=1 THEN cntos=&9ED0
 150 IF b=2 THEN cntos=&9EDF
  160 IF b=4 THEN cntos=&A118
  170 IF b=6 THEN cntos=&A090
  180 IF cntos=0 THEN PRINT"Unknown Basi
c":END
  190 basicrom=&780:E=&781:length=&782
  200 handle=&OD :saveptr=&OE
  210 show1=&787:type=&78A:nhist=&78B
  220 y2=&78C:y=&78D:lenstr=&78E
  230 lim=&78F:Rcount=&790:quotes=&792
  240 num=&793:hists=&794:buff=&600
  250 Hnum=7:filename=&7D0
  260 match=&7E0:HIMEM=&7800
  270 FOR opt=4 TO 6 STEP 2
  280 P%=R%:0%=HIMEM
  290 [OPT opt:TYA:PHA
  300 LDA #187:LDX #0:LDY#&FF:JSR &FFF4
  310 STX basicrom
  320 LDA #252:LDX #0:LDY#&FF:JSR &FFF4
  330 CPX basicrom: BEQ go
  340 EQUW 0:EQUS "Not BASIC":EQUB 0
  350 .go LDX #&80:LDA #0
  360 .loop STA &780, X:DEX:BNE loop
  370 PLA: TAY: JSR skipsp: CMP #13: BNE ov0
  380 .help EQUW 0
```

A Datafile Search Utility

390 EQUS "Use *DFIND <file> <string> [El":EOUB 0 400 .ov0 STA filename, X: INX: INY 410 LDA (&F2), Y:CMP #13:BEQ help 420 CMP #32:BNE ov0 430 LDA #13:STA filename, X:JSR skipsp 440 CMP #Q:BNE ov1:DEC quotes 450 INY:LDA (&F2), Y:CMP #Q:BNE ov! 460 LDA #0:STA match 470 INY: JMP next1:.ov1 LDX #0 480 .loop STA match, X: INX: INY 490 LDA (&F2), Y:CMP #13:BEQ finstr 500 BIT quotes: BMI ov2 510 CMP #32:BEQ next1:BNE loop 520 .ov2 CMP #Q:BNE loop:INY 530 .next1 JSR skipsp:CMP #13 540 BEQ finstr: AND #&DF: CMP #ASC"C" 550 BEO setE 560 EQUW 0:EQUS "Option?":EQUB 0 570 .setE DEC E 580 .finstr STX lenstr 590 LDA #0:STA match, X:TXA:LSR A 600 STA lim:LDY #0:DEX 610 .loop LDA match, X:JSR uc:PHA 620 LDA match, Y: JSR uc: STA match, X 630 PLA:STA match, Y:DEX:INY 640 CPY lim: BNE loop 650 LDA match, X: JSR uc: STA match, X 660 LDA #&41:LDX #filename MOD 256 670 LDY #filename DIV 256 680 JSR &FFCE:CMP #0:BNE ok 690 EOUW 0:EOUS "Not found":BRK 700 .ok STA handle 710 .newrec BIT &FF:BPL cont:JMP esc 720 .cont LDA Rcount+1:BNE more 730 LDA Rcount:CMP #Hnum:BCS more 740 STA nhist: JMP ov4 750 .more LDA #Hnum:STA nhist 760 EOUS FNcopy3(hists, show1) 770 .ov4 LDX #3 780 .loop LDA hists, X:STA hists-3, X 790 INX:CPX #Hnum*3:BNE loop 800 LDA #0:JSR args 810 EQUS FNcopy3(&2A, hists+3*Hnum-3) 820 JSR bget: BCS eof 830 INC Rcount: BNE skiph: INC Rcount+1 840 .skiph CMP #0 :BEQ string 850 CMP #&FF:BEQ real:CMP#&40:BEQ int 860 JMP newrec 870 .real JSR bget: BCS eof 880 .int LDX #4:.lp JSR bget:BCS eof 890 DEX: BNE lp: BEQ newrec 900 .string JSR bget:BCS eof 910 STA length:LDY #0 920 .newstart LDX #&FF 930 .nextbyte INX:LDA match, X 940 BEQ matches: CPY length: BEQ eos

950 INY: JSR bget: BCS eof: JSR uc 960 CMP match, X:BNE newstart 970 BEQ nextbyte 980 .matches JSR show 990 .eos JMP newrec 1000 .eof LDY handle:LDA #0:JMP &FFCE 101-0 .show LDX #0 1020 .loop LDA msg1, X:JSR &FFE3 1030 INX:CPX #LENmsg1\$:BNE loop 1040 EQUS FNcopy3(show1, &2A) 1050 LDA #1:JSR args 1060 LDA nhist:EOR #&FF:TAY:INY 1070 .loopy JSR showrec: BCS ret1 1080 CPY #0:BNE ovr:LDA #0:JSR args 1090 EQUS FNcopy3(&2A, saveptr) 1100 .ovr INY:CPY #Hnum+1:BNE loopy 1110 .ret1 EQUS FNcopy3(saveptr, &2A) 1120 LDA #1:JSR args:LDX #26:LDA #45 1130 .loop JSR &FFE3:DEX:BNE loop 1140 LDX #0 1150 .loop LDA msg2, X:JSR &FFE3 1160 INX:CPX #LENmsg2\$:BNE loop 1170 JSR &FFE0:BIT &FF:BPL ov11 1180 .esc LDY handle:LDA #0:JSR &FFCE 1190 BRK: EQUB 17: EQUS "Escape"+CHR\$0 1200 .ov11 RTS 1210 .bget STY y:LDY handle:JSR &FFD7 1220 LDY y:RTS 1230 .args LDX #&2A 1240 STY y:LDY #0 1250 STY &2D:LDY handle 1260 JSR &FFDA:LDY y:RTS 1270 .col LDA #ASC":":JMP &FFE3 1280 .showrec JSR bget:BCS ov11:PHA 1290 LDA #32:CPY #0:BNE ov6:LDA #42 1300 .ov6 JSR &FFE3:TYA:CLC:ADC Rcount 1310 STA &2A:LDA Rcount+1:STA &2B 1320 LDX #4:JSR decout:LDA #0:JSR args 1330 LDA &2A:SEC:SBC#1:STA &2A:BCS ov7 1340 LDA &2B:SBC #1:BCS ov7:DEC &2C 1350 .ov7 1360 LDX #5:JSR decout:PLA:STY y2 1370 STA type:CMP #0:BEQ strout 1380 CMP #&40:BEQ intout 1390 CMP #&FF:BEQ realout 1400 LDA #ASC"B":JSR &FFE3:JSR col 1410 LDA type:STA &2A:LDA #0: 1420 STA &2B:STA &2C:STA &2D: 1430 LDX #3:JSR decout:CLC:JMP endout 1440 .strout LDA #ASC"S":JSR &FFE3 1450 JSR col: JSR bget: BCS err 1460 STA length:LDX #0 1470 .loop JSR bget: BCS err: STA buff, X 1480 INX:CPX length:BNE loop 1490 .loop DEX:CLC:BMI endout 1500 LDA buff, X:JSR &FFE3:JMP loop

Continued on page 42

Music Programming In Ample (Part 1)

Music expert Ian Waugh discusses ways to improve your programming of Hybrid Technology's popular Music 5000 system in this first article of a new series.

INTRODUCTION

It would not be an overstatement to say that the Hybrid Technology Music System has revolutionised music education. It supports all the fundamental principles of music (something which MIDI-based systems cannot do), and it can be used at many levels by anyone from an absolute beginner to an academic musicologist.

Because of its musical power and flexibility and low price compared with other music systems - it has also attracted a large army of dedicated home musicians. Several professional composers have used the system too.

The Hybrid Music System is unique in being the only affordable *computer-based* music system on the market. While it must be admitted that MIDI synthesizers can produce better sounds - at a price - no MIDI sequencer approaches HMS's range of music and programming facilities.

THE HARDWARE AND THE SOFTWARE

The basic unit in the HMS is the Music 5000. This consists of a disc-drive sized box which plugs into the BBC micro and an Ample Nucleus ROM which holds the Ample music language. The Music 5000 can produce 16 channels of sound which are normally paired to form eight voices. You can create your own sounds using a simple control panel editor.

The system is controlled through software editor modules which make many of the programming functions transparent to the user. The Stave Editor lets you enter music on the stave in traditional music notation. The Recorder module is used to enter music from the optional Music 4000 keyboard. The Mixing Desk can be used to mix a piece of music in real-time and to create mix 'snapshots' which can be called up by the music at any time.

The Notepad is a text editor, used to enter music in Ample's MCL (Music Composition

Language). It is also used to edit music, write Ample programs, define instruments, create menus and so on. It's the most powerful of all the editors and the most appropriate for the Ample language.

One of the beauties of the HMS (and one of the reasons why it is so popular in education) is its ability to save *all* music and sound data in one file. This is generally impossible or at least extremely difficult to do with a MIDI system.

Additional hardware units are available to expand the system. The Music 3000 adds another 16 channels of sound to the Music 5000 which can be used to produce more complex instruments (by using more than the usual two channels per voice), and simply to increase the number of voices.

The Music 2000 MIDI interface lets you control drum machines and synthesizers from Ample. It uses special words to send MIDI channel number, program change and pitch bend messages and so on. Being software-based it can transmit *any* MIDI message and it is probably the most comprehensive MIDI controller available.

The latest hardware release is the entry-level Music 5000 Synthesizer Universal. This channels sound commands directed at the BBC's sound chip to the Music 5000 thereby greatly increasing the dynamic range and response of existing music programs. It's transparent in use and works with all correctly written music software.

In addition to the hardware, Hybrid has recently released the Ample Toolbox, a set of software extensions which provides extra facilities for advanced users. These include a full-feature text editor, an image editor for creating mode 7 screens, several utilities such as program merge and compile, an Ample program recover for programs which have accidentally been deleted

and a sideways RAM utility for storing Ample software modules. Highly recommended for the Ample programmer.

Finally, an essential purchase for all programmers is Hybrid's Ample Nucleus Programmer Guide. This explains the Ample programming language in greater detail than the User Guides and contains several applications and example programs.



Using the Studio 5000 Mixing Desk

That, in a nutshell, is the Hybrid Music System. It can be used at Editor level (the Stave, Recorder and Mixing Desk) to produce excellent pieces of music but it's only when you delve below the surface and begin to explore the language that other possibilities reveal themselves. These range from creating simple music instructions such as pitch bend and delayed vibrato to full scale utility programs. For example, I have recently written a Performance Editor for Yamaha's TX81Z expander using Ample's Notepad control panel facility.

PROGRAMMING IN AMPLE

In this three-part series we'll explore some programming ideas and techniques. Some may be familiar to old hands, but I hope everyone will find something new and useful here. Remember - we all have to start somewhere.

Beebug April 1990

However, this series is not for complete beginners (if you are interested in seeing such an introduction to Ample published in BEEBUG, I suggest you write to the Editor). I'm assuming you are familiar with the editors and can use the Notepad. Most of the examples will be based on the Music 5000 although some of the other hardware modules will be mentioned where a technique can be applied to them.

FIRST STEPS

But before you boot your system disc, we'll begin with a few words about musical arrangements. The major drawback with the HMS is not the system itself but the lack of memory in the machine it runs on. The 10K (or possibly less) you have free for your program needs to be used with care. The HMS, however, positively encourages you to enter music in sections and, unless the piece is short, this really is the way you should work. If you are composing a piece on the system 'as you go' this may not be feasible, but if you do work in this way you can select exactly the sections you need, from those completed, for the final piece.

Entering music from a written score is another matter entirely. The music is fully laid out for you - all you have to do is enter it! However, music notation is a linear medium: it starts from the beginning and goes on to the end (although it may include some repeats). Unless, again, the piece is very short there will not be sufficient memory to allow you to enter it in linear fashion nor is it desirable to do so. It is far better to enter it in sections and use Ample to link them together.

Before you even sit down at your system, dissect the music. The first step is to recognise and isolate the individual music parts. Nominally, each music part will be allocated an Ample part word - part1, part2, part3 or part1a, part1b, part2a, part2b and so on - which, in turn, will be made up from shorter sections of music.

If the music contains monophonic (one note) lines or regular chords, this is easy. Arrangements for monophonic ensembles such as a brass band or string quartet are usually easy to handle, too. Most piano music has two obvious parts - the left

Music Programming in Ample

and the right hand - but on closer examination they may reveal further music parts within each hand as in Bach's three-part inventions. If you examine some of his other pieces - fugues, for example - you may see four parts.

Most pop sheet music arrangements have a monophonic bass line in the left hand and in the right a chord accompaniment plus a melody line. Sometimes the accompaniment will disappear and the melody line will play with block chords.

Other music may reveal additional complications. Some parts may chug along monophonically then suddenly burst into life with a chord. As the HMS only has eight voices under normal circumstances, they need to be allocated carefully. To assign three or four voices to a part which is mostly monophonic to enable it to play a couple of chords is extremely wasteful - unless you have voices to spare.

Note: Using the system's default of two channels per voice, the Music 5000/3000 combination has 16 voices; the Music 2000 allows 32 voices via MIDI. Using Ample's powerful ACT command you can also create one-channel instruments - a potential 16 voices. This, however, requires a rather sophisticated programming technique which is beyond the scope of this series so we'll ignore this option. Interested readers are referred to the Ample Nucleus Programmer Guide.

To avoid any confusion between the terms *instrument* and *voice*, here's an explanation: a voice is the part of the system which plays a single line of notes and an instrument is a group of settings which determine the character of the voice's sound.

REPEATS

Having worked out how many parts the music has, the next step is to look for music lines which repeat. As each music part is programmed separately, you are not concerned at this stage with global repeats such as those defined by repeat bar lines, but with repetitive phrases and riffs. These may occur anywhere in the music. Bass lines are the most common source although it's quite common for the same line to be used in separate parts, possibly transposed - fugues are a perfect example.

The actual number of sections you end up with will depend completely upon the music you are arranging, but typically you may expect the amount of music you have to enter to be cut in half, more if it contains global repeats (DS al Coda and repeat bars).

You'll realise the importance of keeping track of all the sections and how they fit together. It's a good idea to write out a section list for each part. This will help during debugging, too.



Using the Studio 5000 Staff Editor

There's one more thing to do and that is to see how many voices the piece uses. To do this, add up the maximum number of notes used by each part at any one time. If this matches the number of voices on your system (taking into account any instruments using more than two channels) then all's well with the world. You can simply assign each part the maximum number of voices it requires.

But what if there are too many voices? Well, then it's decision time. You have two choices alter the arrangement to reduce the number of voices to the maximum available or share the voices between the parts. This assumes that no more than eight voices (or the maximum your system supports) are actually playing at once. Option one is the easier. You may have to

reduce the number of notes in some chords - try this first - or even leave out an entire music line. Let's look at an example:

max notes	instrument
1	Slapbass
3	Upright
3	Elguit
3	Vibglock
	max notes 1 3 3 3

This has a total of 10 voices. My first instinct would be to look at the piano, guitar and vibes parts where three notes occur and see if any could be reduced to two notes. If so, go for it. If you really don't want to do that then voices must be shared or transferred. It's relatively easy in principle but you must be wellorganised.

If all this preamble seems unnecessary, try entering a piece of some complexity without any prior organising. You've been warned!

At this stage you can start entering the music. If you know exactly how many music parts you are going to use - and you do, don't you? - you can begin by assigning voices to them. Then you can play the music at any time as you enter it. Enter the Mixing Desk and, using the example above, define four empty part words in command mode:

"part1"	[]
"part2"	[]
"part3"	[]
"part 4"	[]

Press Tab to enter the editor and press R to run the piece. There's no music to play but the first four parts will appear in the Desk using Simpleins. Make sure the Group function is ON (press G if it isn't). Change the first instrument to Slapbass. Place the cursor on the second, press Tab, type 3 VOICES and press Tab again. Change Simpleins to Upright and the other two voices will appear. Place the cursor on the third Simpleins and repeat, selecting Elguit. We haven't enough voices for Vibglock (we'll transfer them later from the Elguit part) so ignore voice eight for now. Okay, so they're in the wrong order. MAKE the mix and run again and, hey presto, they're in the right order. Now let's put the music in.

SUBMIXES AND UNMIXES

Enter the music using whichever method you prefer - the Music 4000 keyboard, Stave Editor or the Notepad. Arrange the music in sections and bring each part together under a part name - part1, part2 and so on. Part4 won't play, of course, as no voices have been assigned to it, but check that the other parts play correctly.

In true Blue Peter tradition, Program 1 is an example I prepared earlier to demonstrate the voice transfer procedure. The piece revolves around a two-bar riff. The bass and piano parts remain unchanged. In the first bar, Elguit plays three-note chords but in the second it only plays a one-note line. In bar one the Vibglock is silent but in bar two it requires three voices. We want to give two of the Elguit voices to part4 in bar two.

We'll make all the arrangements from the Mixing Desk so enter it, run the program to set up the mix and press Escape. Now we want to free two voices on part 3. To do this we create an unmix which frees voices with the UNUSED command. Enter M5MIX (this is the 'new mix' command), press Tab and remove the brackets around voices six and seven (site the cursor on them and press B). Press I, P and V to include the instrument, pan and volume settings. Press Tab and enter:

"chuck3" NAME UMAKE

You have now made an unmix which will free voices six and seven. Now enter:

"fetch3" NAME MAKE

This is a submix which effectively puts the voices back. It's exactly the same as a normal mix except it only works on selected voices - those without brackets around them. Now enter:

4 SHARE 3 VOICES Vibglock This sets up the three voices we require for part4 (the latest Mixing Desk software, with a D extension, will let you do this using the Delete and Copy keys - contact Hybrid for a free update). Now enter:

"fetch4" NAME MAKE "chuck4" NAME UMAKE

You see how easy it is to create unmixes - they are basically the complement of the submix.

Music Programming in Ample

You can check the effect of these words by entering them and seeing how they affect the Desk. Program 2 shows what the words should contain. Use it as a check only - submixes and unmixes are far better created from the Mixing Desk. Remove the % comments from "v1" and run again. You'll see the Desk reflect the changes as they occur and all parts in the piece should now play correctly, transferring voices from part3 to part4 and back as required.

From this example you will notice that mixes do not have to be named "mix1", "mix2" and so on, and they can be called from anywhere in a program by any part. The User Guide explains how to use submixes in a play line like this:

"1234-12ab3ab2ac"PLAY

but calling them from a part is probably more flexible.

In the next article we'll see how to add articulation, vibrato, pitch bend, echo, auto pan and other musically expressive instructions to an arrangement.

NOTE: The full Ample program to go with part one is included on this month's magazine disc.

Listing 1

"part1" [] "part2" [] "part3" []
"part4" [] "mix" [] "b1" []
"p1" [] "g1" [] "v1" []
"part1" [12 FOR(b1)FOR]
"part2" [12 FOR(p1)FOR]
"part3" [12 FOR(g1)FOR]
"part4" [12 FOR(v1)FOR]
"mix" [M5MIX 48,90=T
1 SHARE 1 VOICES
1 VOICE Slapbass 128 VOL 0 PAN
2 SHARE 3 VOICES
1 VOICE Upright 128 VOL 0 PAN
2 VOICE Upright 128 VOL 0 PAN
3 VOICE Upright 128 VOL 0 PAN
3 SHARE 3 VOICES
1 VOICE Elguit 128 VOL 0 PAN
2 VOICE Elguit 128 VOL 0 PAN

```
3 VOICE Elguit 128 VOL 0 PAN
PNUM SHARE
```

```
"b1" [%STAFF
SCORE48,4BAR
-1:24,CCCCCCCC|FFFFGGGGG|
```

"p1" [%STAFF SCORE48,4BAR -1:192,G(CE)|96,A(CF)B(DG)|

```
"g1" [%STAFF
SCORE48,4BAR
24,^-1:G(CE^)^(^)G(CE)^(^)G(CE)
24,^(^)-1:G(CE)|F(^)edcbagB|
```

"v1" [%STAFF SCORE48,4BAR 192,^| % chuck3 fetch4 24,^0:F(AC)A(CF)f(AC)d(GB)/G(BD)/| % chuck4 fetch3

Listing 2

```
"chuck3" []
             "fetch3" []
"chuck4" [] "fetch4" []
"chuck3" [M5MIX
3 SHARE 3 VOICES
 2 VOICE UNUSED
 3 VOICE UNUSED
PNUM SHARE
"fetch3" [M5MIX
3 SHARE 3 VOICES
 2 VOICE Elguit 128 VOL 0 PAN
 3 VOICE Elquit 128 VOL 0 PAN
PNUM SHARE
"chuck4" [M5MIX
4 SHARE 3 VOICES
 1 VOICE UNUSED
 2 VOICE UNUSED
 3 VOICE UNUSED
PNUM SHARE
"fetch4" [M5MIX
4 SHARE 3 VOICES
 1 VOICE Vibglock 128 VOL 0 PAN
 2 VOICE Vibglock 128 VOL 0 PAN
 3 VOICE Vibglock 128 VOL 0 PAN
PNUM SHARE
                                         B
```



Improving Basic Programs Sideways and Shadow RAM by Mike Williams

This month I want to deal with two topics.

First of all, I want to discuss another programming technique which can help you to produce better and shorter programs, in line with the theme for my previous two articles in this series. Secondly, in response to various requests, I want to deal with the subject of sideways and shadow RAM, a topic which still causes confusion for many readers.

Incidentally, if you have any examples of how programs can be improved by the use of a good technique then let me know. Any ideas contributed which can be incorporated into our First Course series will be paid for pro rata as with other contributions.

This month's ideas again derive from a program which I was editing for the magazine. Originally the author had written a line which was as follows:

IF r%=1 THEN r%=3 ELSE r%=1

Clearly the intention was that the variable r% should oscillate between just two values, 1 and 3. The simplest case of this is where a variable is required to be either TRUE (value -1) or FALSE (value 0), which, following the pattern above, might be written:

IF r%=-1 THEN r%=0 ELSE r%=-1 First of all, since r% can only be TRUE or FALSE we could rewrite this as:

IF r% THEN r%=FALSE ELSE r%=TRUE However, there is an even simpler solution to

this example, to wit:

r%=NOT r%

We have done away with the IF statement altogether, resulting in a very much reduced format. Since r% is restricted to the two logical values of TRUE and FALSE, this works a treat, because NOT(TRUE) produces FALSE, and NOT(FALSE) produces TRUE.

I have introduced this particularly simple example because it gives a first clue as to how we can rewrite the original instruction. What we will do is to replace, completely, the original IF

Beebug April 1990

statement with a straightforward assignment. However, we cannot use logical values this time, so we need to find some other means of making the values of r% cycle repeatedly between 1 and 3. The answer lies in the use of the two operators DIV, and particularly MOD.

Whenever you need to cycle repeatedly around a restricted set of values, MOD will often come to the rescue. MOD gives you the remainder after the integer division of one number by another. Thus 4MOD3 gives the result 1. Given a variable like r%, we need to increment this, and then apply MOD so that if we start with 3 we end up with 1, and if we start with 1 we end up with 3.

Now, if we write, when r%=1:

r%=(r%+2)

the result will clearly be 3 (but not if we start at 3, when the result will be 5). But if we divide that by 4 the remainder will be 1, which is what we want. So our original IF statement can be replaced by:

r%=(r%+2)MOD4

and if you check it out you will find it does just what we want.

Before leaving this topic, let us have a look at some more examples, and a particular problem which can arise. Suppose we want a variable to count 1, 2, 3, 4, 5 repeatedly. You might think a good solution would be to write:

r%=(r%+1)MOD6

but if you take any suitable starting value, and apply this formula progressively you will find that it counts 0, 1, 2, 3, 4, 5. This is one more value than we want, and as so often the computer insists on having zero as one of the values. The key lies in the fact that not only is there a zero, but more importantly that there are six possible values, not five as we need. To get five values we need to use MOD5 (rather than MOD6), but in turn this gives the sequence 0, 1, 2, 3, 4, which is still not quite right. Quite the simplest solution is to add 1 to the value of r% when you need to use it (thus giving 1, 2, 3, 4, 5).

First Course

What you must not do is write:

r%=(r%+1)MOD5+1

Test it out and you will see what I mean. A more complicated solution which does do the trick, is to write:

r%=(r%+1)MOD6-(r%=5)

This reverts back to the previous formula using MOD6 which gives the sequence 0, 1, 2, 3, 4, 5. The additional term will be TRUE (value -1) when r%=5, and zero otherwise. The effect is therefore to add an extra 1 to the value of r% when r% equals 5, thus causing it to skip over the unwanted zero.

The DIV operator has little or no worth in this context, giving simply the result of an integer division (the dividend) when used. The use of MOD, however, can be extremely valuable when there is a need to cycle through a limited range of numbers, as I hope the examples above have shown.

SIDEWAYS RAM, SHADOW RAM AND PRIVATE RAM

One major topic which recent correspondence (see our Postbag pages) shows is of continuing interest is that of *sideways RAM*, and to a lesser extent the use of so-called *shadow RAM* and *private RAM*. As we expect to feature a number of further articles and programs related to this in forthcoming issues, I thought it would be appropriate to discuss this topic at some length for the benefit of those who still feel just a little confused.

USER MEMORY (RAM)

All this derives from the somewhat limited main memory with which the original model B (and model A, for those who remember) was provided. The BBC micro was called a 32K machine, but as users know only too well, this is misleading. That is because this 32K has to provide space for the current screen display, which will take from 1K to 20K depending upon the screen mode in use, while memory up to address &E00 is used by the operating system for the temporary storage of a wide variety of information, and for other purposes. Add a disc filing system to the model B and even more memory is taken up as work space for that. Thus on a DFS disc-based system the available area of memory usually starts at &1900. If you switch on your micro and type:

PRINT ~PAGE

the value for your machine will be displayed. If it is a model B with DFS the result will be &1900, if you have Econet, the ADFS or certain ROMs fitted the value can be even higher (Master owners will normally find this is at &E00 regardless for reasons we shall see shortly).



Figure 1. Basic RAM allocation

The position of screen memory is indicated by the value of HIMEM, which will normally have one of the values shown in Table 1. This also shows the memory available to the user in each mode, assuming a DFS system with PAGE at &1900. This arrangement is also shown more graphically in Figure 1.

Screen Mode	HIMEM	Available Memory
Mode 7	&7C00	24.75K
Mode 6	&6000	17.75K
Modes 4,5	&5800	15.75K
Mode 3	&4000	9.75K
Modes 0,1,2	&3000	5.75K

Table 1. Memory allocation in different screen modes

READ ONLY MEMORY (ROM)

As you know, when you switch on your micro, the operating system (which processes most star commands among other things) and Basic are immediately present. Both OS and Basic are themselves just programs, albeit rather complex. They are stored in a special type of memory called ROM (read only memory), because you cannot alter (or write to) this type of memory. In contrast, user memory is of a type called RAM (random access memory) where you can both store and retrieve information.

In theory, therefore, a BBC micro has, in addition to 32K of RAM, a further 32K of ROM divided into 16K for Basic (from &8000 up to &BFFF) and 16K for the operating system (from &C000 to &FFFF). This is shown in Figure 2.

Now the designers of the BBC micro, given these apparent limitations, were quite clever. First of all, the machine was designed so that Basic, which is a 16K ROM, could be replaced by an alternative 16K ROM. So if you didn't want to program in Basic, but to use a word processor, just replace the Basic ROM by a suitable word processor ROM, switch on and that's what vou would get.

But want if you want both? Well the designers

provided additional sockets in the BBC micro so that as many as four 16K ROMs could be accommodated, though only one could be active at a time, and each ROM appears to occupy the same 16K of memory (from &8000). As a result of this design, sideways ROM became the preferred format for many software products with the advantages of speed and immediacy over disc-based software. But this popularity of format had its own problems.

SIDEWAYS RAM

Although the original model B contained sockets for only four sideways ROMs, the overall design of the machine was capable of handling up to 16 ROMs. Thus third party companies like Watford Electronics and ATPL stepped in to provide add-on boards to provide all the extra sockets required. Suppliers also realised that it was quite feasible for some sockets at least to be designed to accommodate 16K RAM chips. These days most sideways ROM/RAM boards can accommodate both types of chip.

A puzzle to many users though, has always been to know to what purpose sideways RAM may be put. To answer that involves understanding a further level of complexity. Sideways ROM chips contain a program, be it

> word processor, printer dump or whatever. As such, the self same software could exist on disc, and be loaded into sideways RAM, at which point it behaves just like a ROM. We have published many programs in BEEBUG which are designed to be used in just this way, and our current EdiKit series is a prime example.

Sideways RAM can also be used to support what is called the ROM Filing System (RFS), and we have covered this quite recently in the magazine (Vol.8

in the magazine (Vol.8 Nos.8 & 9). It is also possible to use sideways RAM for other purposes, for example to store a library of functions and procedures called from a program in main memory, but this is much more difficult to achieve, though still possible. Again, we have published articles on this aspect in the past, and we have further articles for future issues.

SHADOW RAM

One of the major uses for user RAM on a model B is to store the current screen display. Alternatively, if more memory were provided, the screen image could be stored separately, and memory to do this on a BBC micro is called shadow RAM. Once this has been installed, all programs have at least 25.75K of memory available at all times regardless of mode when shadow RAM is in use (by adding 128 to the



First Course

mode number, or using *SHADOW). Add-on shadow RAM boards are available for the model B from various sources.

In practice, RAM chips are normally 16K or 32K in size, so that although the most memory

consuming screen mode is just 20K, shadow RAM always provides 32K. The surplus 12K is sometimes referred to as private RAM (see later notes on the Master). Depending on software support this may be usable as a printer buffer.

Details of both sideways and shadow RAM usage may vary from one make to



Figure 3. System with sideways and shadow RAM

another, so you need to refer to the instructions supplied with any such boards.

THE MASTER SERIES

Unlike the model B, the Master 128 and Master Compact are provided with sideways RAM, shadow RAM and private RAM as standard. Furthermore, Acorn took advantage of this to redesign both the DFS and ADFS so that any workspace required was located in private RAM. The general arrangement of a system fitted with both sideways and shadow RAM is shown in Figure 3, and in principle this applies equally to an upgraded model B, or to a Master 128 or Master Compact. So far I have tried to give an outline description of sideways and shadow RAM without yet discussing in any detail how to make use of it (though shadow RAM is usually straightforward). I hope to deal with this next time.

This applies also to function key definitions and

to 'exploding' the character set (see the relevant

manual for your machine). As a result, the value of PAGE on all Master series machines is set at

&E00, giving 28.5K of user RAM available at all

times (when shadow modes are used).



EdiKit (Part 4)

Bill Hine adds the finishing touches to this collection of utility commands for Basic programmers.

This is the fourth and final instalment in the development of the EdiKit ROM which provides commands to assist in the development and editing of Basic programs. This month's addition implements the final three commands, SYSINF, VARLIST and FKDEFS. In the next issue, we will look at how you may add your own commands (or previously published programs) to the ROM, taking as an example David Spencer's popular Partial Renumber program from BEEBUG Vol.7 No.7.

ENTERING THE PROGRAM

Enter the program from the accompanying listing. Readers with Basic IV should substitute the listing headed EDIKIT4M for lines 3190 to 3590. If you do not have the standard Acorn Sideways RAM, substitute your own commands in lines 170 and 180 in place of *SRLOAD and *SRSAVE. Save the program as EDIKIT4. Then,

as before, load the previously assembled ROM image into a suitable sideways RAM slot, replacing the 'X' in lines 170/180 with the same slot value before proceeding. Run EDIKIT4 and reply Y to the prompt "SAVE and LOAD ROM?". The new segment of ROM will be saved as Edirom4 and the whole ROM to date as Edirom. Press Ctrl-Break and enter OLD. You are now ready to test the new commands.

USING THE NEW COMMANDS

*HELP EDIKIT will print out a list of commands. All should now work apart from SCOMM to ZCOMM which should be greeted with a "Not implemented" message. These are the spare command names which you will be able to use when you add any extra commands to the ROM.

*SYSINF (no parameters; you may abbreviate the command to *SY. or *sy.) prints out the values of PAGE, TOP, LOMEM, FREEMEM (the first available free byte above the program and variables) and HIMEM. It also prints the size of your program, the amount of memory taken up by the variables, and the amount of free memory. Finally, it prints the current screen mode

Beebug April 1990

(including modes 128 to 135 where appropriate).

*VARLIST prints out a list of the names of the program variables. If used without a parameter (you may abbreviate to just *V.) it will list all the non-zero resident integer variables (@% to Z%), together with their hex values, and the names of all the other active variables. Notice that if you edit your program, the variables will cease to be active; before you use *V. again, you will need to rerun the program. *VARLIST can produce a very long list, so you may wish to qualify it to produce only a selection of the variable names. *V.% will list the resident integer variables only, while *V.a will list only those whose names begin with "a". The full format of VARLIST is:

*VARLIST [%], [<initial letter>]

>+Help EdiKit EDIKIT 1.00 FBASIC (element) FTEXT /(string)/ FPROCFN LFROM (element) LPROC (procname) LFNC (procname) KTEXI (string1)/(string2)/ URRLST (string2)/ URRLST (string2

Users of the Master 128 are able to read the function key definitions using *SHOW but this facility is not available on the model B. *FKDEFS (*FK. will do) prints out all the function kev definitions, including any which may have been entered for keys 11 to 15 (the cursor keys and Copy). It also tells vou the status of these additional keys; whether they have been enabled as func-

List of EdiKit commands by using *HELP EDIKIT

> tion keys, to return control codes or to act in their normal editing role. These functions can be modified using *FX4. Finally, it tells you how much memory is unused in the function key buffer.

EdiKit

Listing 1

```
10 REM Program EDIKIT4
   20 REM Version B1.1
  30 REM Author Bill Hine
  40 REM BEEBUG April 1990
  50 REM Program subject to copyright
  60 :
  100 MODE 7
  110 start=&9800:size=&600:DIMcode &630
  120 PROCinitvars: PROCassemble
  130 PROClinks
  140 PRINT"SAVE and LOAD ROM? Y/N"
 150 A=GET AND&DF: IF A<>ASC"Y" THEN END
 160 OSCLI ("SAVE edirom4 "+STR$~code+"+
"+STR$~(0%-code))
 170 *SRLOAD edirom4 9800 X
 180 *SRSAVE edirom 8000 9E30 X
 190 PRINT"Press Ctl-Break to initialis
e"
 200 END
 210 :
1000 DEF PROCinitvars
1010 page=&18:cmdline=&F2:newline=&OD
1020 fktable=&B00
1030 oswrch=&FFEE:osnew1=&FFE7
1040 osbyte=&FFF4:osword=&FFF1
1050 osrdch=&FFE0:osrdrm=&FFB9
1060 romexit=&8803:escexit=&8806
1070 prnttext=&8815:notimp=&8818
1080 key=&70:byte=&71:kptr=&72:max=&74
1090 x=&75:kend=&76:lo=&78:hi=&79
1100 flag=&7A:vptr=&80
1110 ENDPROC
1120 :
1130 DEF PROCassemble
1140 FOR pass=4 TO 7 STEP 3
1150 P%=start:0%=code:[OPT pass
1160 .calltable
1170 JMP sysinf: JMP varlist: JMP fkdefs
1180 .pcomm JMP & 9E00
1190 .qcomm JMP &9E03
1200 .rcomm JMP & 9E06
1210 .scomm JMP &9E09
1220 .tcomm JMP &9E0C
1230 .ucomm JMP & 9EOF
1240 .vcomm JMP & 9E12
1250 .wcomm JMP & 9E15
1260 .xcomm JMP &9E18
1270 .ycomm JMP &9E1B
1280 .zcomm JMP &9E1E
1290 :
1300 .sysinf
1310 JSR prnttext
1320 EQUS "Values": EQUB 0
1330 LDX#14:JSR spaces:JSR prnttext
1340 EQUS "Sizes": EQUW newline
1350 JSR prnttext: EQUS "-----": EQUB 0
```

1360 LDX#14:JSR spaces:JSR prnttext 1370 EQUS "----":EQUW newline 1380 JSR prnttext: EQUS "PAGE" 1390 EQUD & 20202020: EQUB 0 1400 LDA page: JSR prnthex 1410 LDA#0:JSR prnthex:JSR osnewl 1420 JSR prnttext: EQUS "TOP" 1430 EQUD & 20202020: EQUW & 0020 1440 LDA &13:JSR prnthex 1450 LDA &12:JSR prnthex 1460 LDX#8:JSR spaces:JSR prnttext 1470 EQUS "Prog": EQUW &0020 1480 LDA &13:SEC:SBC page:JSR prnthex 1490 LDA &12: JSR prnthex: JSR osnewl 1500 JSR prnttext: EQUS "LOMEM" 1510 EQUD &00202020 1520 LDA 1:JSR prnthex 1530 LDA 0:JSR prnthex:JSR osnewl 1540 JSR prnttext: EQUS "FREEMEM" 1550 EQUW &0020 1560 LDA 3: JSR prnthex 1570 LDA 2: JSR prnthex 1580 LDX#8:JSR spaces:JSR prnttext 1590 EQUS "Vars": EQUW &0020 1600 LDA 2:SEC:SBC 0:PHA 1610 LDA 3:SBC 1:JSR prnthex:PLA 1620 JSR prnthex: JSR osnewl 1630 JSR prnttext: EQUS "HIMEM" 1640 EQUD &00202020 1650 LDA 7: JSR prnthex 1660 LDA 6: JSR prnthex 1670 LDX#8:JSR spaces:JSR prnttext 1680 EQUS "Free": EQUW &0020 1690 LDA 4:SEC:SBC 2:PHA 1700 LDA 5:SBC 3:JSR prnthex:PLA 1710 JSR prnthex: JSR osnewl 1720 JSR prnttext: EQUS "Screen Mode" 1730 EQUW &0020:LDA#&75:JSR osbyte 1740 TXA:AND#16:ASLA:ASLA:ASLA 1750 CLC:ADC &355:STA lo:LDA#0:STA hi 1760 JSR prntdec: JSR osnewl: JMP romexit 1770 .prnthex PHA 1780 LSRA:LSRA:LSRA:LSRA:JSR pnib:PLA 1790 AND#&F:JSR pnib:RTS 1800 .pnib CMP#10:BCC pnib2:CLC:ADC#7 1810 .pnib2 1820 CLC:ADC#&30:JSR oswrch:RTS 1830 .prntdec 1840 LDY#4:LDA#0:STA flag 1850 .digloop LDX#ASC"0" 1860 .subloop 1870 SEC:LDA lo:SBC powers, Y:PHA 1880 LDA hi:SBC powers+1, Y:BCC prntdig 1890 STA hi:PLA:STA lo:INX:BCS subloop 1900 .prntdig 1910 PLA: TXA: CMP #ASC" 0": BEQ zero 1920 JSR oswrch: INC flag: JMP nextdigit

EdiKit [°]

1930 .zero TYA:BEQ prntzero 1940 LDA flag: BEQ nextdigit 1950 .prntzero TXA:JSR oswrch 1960 .nextdigit 1970 DEY: DEY: BPL digloop: RTS 1980 TXA: JSR oswrch 1990 .powers EQUW1:EQUW10:EQUW100 2000 .varlist 2010 JSR varlist2: JMP romexit 2020 .varlist2 2030 JSR skspc:LDA(cmdline),Y 2040 CMP#newline:BEQ allvars 2050 TAX: CMP #ASC"%": BNE varlist3 2060 JMP resvars 2070 .varlist3 2080 CMP#ASC"@":BNE varlist4:JMP resvx 2090 .varlist4 2100 CMP#ASC"A":BCC error 2110 CMP#ASC"Z"+1:BCC varx 2120 CMP#ASC"a":BCC error 2130 CMP#ASC"z"+1:BCC varx 2140 .error 2150 LDA#0:STA &100:LDA#105:STA &101 2160 LDX#0 2170 .errlp 2180 LDA rprt, X:STA &102, X 2190 INX:CMP#0:BNE errlp:JMP &100 2200 .rprt 2210 EQUS " Invalid parameter": EQUB 0 2220 .skspc 2230 LDA(cmdline), Y: INY 2240 CMP#ASC" ":BEQ skspc:DEY:RTS 2250 .allvars 2260 LDA#ASC"@":PHA 2270 TAX: JSR resvx: PLA: TAX 2280 .allvloop 2290 INX: JSR varx 2300 TAX:CPX#ASC"z":BNE allvloop 2310 RTS 2320 .varx 2330 TXA:PHA:PHA:JSR resvx:PLA 2340 ASLA: TAY: LDA &400, Y: STA vptr 2350 LDA &401, Y:BEQ vxexit2:STA vptr+1 2360 .varx2 2370 PLA:PHA:JSR oswrch:LDY#2 2380 .varxlp 2390 LDA (vptr), Y:BEQ nextv:CMP#ASC"(" 2400 BEQ ptarray: JSR oswrch: INY 2410 JMP varxlp 2420 .ptarray 2430 JSR prnttext: EQUS " (array)" 2440 EQUB 0 2450 .nextv 2460 JSR osnewl:BIT &FF:BMI vesc 2470 LDY#0:LDA (vptr), Y:PHA 2480 INY:LDA (vptr), Y:BEQ vxexit 2490 STA vptr+1:PLA

2500 STA vptr: JMP varx2 2510 .vxexit PLA 2520 .vxexit2 PLA:RTS 2530 .vesc JMP escexit 2540 .resvars LDA#ASC"@"-1 2550 .resvlp 2560 BIT &FF:BMI vesc:TAX:INX:TXA 2570 PHA: JSR resvx: PLA 2580 CMP#ASC"Z":BNE resvlp:RTS 2590 .resvx 2600 CPX#ASC"Z"+1:BCS vxexit2+1 2610 TXA: PHA: INX: TXA: ASLA: ASLA 2620 TAY:DEY:STY &70 2630 LDA &400, Y:BNE prntresv 2640 DEY:LDA &400,Y:BNE prntresv 2650 DEY:LDA &400,Y:BNE prntresv 2660 DEY:LDA &400,Y:BEQ vxexit2 2670 .prntresv 2680 PLA: JSR oswrch 2690 LDA#ASC"%": JSR oswrch 2700 LDA#ASC" ": JSR oswrch 2710 LDX#4:LDY &70 2720 .prntresvlp 2730 LDA &400, Y: JSR prnthex 2740 DEY:DEX:BNE prntresvlp 2750 JSR osnewl:RTS 2760 .fkdefs 2770 JSR initend: JSR ptfktitle: LDX#0 2780 .fkeyloop 2790 STX key: JSR ptkeyno: JSR finddef 2800 LDX key: INX: CPX#16: BNE fkeyloop 2810 JMP romexit 2820 .ptfktitle 2830 JSR prnttext 2840 EQUS"FUNCTION KEYS - bytes free: " 2850 EQUB 0:JSR prntfree 2860 LDA#&ED:LDY#&FF:LDX#0:JSR osbyte 2870 TXA: BEQ pfkt0: AND #1: BNE pfkt1 2880 JMP pfkt2 2890 .pfkt0 JSR prnttext 2900 EQUS " Keys 11-15 enabled for curs or editing" 2910 EQUB newline: EQUW newline: RTS 2920 .pfkt1 JSR prnttext 2930 EQUS " Keys 11-15 return control c odes" 2940 EQUB newline: EQUW newline: RTS 2950 .pfkt2 JSR prnttext 2960 EQUS " Keys 11-15 enabled as funct ion keys" 2970 EQUB newline: EQUW newline: RTS 2980 .ptkeyno 2990 TXA:CMP#10:BCS pk10:JSR ptkey 3000 CLC:ADC#ASC"0":JSR oswrch 3010 .pkn2 3020 LDA#ASC" ":JSR oswrch 3030 LDA#ASC"""":JSR oswrch:RTS

EdiKit

```
3040 \
 3050 .ptkey PHA:LDA#ASC"K":JSR oswrch
 3060 LDA#ASC".": JSR oswrch: PLA: RTS
 3070 .pk10
 3080 LDA key:SEC:SBC#10:LDX#0
 3090 .pk101p
 3100 CMP#0:BEQ pk10exit:PHA
 3110 TXA:CLC:ADC#8:TAX:PLA
 3120 SEC:SBC#1:JMP pk101p
 3130 .pk10exit LDY#&F8
3140 .pk10plp
3150 LDA pktxt, X: JSR oswrch: INY: INX
 3160 CPY#0:BNE pk10plp:JMP pkn2
 3170 .pktxt
3180 EQUS "BRK K.10CPY K.11 < K.12 >
K.13 v K.14 ^ K.15"
3190 .initend
3200 LDA &B10:STA kend:RTS
3210 .prntfree
3220 LDA#&FF:SEC:SBC kend:STA lo
3230 LDA#0:STA hi:JSR prntdec:JMPosnewl
3240 :
3250 .finddef LDX key
3260 LDA fktable, X:CMP kend: BEQ klpexit
3270 STA kptr:LDA#&FF:STA max:LDX#0
3280 .kloop
3290 LDA fktable, X
3300 CPX key:BEQ nextk
3310 CMP kptr:BCC nextk
3320 CMP max: BCS klp2:STA max
3330 .klp2
3340 CMP kptr:BNE nextk:LDA#&FF:STA max
3350 .nextk
3360 INX:CPX#17:BNE kloop:JSR prntkeyde
3370 .klpexit
3380 LDA#ASC""""
3390 JSR oswrch: JSR osnewl
3400 RTS
3410 .prntkeydef
3420 LDX kptr
3430 .pkdloop
3440 INX:LDA fktable, X:STA byte
3450 CMP#ASC"""":BNE pkdlp2
3460 LDA#ASC"""":JSR oswrch
3470 .pkdlp2
3480 CMP#&81:BCC pkdlp3
3490 LDA#ASC"|":JSR oswrch
3500 LDA#ASC"!": JSR oswrch
3510 SEC:LDA byte:SBC#&80:STA byte
3520 .pkdlp3
3530 CMP#&20:BCS pkdlp4
3540 LDA#ASC"|":JSR oswrch
3550 LDA#&40:CLC:ADC byte:STA byte
3560 .pkdlp4
3570 LDA byte: JSR oswrch
3580 CPX max: BNE pkdloop
```

3590	RTS
3600	:
3610	.spaces LDA#ASC" "
3620	.spcloop
3630	JSR oswrch:DEX:BNE spcloop:RTS
3640]NEXT:ENDPROC
3650	:
3660	DEF PROClinks
3670	P%=start+size:0%=code+size
3680	FOR 1%=0 TO 10
3690	[OPT7:JMP notimp:]
3700	NEXT·ENDPROC

Listing 2

```
10 REM Program EDIKIT4M
  20 REM Version B1.00
3190 .initend
3200 LDA#&10:LDX#0:JSR getptr:STA kend
3210 LDA#&21:LDX#0:JSR getptr:STAkend+1
3220 RTS
3230 .getptr
3240 STXx:STA&F6:LDA#&80:STA&F7:TXA:CLC
3250 ADC&F6:STA&F6:LDA#0:ADC&F7:STA&F7
3260 LDY#&80:JSR osrdrm:LDXx:RTS
3270 .prntfree
3280 LDA#0:SEC:SBC kend:STA lo
3290 LDA#&84:SBC kend+1:STA hi
3300 JSR prntdec: JMP osnewl
3310 :
3320 .finddef
3330 LDX key:LDA#0:JSR getptr:STA kptr
3340 LDX key:LDA#&11:JSR getptr
3350 STA kptr+1
3360 LDX key: INX: LDA#0: JSR getptr
3370 SEC:SBC kptr:STA max:BEO fdefexit
3380 JSR prntkeydef
3390 .fdefexit
3400 LDA#ASC"""":JSR oswrch:JSR osnewl
3410 RTS
3420 .prntkeydef LDY#1
3430 .pkdloop
3440 TYA:PHA:LDAkptr:STA&F6:LDAkptr+1
3450 STA &F7:LDY#&80:JSR osrdrm
3460 STA byte:CMP#ASC"""":BNE pkdlp2
3470 LDA#ASC"""":JSR oswrch
3480 .pkdlp2
3490 CMP#&81:BCC pkdlp3:LDA#ASC"|"
3500 JSR oswrch:LDA#ASC"!":JSR oswrch
3510 SEC:LDA byte:SBC#&80:STA byte
3520 .pkdlp3:CMP#&20:BCS pkdlp4
3530 LDA#ASC"|":JSR oswrch
3540 LDA#&40:CLC:ADC byte:STA byte
3550 .pkdlp4
3560 LDA byte: JSR oswrch: PLA: TAY: CPYmax
3570 BEQ pkdloopexit: INY: CLC: INC kptr
3580 BNEpkdloop:INCkptr+1:JMP pkdloop
3590 .pkdloopexit RTS
```

Music To Your Ears

Alan Wrigley presents a round-up of some recently-released music discs for the Music 5000.

Product	Music Discs
Supplier	Hybrid Technology Ltd 273 The Science Park
	Cambridge CB4 4WE
Price	£4.95 each inc. VAT

Moments in Time by The Noige, from Panda, has a rather different feel to it, paying less attention to melody but focussing instead upon rhythm and sound texture. Unfortunately this tended to make many of the tracks sound very similar,

If you have read Ian Waugh's article on Ample programming elsewhere in this issue, but do not yet feel proficient enough to create your own ProductMusic DiscsSupplierPanda DiscsFour Seasons, TinkersLane, Brewood, StaffordST19 9DEPrice\$5.00 each inclusive

though if you like music which inspires you to tap your feet, then this is no great disadvantage. There are some good examples of rhythmic tracks, with a steady beat and swirling synthesisers.

MUSIC FOR MONSTERS

masterworks, you may be interested to know that there are a number of music discs available featuring the work of experienced Ample programmers. All the discs reviewed here are designed to be played only with Hybrid's Music 5000 system, and if you have not heard any before, you might be quite astounded at the range of sounds which can be coaxed out of this excellent little system.

In this review I shall be looking at seven discs, two of them from Hybrid itself, and five from Panda discs, which originate from the Ample DCT service at Dudley College of Technology. I should point out at the start that I have no experience of creating computer music myself, and I therefore approached this review from the viewpoint of a music lover rather than a programmer. Music is, of course, an extremely subjective topic in any case, and it is impossible for a reviewer to say that a piece of music is "good" or "bad", merely that it does not reflect his or her tastes.

The first disc I listened to was *Shivering Again* by Michael Harbour, from Hybrid. This has quite a distinctive feel to it, sounding at times a little like Mike Oldfield, though none the worse for that. Musically it is unpretentious, with simple but catchy melodies, but the instrumentation is imaginative, and the disc as a whole is a nice blend of serious and novelty items. If you like pleasant music which is immediately appealing, this disc is an excellent introduction to computer music.

Beebug April 1990

delightful disc from Panda called The Monster Computations Nursery Rhyme Songbook, aimed at "little monsters everywhere". This is a compilation of nursery rhymes, imaginatively scored and each complete with a screen display of the words, with graphics thrown in for good measure in some cases. The disc contains some 26 children's favourites, such as Ding Dong Bell, Looby Loo, Frère Jacques, and so on. Amongst the contributors are Rupert and Oliver Dudley, 6 and 8 years old respectively. If you have young children and a Music 5000, then this disc is an absolute must. It's just the thing to get the family together around the computer, or to provide a welcome diversion from flying ice cream at children's parties. Hard pressed infant teachers, too, will no doubt find that it makes music lessons more bearable.

At the other end of the musical spectrum is a

Panda's Jean-Michel Jarre Special, as you might expect, is based on a selection of works by the master of synthesized music himself, programmed by Bernie Dawson. The complete *Rendezvous* suite is included, together with selections from *Oxygene*, *Equinoxe*, and *Magnetic Fields*. No-one would expect the Music 5000, with its limited number of channels, to be able to reproduce the sheer breadth of sound of the originals, but Bernie Dawson has done an excellent job in coaxing the maximum performance out of the system. This disc does full justice to the music, given the limitations

Music To Your Ears

mentioned above, and J-M J fans will not be disappointed.

Also from Panda comes the Children in Need Compilation. This contains a wide variety of music of all kinds, and all the profits and royalties will be donated to the Children in Need charity. The disc kicks off with a catchy title track to which appropriate on-screen lyrics have been provided. There are songs for children, such as Nellie the Elephant, some very good renditions of pop songs, for example Chris de Burgh's Lady in Red, classicallyinspired pieces, TV music, Christmas carols, and some tracks by artists featured elsewhere in this review, such as The Noige and Phil Comber. The range is excellent and there is something for everyone. In addition, of course, buying this disc supports a very worthy cause.

The last Panda disc is *Deux* by Phil Comber. This is heavily dependent on rhythm, but unlike the bright and breezy sound of The Noige, the music on this disc has an earthy feel to it. The insistent rhythms are also very catchy, and the disc as a whole grew on me on subsequent hearings. It will definitely bear repeated listening, and will have you tapping your feet all the way through.

The best things are usually better left until last, and *Contrast* by Pilgrim Beart, from Hybrid, is no exception. Inventive, often introspective and brooding, not always predictable, this music is, to my ears at least, in a class apart from the other discs. It is full of rich texture and occasional surprises, and gives me the impression of being produced by a composer rather than a musician. It is by far the best example among this selection of the limits to which the system can be pushed.

CONCLUSIONS

All in all, I was most impressed by my first introduction to computer music. All the discs I have reviewed are worth hearing, and I look forward to listening to some more. For BEEBUG readers who have a Music 5000, Panda and Hybrid have kindly allowed us to include samples on this month's magazine disc. *MITdemo* is a collection of themes from *Moments in Time, For a Dying Man* is from *Shivering Again,* and *Tour de Farce* is taken from *Contrast.*

A Datafile Search Utility (continued from page 27)

1510 .intout:LDA #ASC"I":JSR &FFE3 1520 JSR col:LDX #3:LDA #&40 1530 .loop JSR bget:BCS err 1540 STA &2A, X:DEX:BPL loop 1550 .numcr JSR numout:CLC 1560 .endout PHP:JSR &FFE7:LDY y2:PLP 1570 RTS 1580 .realout:LDA #ASC"R":JSR &FFE3 1590 JSR col: JSR bget: BCS err: STA & 34 1600 JSR bget: BCS err: STA &33 1610 JSR bget: BCS err: STA &32 1620 JSR bget: BCS err: LDY #0: CMP #0 1630 BPL ov10:LDY #&80 1640 .ov10 STY &2E:ORA #&80:STA &31 1650 JSR bget: BCS err: STA &30 1660 JMP numcr 1670 .err LDA #63:JSR &FFE3:JSR &FFE3 1680 SEC: JMP endout 1690 .numout TYA:PHA:LDY #0:STY &15 1700 LDA basicrom:STA &FE30 1710 LDY type: JSR cntos: LDA &F4 1720 STA &FE30:LDX #0 1730 .loop LDA &600, X: JSR &FFE3: INX 1740 CPX &36:BNE loop:PLA:TAY:RTS 1750 .uc BIT E:BMI retn 1760 CMP #ASC"a":BCC retn 1770 CMP #ASC"z"+1:BCS retn:AND #&DF

1780 .retn RTS 1790 .skipsp LDA (&F2),Y:CMP #32 1800 BNE retn: INY: JMP skipsp 1810 .decout:LDA &400:PHA:STX &400 1820 LDX #1:.loop LDA &400,X:PHA 1830 LDA #0:STA &400,X:INX:CPX #4 1840 BNE 100p 1850 LDA #&40:STA type:LDA #1:STA &403 1860 JSR numout 1870 .loop LDA #32:JSR &FFE3:INX 1880 CPX &400:BCC loop:LDX #3 1890 .loop PLA:STA &400, X:DEX:BPL loop 1900 RTS 1910 .msg1 EQUS msg1\$ 1920 .msg2 EQUS msg2\$ 1930] 1940 NEXT 1950 c\$="SAVE DFIND "+STR\$~HIMEM+" "+ST R\$~0%+" "+STR\$~R%+" "+STR\$~R% 1960 PRINTC\$ 1970 OSCLI C\$ 1980 END 1990 : 3000 DEFFNcopy3(a,b) 3010 [OPT opt 3020 LDA a:STA b:LDA a+1:STA b+1 3030 LDA a+2:STA b+2:]:=""



512 Forum

by Robin Burton

After the last two Forums which were pretty easy going,

it's time to move on to more pithy matters.

The next three Forums are the direct result of reader's letters. By the way, I haven't said it for quite a while, but if there's something you'd like to see in the Forum, or a view you wish to air, write to me and say so!

The first topic is for the benefit of those who know little or nothing about the 512, while next month's Forum is for those of you who are fairly experienced and a bit adventurous too. Read on if you want to know more.

THE INCOGNIZANT

The first letter was from Graham Lowe, who says he reads the Forum every month even though he doesn't own a 512. This of course is very gratifying, but the main point of his letter is that while he finds many topics fascinating (his word, not mine), they would be even more so if he knew more about the 512. Specifically, 'what exactly is it and what does it do'. The other point he makes is that he might well be tempted to buy a 512 if he knew more about it. Could I therefore spare a bit of space in the Forum to explain?

Although many 512 users will know most of this, I thought Graham's letter made a reasonable request. It's easy to overlook if you're not in this category, but there may be many BBC micro users who know of the 512 but who, if they have no dealings with PCs in everyday life aren't likely to know much else. In that situation they're not likely to buy a 512, even secondhand, because they can't appreciate the machine's merits and capabilities.

Perhaps you knew and had used CP/M and 86 series PC machines when you purchased your 512. Maybe you found a dealer who was keen

Beebug April 1990

to make a sale and was prepared to demonstrate. Well, of course it's easy then, but Graham's letter made me consider the fact that not everyone automatically falls into the first category, while these days it's unlikely that anyone falls into the second. As I said, I think it's a perfectly reasonable request, so here goes.

This month's Forum is aimed at those who know little or nothing about the history of DOS and PCs. It is, I hope, entertaining for all readers, but above all I hope it's a reasonable insight into why the 512 exists and what it provides for those who may perhaps yet become 512 users.

IN THE BEGINNING

We all know that home micros come in many shapes and sizes and from various manufacturers. In the main, apart from particular cases like the BBC and the Z88 portable, most of these machines have little or nothing in common in terms of operation, facilities, commands, disc formats, program code and so on. Of course all these machines work in broadly similar ways, but nothing is directly compatible with anything else.

For home micros this is acceptable, as virtually every user is isolated and, apart from BBC micros, the majority of machines are used to play games, but it's a far from ideal situation in business.

Digital Research recognised this problem very early on in the micro story which began in the 1970s. They therefore designed an operating system to overcome this limit to growth. During that decade, as processor power began to grow to useful levels, Digital Research virtually cornered the business micro operating system market with something called CP/M (CP/M by the way stands for 'Control Program for Microcomputers' - how's that for imagination?).

512 Forum

CP/M was 'designed from the ground up' with the objective of becoming the 'de-facto' standard business operating system for microcomputers (they weren't called PCs in those days). To achieve this, Digital knew that a revolutionary modular approach to system design was required, especially if the system was to be capable of growth as well as running on almost all machines without throwing everything away and starting again each time.

The crux of the problem was that, although CP/M would run in an 8 bit Zilog Z80 or Intel 8080 based machine (because these were the two processors of the day) various parts of the hardware in different manufacturer's micros (e.g. the discs, the filing system controller, the screen and keyboard) were still different, just like home micros are now.

However, the deliberate modular design of CP/M meant that only one part of the operating system needed changing to allow it to run on a new manufacturer's hardware, specifically that part concerned with basic input/output functions. This module is called the BIOS (Basic Input Output System - more imagination at work here). In fact to make it as easy as possible for manufacturers, Digital supplied their operating system to hardware manufacturers in a sort of kit form.

The parts of the BIOS which control the peripherals were therefore customised by the manufacturer to suit the particular hardware devices employed, while the rest of the operating system remained fixed. Each manufacturer provided only the peripheral handling code for the BIOS and could then very simply generate a complete new CP/M operating system with the tools supplied.

A SUCCESSFUL CONCEPT

Software houses were naturally quick to see the opportunities this approach offered. It meant they could write an application for any type of CP/M machine and it would work in virtually all of them immediately. This was because the application's interface to the operating system no longer changed with the hardware, only the input/output part of the operating system which drove peripherals did. Of course applications (should) only communicate with the operating system by legal calls, and if they did compatibility was virtually guaranteed.

The result was that software wasn't tied to specific makes of hardware, only to an operating system which could be customised to suit almost any machine of the day. The converse was that hardware manufacturers felt obliged to support CP/M if their machines were to be capable of running this ever growing pool of business software.

Business users were keen on the idea too. Now they weren't tied to a single type of machine or to a single hardware or software supplier. Any CP/M machine could run any CP/M program in the same way with the same user interface regardless of who manufactured the micro and who wrote the software. The personal business computer (as opposed to the anonymous mainframe which 'belonged' to no individual) was created, or perhaps more accurately it was inevitable.

At the beginning of the eighties 16 bit processors appeared commercially, which allowed more memory to be provided on 'small' micros at less cost (8 bit processors, like the BBC's 6502 or the 80 series can only address 64K bytes, while 16 bit processors can address up to 1Mbyte). For these new processors a new operating system was needed of course, and with the huge success of CP/M proving that the design was absolutely correct it was repeated for these new processors.

In the event the new 'standard' operating system has turned out to be DOS rather than CP/M(86). Essentially, after the brilliance of CP/M, Digital sat on its laurels a bit too long and missed the opportunity of a single-handed encore with the new processors. Even so, if you investigate a little you'll find that all versions of DOS, including MS-DOS and PC-DOS are direct descendants of CP/M with precisely the same internal design concepts as the original, even to the point where many operating system calls are exactly the same.

THE PC

IBM was keen to get into this now rapidly expanding market and so with the arrival of 16 bit processors they produced the IBM Personal Computer, but instead of selling hardware they marketed a concept - the PC. They originally intended to offer a choice of at least two operating systems with their new PCs, but as Digital were a bit slow this time it didn't happen. Instead only one, DOS, produced by a then small, almost unknown software house called the Microsoft Corporation was offered, though in a modified form to suit IBM hardware. This was, and still is, PC-DOS.

Other manufacturers began to produce PCs, also using DOS, this time the original Microsoft offering, so a second version of DOS was assured. This one, with the by now expected flair, is called MS-DOS. You can work the name out for yourself.

Right up to the present, IBM uses PC-DOS and all PC clones (everything except IBM machines) use MS-DOS. In fact apart from the hardware there's almost no difference between them. Whenever a new version of one appears the other follows almost at once, so they keep facilities in step and even use the same version numbers. Digital Research, though late into this battle, also produced their own version of DOS, which they called DOS Plus, though the current version is now called DR-DOS.

Just like the original design of CP/M, all versions of DOS operate in the same way in all machines as far as both the user and the application is concerned. The BIOS is customised to suit the hardware, while the rest of the operating system and its facilities remains fixed. If code is legally written it will run in any DOS machine.

This more or less brings us up to date. DOS continues to develop, but more slowly now than in its first few years, and over the (nearly)

ten years of its existence as the standard PC operating system, literally tens if not hundreds of thousands of programs have been written covering just about every conceivable application.

THE 512 CO-PROCESSOR

So where does the 512 fit in? Simple - it allows BBC users to run a version of DOS which, apart from some difficulties with programs that aren't strictly legal (caused by unavoidable differences like the two processor configuration) extends the same facilities to Acorn users as are enjoyed by PC users, without the need for the considerable expense of a 'real' PC.

Remember that when the 512 appeared Amstrad hadn't entered the market, so the 'cheap' £500 PC didn't exist. In the early 80s even the most meagre PC would cost upwards of £3000. Things have changed somewhat in the last few years, but even now a PC to match the 512's performance will cost around £1000 and quite possibly more unless you import it from Taiwan yourself. At its original price the 512 was a snip. Now second-hand prices seem to be increasing, reflecting the almost cult following the 512 has developed. That said you can still buy a 512 for less than (just) the price of a second-hand model B, but look at what the 512 can do.

It's true a few business machines don't use DOS even now, the Apple Macintosh for example, and in larger systems UNIX tends to be used. Even so, it's almost certain that, worldwide, PCs using DOS outnumber all other types of computer put together. I don't know how many PCs are estimated to be in use, but it's certainly tens of millions.

This explains the attraction of the 512. For a modest outlay the BBC micro owner can tap into the biggest pool of software the world has ever seen, with enough memory and enough processor power to tackle jobs that the BBC micro, excellent though it is, can't even think about (and you don't even have to give up your BBC micro to do it).

Improve Your Master (and Beeb!)

Andrew Rowland offers two programs to upgrade the Master 128, one of which can also be used with the model B.

Acorn has at last brought out the improved version of the Master's one Megabit chip, but at a price! If what's left of your money won't run to the full upgrade, these routines will provide you with two of the improvements.

Firstly there is a remedy for that *CLOSE bug to which Derek Gibbons recently drew attention in Vol.8 No.6. Listing 1 creates a utility called DFSfix which completely cures the bug in the Master 128's DFS. It is installed, by typing *DFSfix, in private RAM at &DD00, which is theoretically reserved for transient utilities, but as I find that all my transient star commands run at &900, &DD00 is a safer bet. Once installed, it remains active until the machine is switched off, surviving any change of filing system, or hard or soft Break. It does not object to another filing system being used, or to being called twice by accident.

As Derek Gibbons covered the theory in detail, I will say only that it is a machine code equivalent of his Basic program. When it detects that a *CLOSE or CLOSE#0 has been issued, it tries to close each of the five possible file handles in turn, trapping the Channel error that will occur when a handle does not relate to an open file.

MORE CHARACTERS

The new Megabit chip offers improved support for international characters, which in practice means configuring the keyboard like that of the Compact. If you haven't seen one, the familiar @ key is replaced by one with a strange symbol dubbed 'code'. This works a bit like the Alt key on some other computers, providing an alternative font. However, it doesn't function in the same way as the Shift and Ctrl keys: you press the code key first, then another key, and thus obtain one of the characters from the Master's international font (codes 128 to 255). The character you get is the ASCII code of the second keypress plus 128, and includes accented characters, a Greek font, mathematical symbols and characters for creating forms with lines and boxes. The @ symbol itself is obtained by pressing Shift and zero together.

Listing 2 will create a program called Alt which emulates the Compact's keyboard on a Master 128, and on a model B with the extra characters defined - see below. Once the machine code generated has been saved to disc it can be installed by typing *Alt, which installs the code at &A00. Avoid doing this twice without pressing Break in between.

Once installed, it gives access to the Master's extended font direct from the keyboard. To see what is available, select any screen mode except 7 and type:

FOR 1%=128 TO 255:P.;A%" "CHR\$A%:N.

USING THE EXTRA CHARACTERS

You will find Acorn planned its font carefully, so the combinations are easy to remember. Thus @ (or rather, *code*) followed by 'a' gives a lower case alpha, @ A an upper case alpha, and so on. Don't forget to try it with Ctrl as well: @ followed by Ctrl G gives the copyright symbol, for example.

You will have to learn which keys produce the characters you need, and you should be aware that many word processors won't accept the extended font - such as View, for instance. The Master's Edit is quite happy to take them though, but some will be regarded as function or cursor keys, so be careful. It is not usually possible to print the extra characters, but last month's magazine contained an article (*Printing Characters 128 to 255*) which described how to download the complete extended font to an Epson printer.

HOW IT WORKS

Let's briefly consider what happens when you press a key. After the computer has worked out which key you pressed and has taken into account the state of the Shift and Ctrl keys, Caps Lock etc., it puts the ASCII code for the key into the keyboard buffer. Here it waits in a queue until needed, so you can type ahead. When a program requires some input, it calls the operating system routine OSRDCH, which takes the next character from the buffer, performs any function key expansion required, handles cursor copying if active and returns the character to the program which requested it. The Basic keyword GET uses this routine, as does INPUT.

To implement the *code* key, we want to look out for an @, get the next character from the buffer and add 128 to that before returning.

This is achieved by the part of the program beginning at line 350. It starts by calling the real OSRDCH routine, looks to see if the character returned is 250 (more on that in a moment), and if it is, calls OSRDCH again and adds 128 to the next character returned, which will be the key you pressed after the @key. This action must be carried out *after* the characters have been handled by OSRDCH to avoid our alternative characters being treated as function or cursor keys.

Secondly, a new @ key must be provided as Shift zero. This action, on the other hand, needs to be done as early as possible, for if we waited until the character is removed from the buffer, the Shift key may no longer be held down. We therefore intercept the routine which puts characters into the keyboard buffer and check for the ASCII code for zero (line 250). If one is found, we look at the keyboard status byte which tells us if. Shift is pressed or Shift Lock engaged, and if necessary substitute an @. We then return control to the operating system to insert the code into the buffer.

A problem arises here: if an @ is removed from the buffer, is it a Shift zero or a real @; should it be left as it is or treated as the *code* key? The answer is to change any real @'s into a character 250 (hence the reference to this character above) so there is no confusion (there is a tendency for some codes to be interpreted as cursor keys, but 250 appears to work correctly).

The first part of the program, *install*, diverts the two OS vectors INSV and RDCHV to *entry* and *rdch* respectively.

Beebug April 1990

WHAT ABOUT THE MODEL B?

Last month's article, mentioned above, also described how to provide some or all of the Master's extended character set on the model B. The disadvantage of the method described was that, in order to define more than 32 extra characters, the value of PAGE had to be raised. On this month's disc, however, you will find a routine, Mfont, which will install the full Master font on a model B, using memory from &1300 to &16FF. This area is normally DFS workspace, but should be usable provided that you do not have more than one DFS file open at any one time. To install the set, just type *Mfont, and the characters can then be accessed as described for the Master above.

One final tip: programmers will find the extended keyboard useful for inserting teletext control codes into listings, and even as a way of typing Basic keywords with only two keypresses. Try typing a line of Basic and include @b. When you list the program, ENVELOPE appears as if by magic. Happy extended typing!

Listing 1

10	REM Program DFSfixB
20	REM Version B1.00
30	REM Author Andrew Rowland
40	REM BEEBUG April 1990
50	REM Program subject to copyright
60	:
100	osbyte=&FFF4:osargs=&FFDA
110	findv=&21C:bytev=&20A
120	orkv=&202:channelerr=222:dfs=4
130	2%=&DD00
140	FOR pass = 0 TO 3 STEP 3:P%=Q%
150	[OPT pass
160	LDA #247:LDX #&4C
170	LDY #0:JSR osbyte
180	DA #248:LDX #break MOD 256
190	DY #0:JSR osbyte
200	DA #249:LDX #break DIV 256
210	DY #0:JSR osbyte
220	JSR vectors:RTS
230	break
240	BCS vect2:RTS
250	vectors
260	JDA findv
270	CMP #code MOD 256:BEQ vect2
280	STA oldfindv:LDA findv+1
290	STA oldfindv+1:SEI

Improve Your Master (and Beeb!)

```
300 LDA #code MOD 256:STA findv
310 LDA #code DIV 256:STA findv+1
320 .vect2
330 LDA bytev
340 CMP #newosbyte MOD 256
350 BEQ changed
360 STA oldbytev:LDA bytev+1
370 STA oldbytev+1:SEI
380 LDA #newosbyte MOD 256
390 STA bytev
400 LDA #newosbyte DIV 256
410 STA bytev+1
420 .changed
430 CLI:RTS
440 .newosbyte
450 CMP #&8F:BNE byteout
460 CPX #&F:BNE byteout
470 PHA: PHX: PHY: JSR vectors
480 PLY:PLX:PLA
490 .byteout
500 JMP (oldbytev)
510 .code
520 CMP #0:BNE notme
530 CPY #0:BNE notme
540 PHA:PHX:PHY:LDY #0:TYA
550 JSR osargs
560 CMP #dfs:BNE exit
570 TSX:STX stack
580 LDA brkv:STA oldbrkv
590 LDA brkv+1:STA oldbrkv+1
600'LDA #err MOD &100:STA brkv
610 LDA #err DIV &100:STA brkv+1
620 LDA #119:JSR osbyte
630 LDY #&11:STY channel
640 .loop
650 LDY channel:LDA #0:JSR notme
660 .rentry
670 INC channel:LDY channel
680 CPY #&16:BNE loop
690 LDA oldbrkv:STA brkv
700 LDA oldbrkv+1:STA brkv+1
710 PLY:PLX:PLA:RTS
720 .exit
730 PLY:PLX:PLA
740 .notme
750 JMP (oldfindv)
760 .err
770 PHA:PHY:LDY #0:LDA (&FD),Y
780 CMP #channelerr:BNE errexit
790 LDX stack: TXS: JMP rentry
800 .errexit
810 LDA oldbrkv:STA brkv
820 LDA oldbrkv+1:STA brkv+1
830 PLY:PLA:JMP (oldbrkv)
840 .end
850 .oldbrkv EQUW 0
```

```
860 .oldfindv EQUW 0
870 .oldbytev EQUW 0
880 .stack EQUB 0
890 .channel EQUB 0
900 ]
910 NEXT
920 PRINT'"Code ends &";~P%'
930 a$="SAVE DFSfix "+STR$~Q%+" "+STR$-
~(end)
940 PRINT a$:OSCLI a$
```

Listing 2

```
10 REM Program AltB
  20 REM Version B1.00
  30 REM Author Andrew Rowland
  40 REM BEEBUG April 1990
  50 REM Program subject to copyright
  60 :
 100 INSV=&22A:RDCHV=&210:osbyte=&FFF4
 110 FOR pass=0 TO 3 STEP3:P%=&A00
 120 [OPT pass
 130 .install
 140 SEI:LDA #entry MOD &100:STA INSV
 150 LDA #entry DIV &100:STA INSV+1
 160 SEI:LDA #rdch MOD &100:STA RDCHV
 170 LDA #rdch DIV &100:STA RDCHV+1
 180 CLI:RTS
 190 .entry
 200 CPX #0:BNE notme:STA character
 210 TXA:PHA:TYA:PHA
220 LDA character: CMP #ASC"@"
 230 BNE notAlt:LDA #250
 240 STA character: JMP exit
 250 .notAlt
 260 CMP #ASC"0":BNE exit
 270 LDA #202:LDX #0:LDY #&FF
 280 JSR osbyte: TXA: EOR #32
 290 AND #8+32:BEQ exit
 300 LDA #ASC"@":STA character
 310 .exit
 320 PLA: TAY: PLA: TAX: LDA character
 330 .notme
 340 JMP !INSV
 350 .rdch
 360 JSR !RDCHV:BCS out
 370 CMP #250:BNE out1
 380 JSR !RDCHV:BCS out:ORA #&80
 390 .out1
 400 CLC
 410 .out
 420 RTS
 430 .character EQUB 0
  440 ]NEXT
 450 a$="SAVE ALT "+STR$~install+" "+ST
R$~P%
  460 PRINT a$:OSCLI a$
```

Writing a Compiler (Part 5)



David Spencer concludes his discussion of compiler writing.

Last month we developed the actual compiler program which will take an expression and convert it into machine code, albeit by a somewhat convoluted route via a Basic program. We also saw the need for a linker and run-time library, and these are the areas that we shall concentrate on now.

THE RUN-TIME LIBRARY

Remembering back to last month, the function of the run-time library is to provide a common set of routines for by the compiled use program. In the case of our expression compiler this amounts to routines to perform addition, subtraction, multiplication, division and negation, as well as routines to stack a number and print out the result. In a real language compiler there would be routines for each built in function (such as ASC and TAN), and also lower level routines, for example those needed to allocate memory for arrays.

The code for our run-time library is shown in listing 1. This should be typed in and saved. When this program is run it will assemble the library and save it with the filename 'RTLib'. You should be able to pick out the various routines from the library, and also three low-level routines for pushing a value onto the stack, and pulling a value off into one of two temporary stores. Because of space limitations, the run-time library listed doesn't support multiplication or division, and prints its results in hex. This month's magazine disc contains a full version of the library. The actual arithmetic routines are totally standard, using the 6502's instructions for addition and subtraction, and shift and add/subtract operations for multiplication and division. The only confusing aspect may be the way in which return addresses are pulled off the stack at the start of each routine. This is necessary because the same stack is also used for the actual operands, and they mustn't get mixed up.

Two points to note about the run-time library are firstly that it is assembled to run from address 0, and hence is not directly executable, and secondly, when the file is saved it is prefixed with a header containing the length of the library, and the offsets within it of the various routines. Both of these features are vital for the linking operation, as we shall see shortly.

RUN-TIME ERRORS

As mentioned in the first part of this series, errors can occur during execution which cannot be detected at compile time. The run-time library detects two such errors - overflow and divide by zero, and reports these in the same way as any other error.

THE LINKER

We now have a program generated by the compiler from last month, and a run-time library. However, as these stand they cannot be executed for two main reasons:

1. Both pieces of code are assembled to run from address 0, and would therefore overwrite each other. Also, this isn't a particularly sensible execution address as the program would corrupt valuable workspace.

2. The compiled program cannot directly call the run-time routines, as it doesn't know their correct addresses. Instead, if you remember from last month, arbitrary numbers were used to identify the routines.

To form an executable program we firstly need to tie the compiled code to the run-time library

Beebug April 1990

Workshop - Writing a Compiler

in such a way that they will execute at a sensible address, and secondly, change all the calls to run-time routines so that they point to the correct address. Another operation which becomes necessary is the relocation of the two pieces of code. This is because both parts of the code have instructions which depend on the execution address. For example, the run-time library has JSRs to internal routines, and the compiled code loads pointers to values to be stacked (see the example given last month).

All these tasks fall to the *Linker* which was described briefly last time. Our Linker, which is written in Basic, is given in listing 2. As always, this should be typed in and saved. Before explaining how it is used, we shall look at how it works.

The Linker starts by looking at both input files to determine the final length of the code. The table of addresses at the start of the run-time library is then read in to allow the calls from the compiled code to be corrected. The two pieces of code are then read into a buffer, relocating the relevant instructions in the process. Finally, the resulting code is saved with the correct execution address.

USING THE LINKER

The moment of truth is drawing near - we almost have an executable program. Start by using the compiler from last month to compile an expression, following the instructions given then. Now, run the Linker program, making sure that the 'Code' file produced by the compiler, and the 'RTLib' file are both in the current directory. The Linker will examine both files and inform you of the length of the final code. It will also prompt you to enter an address for where the final program should be executed. This depends a lot on the computer and the length of the code. If it is under &200 bytes then &900 is a good address to use, as is &DD00 on a Master only. For longer code, the value of PAGE can be used (&E00 on a Master and &1900 on a DFS-based model B). However, in this case, any program in memory will be lost when the compiled code is run. Whatever address you choose, the Linker will read the two files and combine them into an executable program which will be saved under the name 'EXECUTE'. All the relocation and linking is performed automatically at this stage.

All that now remains is to execute the final code by typing:

*EXECUTE

If everything is OK then you should see the answer to the compiled expression printed immediately. If this is not the case then the chances are that you have made a mistake in entering the compiler, linker or run-time library. If this happens then you should check each of the programs very carefully.

SELECTIVE LINKING

In our simple example, the run-time library contained only a few routines and was not particularly long. Therefore, it was acceptable to include all of the library, even though some routines might not be used. However, in a real language the run-time library might be many kilobytes long, and contain several hundred routines. It would be very wasteful to include all these routines when a program might only use a tenth of them.

To overcome this, a technique called *selective linking* can be employed in which only the routines needed are included. This is normally done by scanning the compiled program to find which routines are needed, and then picking each routine out of the library and linking it in separately. A twist to this is that it is possible that one library routine may in turn use another one, and so the scanning and linking must be done recursively until all the necessary routines have been included. Obviously this greatly complicates the linker, but is the price that must be paid for more compact code.

WHAT USE IS IT?

By now you might be thinking that we have gone to an awful lot of trouble to produce a compiler for simple expressions which could be performed on a pocket calculator in seconds. However, the material we have covered in these five articles comprises much of what is needed for any compiler, and writing a compiler for an entire language such as Basic, Pascal or C is really only an extension (albeit a large one) of what we have done here.

Once again, if you are interested in finding out more then I can highly recommend the definitive book on compilers: *Compilers -Principles, Techniques and Tools* by Aho, Sethi and Ullman. This is an 800 page book published by Addison-Wesley and costs £19.95 in paperback form. It should be available from most computer bookshops.

THE END

That brings me to the end of this series on compilers and also to the end of my authoring of the Workshop. I hope that the nineteen Workshops in which I have covered six different topics have proved useful and interesting to at least some.

Listing 1

10 REM Program Run-Time Library 20 REM Version B1.0 30 REM Author David Spencer 40 REM BEEBUG April 1990 50 REM Program Subject to Copyright 60 : 100 DIM code 1000 110 ret=&70 120 ret2=&72 130 acc=&74 140 acc2=&78 150 ptr=&7C 160 ret3=&7E 170 oswrch=&FFEE 180 osnewl=&FFE7 190 : 200 FOR pass=4 TO 7 STEP 3 210 P%=0:0%=code 220 [OPT pass 230 .stack 240 PLA:STA ret 250 PLA:STA ret+1 260 STX ptr:STY ptr+1 270 LDY #3 280 .stack2 290 LDA (ptr), Y:PHA:DEY 300 BPL stack2 310 LDA ret+1:PHA 320 LDA ret: PHA: RTS 330 : 340 .push 350 LDX #acc:LDY #0 360 BEQ stack 370 : 380 .pop 390 PLA:STA ret 400 PLA:STA ret+1 410 LDY #0 420 .pop 2 430 PLA:STA acc, Y:INY:CPY #4 440 BNE pop_2 450 LDA ret+1:PHA 460 LDA ret: PHA: RTS 470 : 480 .pop2 490 PLA:STA ret 500 PLA:STA ret+1 510 LDY #0

520 .pop2 2 530 PLA:STA acc2, Y:INY 540 CPY #4:BNE pop2_2 550 LDA ret+1:PHA 560 LDA ret: PHA: RTS 570 : 580 .add PLA:STA ret2 590 PLA:STA ret2+1 600 JSR pop: JSR pop2 610 .add1 CLC:LDY #0 620 .add2 630 LDA acc, Y: ADC acc2, Y 640 STA acc, Y: PHP: INY 650 CPY #4:BEQ add3 660 PLP: JMP add2 670 .add3 PLP:BVC add4 680 JMP overflow 690 .add4 JSR push 700 LDA ret2+1:PHA 710 LDA ret2:PHA:RTS 720 : 730 .subtract 740 PLA:STA ret3 750 PLA:STA ret3+1 760 JSR negate: JSR add 770 LDA ret3+1:PHA 780 LDA ret3:PHA:RTS 790 : 800 .multiply 810 .divide 820 BRK: EQUB 0 830 EQUS "Not Implemented": EQUB 0 840 : 850 .negate 860 PLA:STA ret2 870 PLA:STA ret2+1 880 JSR pop:SEC:LDY #0 890 .negate2 900 LDA #0:SBC acc, Y:STA acc, Y 910 PHP: INY: CPY #4: BEQ negate3 920 PLP: JMP negate2 930 .negate3 PLP: JSR push 940 LDA ret2+1:PHA 950 LDA ret2:PHA:RTS 960 : 970 .print 980 JSR pop:BIT acc+3:BPL print2 990 JSR push: JSR negate 1000 JSR pop:LDA #ASC"-":JSR oswrch 1010 .print2 1020 LDA #ASC"&":JSR &FFEE:LDY #3 1030 .print3 LDA acc, Y: JSR hex 1040 DEY: BPL print3 1050 JSR osnewl:RTS 1060 : 1070 .hex 1080 PHA 1090 LSR A:LSR A:LSR A:LSR A 1100 JSR hex2:PLA:AND #&F 1110 .hex2 ORA #&30:CMP #&3A 1120 BCC hex3:ADC #6 1130 .hex3 JMP oswrch 1140 :

Workshop - Writing a Compiler

1150	.overflow
1160	BRK:EQUB 0
1170	EQUS "Overflow": EQUB 0
1180]NEXT
1190	H%=OPENOUT"RTLib"
1200	PROCput (H%, P%)
1210	PROCput (H%, add)
1220	PROCput (H%, subtract)
1230	PROCput (H%, multiply)
1240	PROCput (H%, divide)
1250	PROCput (H%, negate)
1260	PROCput (H%, stack)
1270	PROCput (H%, print)
1280	PROCput (H%, &FFFF)
1290	DIM blk 13
1300	?blk=H%:blk!1=code:blk!5=P%
1310	X%=blk:Y%=blk DIV 256:A%=2
1320	CALL &FFD1 : REM OSGBPB
1330	CLOSE #H%
1340	END
1350	:
1360	DEF PROCput (hand, val)
1370	BPUT #hand, val
1380	BPUT #hand, val DIV 256
1390	ENDPROC

Listing 2

10 REM Program Linker 20 REM Version B1.0 30 REM Author David Spencer 40 REM BEEBUG April 1990 50 REM Program Subject to Copyright 60 : 100 H%=OPENIN"Code" 110 IF H%=0 PRINT"Compiled code not fo und": END 120 I%=OPENIN"RTLib" 130 IF I%=0 PRINT"Run-time library not found":CLOSE #H%:END 140 ON ERROR PROCclose: PROCreport: END 150 L%=FNget (1%) 160 L%=L%+EXT#H% 170 DIM buffer L%, blk 18, name 20 180 DIMroutines(20) 190 PRINT"Final code is &";~L%;" bytes long." 200 INPUT "Enter execution address &" AS 210 add%=EVAL("&"+A\$) 220 ptr=buffer:ptr2=buffer+EXT#H% 230 routines=FNgetroutines 240 PROCreadcode (H%, ptr) 250 PROCreadlibrary (1%, ptr2) 260 \$name="Execute" 270 !blk=name:blk!2=add%:blk!6=add% 280 blk!10=buffer:blk!14=buffer+L% 290 X%=blk:Y%=blk DIV 256:A%=0 300 CALL &FFDD:REM OSFILE 310 PROCclose 320 END 1000 DEF FNget (hand%) 1010 =BGET#hand%+256*BGET#hand%

1020 : 1030 DEF FNgetroutines 1040 LOCAL count, off, got:count=0 1050 off=ptr2-ptr+add% 1060 REPEAT 1070 got=FNget(I%) 1080 routines (count) = got+off 1090 count=count+1 1100 UNTIL got=&FFFF 1110 =count-1 1120 : 1130 DEF PROCreadcode (hand%, add) 1140 REPEAT 1150 ?add=BGET#hand% 1160 IF ?add=&A2 PROCldx:GOTO 1190 1170 IF ?add=&A0 PROCldy:GOTO 1190 1180 IF ?add=&20 OR ?add=&4C PROCjsr 1190 add=add+1 1200 UNTIL EOF#hand% 1230 DEF PROCldx 1240 add=add+1:temp=BGET#hand%+add% MOD \$100 1250 ?add=temp:IF temp>&FF over=1 ELSE over=0 1260 ENDPROC 1280 DEF PROCldy 1290 add=add+1:temp=BGET#hand%+add% DIV &100+over 1300 ?add=temp 1310 ENDPROC 1320 : 1330 DEF PROCjsr 1340 add=add+1:temp=BGET#hand% 1350 IF temp>routines OR BGET#hand%<>0 PRINT"Bad library call":END 1360 !add=routines(temp):add=add+1 1370 ENDPROC 1380 : 1390 DEF PROCreadlibrary (hand%, add) 1400 off=ptr2-ptr+add% 1410 REPEAT 1420 ?add=BGET#hand% 1430 IF ?add=&20 OR ?add=&4C PROCreloc 1440 add=add+1 1450 UNTIL EOF#hand% 1460 ENDPROC 1470 : 1480 DEF PROCreloc 1490 temp=FNget (hand%) 1500 IF temp<&FF00 temp=temp+off 1510 add!1=temp:add=add+2 1520 ENDPROC 1530 : 1540 DEF PROCclose 1550 CLOSE #H%:CLOSE #I% 1560 ENDPROC 1570 : 1580 DEF PROCreport 1590 REPORT: PRINT" at line "; ERL 1600 ENDPROC

Elegant Entry to Wordwise Plus

Tom Boyd shows how you can streamline entry into Wordwise Plus, so that everything is set up ready to start work.

I usually use my BBC model B for word processing, using Computer Concepts' Wordwise Plus. !BOOT files can be used to make life simple, but before I got organised in this way, I had to do the following, each time I switched on (Type is used to mean enter what is shown and then press Return. Press means the key shown, only):

Iob	Purpose
Type *WORD.	Invoke Wordwise
Press 2	Select "load file"
Type LT1	This is my letter template
Press Escape	Change to text screen
Press cursor keys	Move to position for date

This is not a great chore, but why do things the computer can do for you? Furthermore, it doesn't mistype things. Now I merely insert my Wordwise boot disc and perform a shift/break.

The boot disc was prepared as follows. Using Wordwise Plus, I created the following text file and saved it as !BOOT (the ':NEW' ensures that I don't get an "Are you sure?" message when rebooting after doing other Wordwise work).

*WORD. :NEW 21.T1

Note that Return is needed after 2LT1 in Wordwise file.

Then I ran the following program, replying with 'BOOT' to the prompt resulting from line 100 (be sure to save the Basic program after typing it in as testing 'BOOT will wipe it from the computer's memory). The program adds a character to the end of the 'BOOT file to simulate pressing the Escape key so that when 'BOOT file is executed, the text to be edited is displayed instead of the Wordwise Plus menu.

```
10 REM Program AddEsc
20 REM Version B1.0
30 REM Author Tom BOYD
40 REM BEEBUG April 1990
50 REM Program subject to copyright
60:
100 INPUT"FILE",F$
```

110 DF=OPENUP(F\$) 120 PTR#DF=EXT#DF 130 BPUT#DF,27 140 CLOSE#DF 150 PRINT"Done."'

Lastly, I typed in *OPT4,3 to set the disc's shift/Break action.

That's all there is to it! To set up other discs similarly, just *COPY the files !BOOT and LT1 and perform *OPT4,3 on any disc that you want to boot into Wordwise Plus. The contents of LT1 are not put into the !BOOT routine so that LT1 may be revised easily.

You can of course improve on these rudiments. The !BOOT file I actually use is:

*WORD. *TKWK *FX202,176,79 (set Caps/Shift lock off) :NEW 2LT1 :CURSOR DOWN 8 :CURSOR RIGHT 9 T (the 9 is followed by two Returns).

*TKWK loads the function key definitions I like to use with Wordwise ("delete line", "select NLQ", etc.). On a BBC B, define the keys as you want them, and then type:

*SAVE TKWK B00 +100

The two cursor commands move the cursor to where I must be to insert the date (the values will of course be different if your letter format differs from mine). This file, "LT1", is formatted as follows (I've changed all "Green" codes to ^, and all "White" codes to ~).

```
^LT1, 21-06-88
^OC27,A%~
^TS0^BS12^EP
^JO^DP35^LM6^DF~
^IN50~T.K. Boyd
Landsea College
Petsea, W.Sx.
GU28 ONB
^SP1^TI50 1990
^CI~Dear
^TI30~Yours,
^OC27,80^bp~
```

Elegant Entry to Wordwise Plus

Note the absence of a "white" code in the 9th line before "1990". This forces an error during printout if I have forgotten to enter a date. Someone with a Master may be able to suggest a way to have the date inserted by the system. The second line allows me to select draft or NLQ printing by entering a value for A% from the menu screen (both handled by function keys, of course). There is a space after "Dear", so that a simple down arrow after the date is all I need to do after entering the date.

While debugging !BOOT files, you do not need to process them with the Basic program given above. The only purpose for this is to add the final touch of entering the text instead of remaining at the Wordwise Plus menu. If you have run the Basic program, and wish to re-edit the !BOOT file then, during the edit, remove the character which will appear at the end of the BOOT file. You will need to re-run the Basic program to restore the "go to text" feature.

In your !BOOT files, it is important to specify the sequence of characters exactly as you would enter them by hand. I find it helpful to do the sequence by manually, noting each key as it is pressed. Then I repeat the sequence from my notes. Only then do I try to create the !BOOT file. Missing or extra carriage returns are the biggest problem. For example, in the !BOOT file given, a carriage return after the 2 in 2LT1 would cause the computer to try to load a file with no characters in its name. Failing to put two carriage returns after :CURSOR RIGHT 9 would leave the computer at "Press a key" if the Basic program hadn't been run, or at the menu if it had been (the "go to text" character would be "spent" answering "Press a key").

An exact copy of my system will work, but I hope I have explained enough to allow you to customise your system so that it conveniently does what you want. Computers are meant to be servants, after all!



Order Out Of Chaos (Part 2)

Jim Vernon concludes his discussion of the world of chaos (and order).

As I stated last month (see part one of this article) the most interesting aspect of the program CHAOS3 has yet to come. The vertical white lines visible in figure 4 (see last month), especially near c=0.96, are not poor printing, but indicate areas where X(n) has suddenly changed again from being chaotic to a number of discrete values. Thus near c = 0.96 there are 3 values only of X(n), namely - 0.149406, 0.488004, and 0.959447 - as calculated using the program CHAOS1.



Figure 5

What is happening near the central one of these 3 points can be seen graphically by again magnifying a limited area using the CHAOS3 program, as in figure 5. This shows that as c increases from 0.956, X(n) quite suddenly moves from chaos to a single value (the other 2 values that X(n) takes at this value of c are not shown at this scale). X(n) then continues first as a single line, then forking and forking again and eventually returning to chaos. The pattern, allowing for the difference in scales, looks very similar to that in figure 4, except that it needs to be reflected about the horizontal axis. Further magnifications can be made for c in the range 0.9622 - 0.9624, and if the distances apart of the values of c at the various nodes are measured, these are again found to decline geometrically, with the same divisor of about 4.67 (known as the Feigenbaum constant).

Beebug April 1990

The same search carried out on other white vertical bars (more of which can be found by magnifying various parts of figure 4), produces many values of c in the chaotic region for which there are sets of discrete values of X(n), the number of sets per value of c varying from 3 to 12 or more. Again, enlargement of any set shows the same forking pattern, on the same geometric scale. In some cases, very minor changes in the value of c move X(n) from chaos to discrete values and back again. For example, if c = 0.99 X(n) is chaotic, for c = 0.9901 X(n) has 4 values only and at c = 0.9906 chaos has returned. The search for values of c which produce discrete values of X(n) can thus be quite tricky. Overall, the apparently chaotic area between c = 0.8925 and 1 is in fact an area of subtle complexity which could be studied almost indefinitely.





OTHER FUNCTIONS

In any iteration, if the function is divergent, X(n) can become unmanageably large quite quickly. Interest has therefore to be concentrated on functions which rise to a maximum and then fall again. Any such function can replace 4c.Z.(1-Z) in line 1020 in programs CHAOS1/2/3. Functions worth trying are c.Z.(1-Z^2), c.Z.(1-SIN(Z)), c.SIN(Z), 1-c.Z^2 and Z.EXP(c.(1-Z)). To start with, a wide range for c of 0 to 5 and for X of -5 to +5

Order Out of Chaos

should be tried in CHAOS3, with low values for D%, M% and T%. This will give a rough

which the area to can be identified. CHAOS2 is least useful in this work the value of X(LIM) would have to be changed.

Figure 6 is an example of applying CHAOS3 to the function:

f(X) = c.SIN(X)

Value of c at node	Distance apart of nodes	1st.distance divided by 2nd.
2.2630	0.3540	
2.6179	0.0794	4.47
2.6973	0.0174	4.56
2.7147	0.0036	4.83
2.7183	0.0008	4.5

Table 3. Nodes in c.SIN(X)

Again the close similarity with figure 4 will be noted, including a white vertical bar near c=3. This graph can be magnified to find the detail of the nodes and the distances apart measured. The values of c at nodes are given in Table 3 and once again it will be seen from the last column that there is a geometric progression with the same divisor of about 4.67.

over a wide range of functions, which has proved of considerable interest to scientists and philosophers.

For convenience, the program CHAOS3, referred to above, has been included again on this month's magazine disc.

B



picture quickly from magnify

Broadly similar results will be obtained with all these equations, with sets of bifurcations appearing, then chaos,

and then once again

various values of c

which produce nodes at

discrete values of X(n).

The distances apart of

the nodes all conform to

the same geometric progression. Thus not

only has 'chaos' its

internal consistencies

for each function, but

there is a remarkable

universality about it

PAGE ZERO MEMORY LOCATIONS FOR LOOP **CONTROL IN BASIC IV**

HINTS HINTS HINTS and tips

Bernard Beeston

When writing a FOR-NEXT loop it is quite possible to specify a memory location directly as the loop control variable. For example, consider:

- 10 ?&70=99
- 20 FOR ?&70=1 TO 9
- 30 PRINT ?&70
- 40 NEXT
- 50 PRINT ?&70

Line 10 'seeds' the address with an out-of-range value just to prove all is well. Line 30 prints the value at the address each time round, while line 50 proves, as always, that the FOR-NEXT loop exits with the value of the control variable one more than the terminating value specified.

As illustrated, you are limited to numbers no greater than 255 (and naturally integers only), but using pling '!' allows up to four bytes per number to be used, for example:

FOR !&70=&8000 TO &8FFF

INVISIBLE DFS CATALOGUES

Stephen Bodman

Using Computer Concept's Disc Doctor, you can make disc catalogues unlistable. This is achieved by including the VDU21 command in the disc title. To do this type:

*DZAP 0

and you will be presented with a hexadecimal listing of track zero, which is where the catalogue is stored. Change one of the numbers on the top row to 15 (hex for 21), and also preferably make the first file name a dummy name. The disc catalogue cannot then be listed, and so neither compacted nor copied.

PROGRAM PROTECTION

Stephen Bodman

You can easily add a small routine to any program to stop it being listed. Only do this when the program is complete. Start by adding the following lines to the start of the program:

1REM** 2REM** 3REM** 4REM**

Then add the following procedure at or near the end of the program:

10000DEFPROCProtect 10010P%=PAGE+1 10020REPEAT 10030IF P%?3=&F4 P%?4=12:P%?5=21 10040P%=P%+P%?2 10050UNTIL ?P%=&FF 10060ENDPROC

HINTS HINTS and tips and tips

After this type:

PROCProtect followed by:

DELETE 10000,10060

to delete these lines. Now save the program. When you list it nothing appears.

MEMORY WIPE

Al Harwood

To wipe all user memory clear use:

*FX200,2

CALL !-4

This works because the FX call clears memory when Break is pressed, and CALL !-4 is equivalent to Break. This can be useful with protected software to ensure no usable code is left in memory on exit.

AUTOBOOTING WORDWISE

Stephen Bodman

Here is a simple example of a BOOT file for more efficient entry into Wordwise Plus:

1*KEY 10 *W. |M N|M 2 Filename |M *FX |M|M 2CALL !-4

This programs the Break key and then instigates a Break sequence. The filename should be the name of any Wordwise file which you wish to load automatically at the start of a session, a standard layout of some kind maybe (see also the article on Wordwise in this month's issue of BEEBUG magazine).

PRINTING IN THE CORNER OF THE SCREEN Mike Williams

An old hint this one but a very helpful suggestion for when the need arises. Normally, printing or displaying any character in the bottom right-hand corner of the screen causes the screen to scroll regardless. This can be undesirable. Scrolling, in this position, can be inhibited by executing:

?&D0=2

The system can be returned to its normal state by typing: B

?&D0=0

RISC USER

The Archimedes Magazine & Support Group

Risc User is enjoying the largest circulation of any magazine devoted solely to the Archimedes range of computers. Now in its third year of publication, it provides support to schools, colleges, universities, industry, government establishments and private individuals. Existing Beebug members, interested in the new range of Acorn micros, may either transfer their membership to the new magazine or extend their subscription to include both magazines.

A joint subscription will enable you to keep completely up-to-date with all innovations and the latest information from Acorn and other

suppliers on the complete range of BBC micros. RISC User has a massive amount to offer to enthusiasts and professionals at all levels.

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

GENESIS

A review of this impressive software package from Software Solutions for generating multi-media information systems.

USING DRAW FILES IN BASIC

A library of routines which enable you to use Draw files within Basic.

PINEAPPLE'S COLOUR DIGITISER

A review of the new real-time colour digitiser for the Archimedes.

UNDER THE LID

A major series explaining the hardware that makes up the Archimedes.

MULTI-COLUMN PROGRAM LISTINGS

A utility for printing multi-column listings with a selectable number of columns.





UNDERSTANDING THE SOUND SYSTEM

A technical explanation of the Arc's sound system.

ASSEMBLER WORKSHOP

A major series for the more advanced ARM processor programmer.

COMMAND LINE HISTORY BUFFER

A utility which allows you to repeat and edit previously typed commands.

MASTERING THE WIMP

A major series for beginners to the Wimp programming environment. In the latest issue lcons: the groundwork,

THE HP DESKJET PLUS PRINTER

A review of this printer - part of the series on high quality printers.

INTO THE ARC

A regular series for beginners. Te latest article is -Emulating a BBC micro.

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

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

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

SUBSCRI	PTION DE	TAILS
Destination	Additio	onal Cost
UK,BFPO &Ch I	s	£ 8.10
Rest of Europe a	nd Eire	£12.00
Middle East		£14.00
Americas and A	frica	£15.00
Elsewhere		£17.00

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



NIMBLE FINGERED

I have found the Nimble Typer program from Vol.8 No.3 extremely useful, but I would like to be able to set the filename containing the abbreviations from outside the program, rather than at a prompt.

Andy Pye

The simplest way to do this is to remove lines 1520 to 1610 from the original program, and then run this to re-assemble Nimble. Now, to run Nimble, type: *LOAD Nimble

then set the file name using: \$&A2F="filename" and run the program using:

CALL &A00

IMPROVING ONE'S HERITAGE

Following your review of Heritage in Vol.7 No.9, many of your readers may have purchased the package and found that David Lane of Bel technology, who wrote the program, has not been answering letters or providing any of the promised updates.

Your readers may like to know that a user group is being set up in order to solve problems and exchange experiences. The excellent program written by David Lane contains a few bugs and annoyances. All those known have been removed, and the update will be available to members of the group for the cost of paper, postage etc. If anyone is interested in the user group would they send an SAE to Bill Woollen, "Greensward", Townsend, Harwell, Didcot OX11 0DX for more details.

Bill Woollen

DISC RECOVERY

The last article in David Spencer's excellent series of articles on disc drives covers the recovery of lost or corrupted files (BEEBUG Vol.8 No.5). His description of the steps to be followed after such a catastrophe, with the aid only of a disc sector editor, is excellent. There is, however, a very simple and visual way of locating possible locations of deleted programs for DFS users, other than the exceedingly tedious method of constructing a free space map using pencil and paper after using *MAP and *INFO. This was published in BEEBUG Vol.5 No.4 as part of Bernard Hill's Disc Manager Utility. By pressing 'S', after running this program, a completely visual representation of possible locations becomes obvious (see printout).

Phil Dyer



One of Mr.Dyer's printouts is reproduced here for interest.

Drive 8 21	tiles	litle	LETTERS-IN				
Filename :\$.BR	ADLEY						
Tracks>							
1		2	3	4	5	6	7
01234567890123	456789	812345	67890123456	789812345	6789012345	6789812345	6789812345678
CANAL A MAR A		111					
C	11 1	83					
	11 1	11	2 2 6 BEERING				
	11	11	S S S NUMBER				
18 188 18881	111 1	11	1111000				
	111 1	111	2 5 5 10000				
	111 1	111	8 2 8 2000				
	111 1	111	1111000				
	111 1	668	2221000				
	111 1	113	311000				
81234567898123	456789	012345	67890123456	78901234	6789812345	6789012345	6789812345678
Disc usage: 29	.7%						
Spacebar to re	turo						

IMPROVING BASIC PROGRAMS

The technique of replacing IF-THEN statements as described in First Course (BEEBUG Vol.8 No.8) is one I have long used myself, but there are two flaws in the article, both concerning the listing in the right-hand column of page 24. Line 100 should start with 'PRINTTAB' and not 'PRINT'. Secondly, it is incorrect to say that the SPC function cannot be used with the INPUT statement. On the contrary, the User Guide states that it can ONLY be used with PRINT and INPUT. Thus the routine can be expressed as:

100 PRINTTAB(0, n) "File name:"

- 110 REPEAT
- 120 INPUTTAB(12, n) SPC20TAB(12, n) fname\$
- 130 UNTIL LEN(fname\$) <8

Another useful technique for cutting the number of IF statements is to remember that all Boolean expressions also have values of 0 or -1. For example, if you want to decide when an 's' (ASCII code 115) should be added to a noun depending on whether some value, x%, is one or more, I would write:

PRINT"I have ";x%;" object";CHR\$(-115*
(x%<>1));"."

If x%=1 then the value in brackets becomes -115*0 which produces the null character, otherwise the product is -115*-1 which yields 115, the code for 's'. Gareth Leyshon

I must own up to an oversight as far as the use of SPC is concerned. Somehow or other I acquired the myth (as it now turns out) that this would not work at all with INPUT. How wrong can one be? The other point is also interesting and useful, and I had already included another example of this technique in this month's First Course before reading Mr.Leyshon's letter.

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.

For Sale: Acorn Electron Plus 1, tapes, magazines, books, data recorder, leads £100. Tel. (0727) 67669.

M128, Philips 14" colour monitor, 40/80T disc drive, View, Viewsheet, Viewstore £550, Morley Teletext Adaptor/ROMs/manual £50, Dumpmaster ROM £12, WYSIWYG ROM £15, AMX Mouse £40, ROM cartridges £6 each, 20Mb Winchester with 40/80T 5.25" floppy disc drive £350, plus various books, software, BEEBUG magazines/discs, Micro User and Acorn User magazines, prices exclude postage. Tel. (0726) 843487.

BBC Master, disc drive, monitor and software £425 o.n.o. Tel. (0277) 625745.

Beeb goodies for sale, BEEBUG magazines to '86, Micro User etc to '88 £7.50, Jet Printer and manual £20, many tapes and software, (1.2 OS) including Elite, Snowball, Graphic ROM & light pen £30, or £50 the lot. Tel. (0707) 323736.

Master 512 with single 80T DS/DD drive in double case with PSU. Hitachi green screen monitor, Mosplus ROM, 2 ROM cartridges & lots of disc software (DOS & ADFS) £460. Taxan Kaga KP810 printer in very good condition, little used, with manual BBC cable and cover £80. 6502 2nd Processor with Hi-Wordwise & 6502 Development package £80. 2x Tandon TM100-2 40T DS/DD full height disc drives, uncased in good working condition, both for £50. Tel. (0483) 39496 (days) and ask for Stewart, or (0273) 605339 (eves) and talk to Linda.

M128 Philips 12" monitor, double disc drive 5.25" 40/80T & 3.5", Master modem with command ROM, Wapping Editor including ROM & discs, View, Viewsheet, Viewstore, Quest Mouse, 80T disc drive other ROMs & books £480. Tel. (0622) 30256 eves & w/ends.

Wordwise +, book by Bruce Smith £5, EPROM assortment; 6x2732, 1x2764, 3x2764 programmed once £15 the lot

or £2 each. WANTED: Colour Screenprint ROM ES651 by ESM. Tel. (0444) 440622.

AMX SuperArt (Master) & mouse £35, BBC board & Replay £30, Stop Press £28, Extra Extra £10, AMX Max £10, Publisher £30, Genie (Master cartridge) £50, Prism modem 2000 £20, all boxed as new, updated to ARC. Tel. (0256) 702227.

Morley Teletext Adaptor with ROMs ATS 2.59 & SUP 4.00, disc and manuals £40, Mini Office II ROM version and manual £20, Disc Doctor ROM & manual £10, View A1.4 ROM & manual £6. Tel. 01-777 3811 after 7pm.

M512 with twin 40/80 disc drives, Taxan Supervision 625 colour monitor, Magic Modem, BBC Word, Viewsheet, Wordwise Plus, BEEBUG programs and magazines, latest version DOS +, Problem Solver, PC database and word processor. All excellent condition with instructions £650. Tel. (0707) 328091.

Psion organiser 2 model XP plus 32k battery backed RAM plus comms link + BBC link + Finance pack + manuals £80. Tel. 01-858 6086 eves.

Mini Office II ROM version (see BEEBUG Jan/Feb '90 pg 5). I have one of these for sale, it is in pristine condition, has never been fitted to any machine and is complete with book, mountings and original box £33. Tel. (0276) 65876.

M128, Pace 40/80T drive, Toshiba printer, Microvitec 653 colour monitor, plinth, Interword, BEEBUG C, Manuals, many books, numerous games and utilities, joystick, complete BEEBUG mags and much more. Tel. (0689) 53045.

Morley Teletext Adaptor, 2 ROMs manuals & disc £50, Morley Master AA board with control ROM & manual £28, Tandy GP115 colour printer/plotter £40, Master Reference manuals 1&2 £12, BEEBUG Sleuth ROM £9, Master ROM £12, BEEBUG Dumpmaster ROM £12, Pineapple ADU Utilities ROM £12, Hyperdriver ROM £12, BEEBUG Help ROM £8, MOS plus ROM £9, BEEBUG Basic Booster ROM £5, all originals complete with manuals. Care Quad ROM cartridge £8, 2 Master ROM cartridges £4 each, BEEBUG Masterfile ADFS 80T £8, Dabs Press Sidewriter DFS 40T £3, Kansas Personal Finance 40T £3, Mini Office I 40T £2, BEEBUG vol.1 to vol.8 offers? Tel. (0734) 667659 after 6.30pm.

Panasonic Daisy Wheel printer KXP3131, 2 yrs old, excellent condition, hardly used (2nd printer), ribbons, tractor feed, cost £385, no sensible offer refused. Tel. (0420) 83555.

Apollo modem and software £45, Commstar £20, Printmaster £30, Stop Press, Super Art and mouse £100, Extra Extra £15, Acornsoft Logo £50, Master Guide part 1 £5, New Advanced User Guide £10, Master Compact Music System £15, Elite (Master Compact £10, Sideways RAM book by Bruce Smith £7, Psion II software. Tel. (0925) 755139 after 5pm.

5.25" disc drive 40/80T in dual case with PSU, plus 50 unused DSDD discs, plus disc box, price including parcel post £70, Morley AA board £30, Master Cartridges £6 each or 8 for £30, Viewstore ROM £30. Tel. (04243) 4500.

BBC model B issue 7, data cassette & games £175. Tel. (0993) 776066.

BBC with Torch Z80 2nd processor and disc pack (runs standard BBC and CPM software). Twin D/S 80T disc drives, Mcirovitec 1451 med. res. monitor, Computer Village sideways RAM/ROM board, Star DP515 printer, Watford mouse, Voltmace joystick, META editor and assembler, AMX Stop Press, GDUMP, Watford Beebmon, Termulator. MCP Torch Perfect Writer, Speller, Filer, Calc. plus books and other software £475 o.n.o. Tel. (0258) 840038 eves. WANTED: Two 6234 LP 8K chips. Tel. (0823) 289378.

Eight complete vols. (1-8) of BEEBUG magazine in binders, all in excellent condition, reasonable offers. WANTED: 5.25° DD, DD/DS 40/80T switchable with PSU. Tel. (0462) 682961.

Acorn 20Mb hard disc, with interface, for Archimedes 300 series £225

ROM podule with 144k RAM and battery backup £40, Sony 1Mb 80T 3.5" DD £40, Acorn AP-100 matrix printer £35. Tel. (0334) 78171.

Akhter single sided 40T drive £35, plus delivery. Tel. (0792) 290966.

M512 DOS 2.1 ROM cartridges, Opus twin D/S 40/80T disc drive own PSU, Philips High Res. Amber monitor. £700 complete. Tel. (0276) 23124.

512 co-processor in Acorn case with PSU M-TEC basic "DOS plus reference guide" (Glentop). Norton "Programmer's Guide to the IBM PC". Write to; 19 Southcliffe, Great Harwood, Lancs, B& 7PP.

Archimedes floppy drive, redundant due to upgrading to hard disc £95 o.n.o. + postage if necessary, System Delta V2 £50, System Delta prog. manual £15, Gamma Plot £40, hardly used. Tel. (0689) 70263 eves.

WANTED: Colossal Adventure by Level 9 on disc for BBC B £5. Tel. 01-270 2604 office hours.

M512 V2.1, Opus 40/80 twin disc drive with own PSU, Philips amber monitor, many ROMs, discs, cassette based software, boxed including manuals. All in excellent condition. £600. Tel. (0742) 876537 after 6.30pm.

WANTED: 5.25" drive, 40/80T switchable, for reasonable price. Tel. (0225) 311715.

BBC Master 512 with DOS 2.1 mouse & GEM software, Zenith 12" green monitor, Cumana 800k 40/80 drives, Juki daisywheel printer, AMX Super Art & mouse, View, Viewsheet, Viewstore, Viewspell, Viewplot, cartridges, all leads, plinth, all manuals, excellent condition £550 o.n.o. will split. Tel. (0307) 63900 after 5.30pm.

Beebug April 1990

Psion Organiser II, model XP, complete with 1x16k & 1x18k Datapak, 1xWidget software utilities pak (e.g. prints diary direct to printer), and Comms link (RS232). Complete with all documentation still under guarantee (Feb 89), worth over £200, accept £100. Tel. (0344) 861988 9-5 weekdays. with Master hence sale. Tel. (0246) 418646.

Z80 2nd processor for BBC B offers please! Surplus; 4 unopened boxes of 10 Datalife mini discs 5.25" 96tpi 25/4D offers please! WANTED: Information on how to set up an MX80 printer on a BBC B. Tel. (0553) 675845.

> Complete set of BEEBUG magazines to date. Offers? Sets of Acorn User and Micro User, free to a good home. MegaROM £65, Viewsheet £15, Novacad £15, CC Graphics ROM £10, Slave ROM £10, Commstar £10, Acorn ISO Pascal £25, AMX mouse with Super Art/Pagemaker for Master £35. Wordwise + on disc for Arc £15, WW+ handbook - P Beverley £7.50, 12 copies of Risc User discs + magazines £15. Tel. (0736) 63918.

> AMX Super Art £25. BEEBUG C with StandAlone Generator £25, EXMON II (ROM) £10, Acorn GO, Superior Speech (disc) £5 each, Mini Ôffice I (disc) £4, FORTH disc and manual £10, all with relevant documentation and accessories, Advanced User Guide £10, Birnbaum's Assembly Language Programming £5. Tel. (0702) 586536.

> For Sale: Centronics 737 printer. Best Offers! Tel. (0533) 714004 after 7pm or wk/ends.

BBC B issue 7 32k + word 40/80 switchable 720k dual disc drives with own PSU, excellent condition £330 o.n.o. Tel. (0923) 858202.

BEEBUG magazines, complete set & in mint condition, from first issue to current with indexes, vols 1 to 7 in blue binders, remainder loose. Offers please!, also other BEEBUG products inc. COMMAND etc. Tel. (0737) 556384.

Z80 co-processor £95, Opus DD 40/80 DS £72, Master Replay with Master to BBC B conversion £32, Dumpout 3 ROM £17, BEEBUG C with Stand Alone Generator £32, View Index £8, Double View(Master Version) £22, Philips Green screen monitor £50, double height plinth £18, also various manuals for sale. P&P inc. on all items. WANTED: Morley AA Board. Tel. 051-647 5367.

INVOICING & ACCOUNTS

The Account Book

Comprehensive small business accounts to trial balance. VAT approved. Absolutely the easiest program to use, with neat final books and hundreds of reports. No entry limits. "The Account Book gets first prize for both price and performance"- comparison of different business programs in Micro User-July '89. "A true user-friendly program. If you buy these packages you will not be disappointed"-Beebug-Oct '88 & Dec '89. Just £27.95.

The Invoice Program

Link with The Account Book or use independently. 700 customer database, Invoices, Statements, Stock presets, Debtor lists, Mail shot labels and loads more. &27.95. Purchase The Account Book and The Invoice Program together for £49.95.

Personal Accounts

Just Released. Multiple bank accounts, Direct Debits. The best report, editing and search facilities. £14.95.



BBC B Watford DDFS, 32k Shadow RAM and solderless ROM board with 16k battery backed sideways RAM £280, Mini Office II on cassette £5, ISO Pascal £35. (may split BBC add ons) Tel. 01-642 6412.

For Sale: CUB Monitor, needs attention, reasonable offers please. WANTED: Software,Hardware for BBC Compact. Tel. 01-954 5712.

WANTED: Juki 6100 Tractor feed. Tel. (03503) 222.

Acorn Hard disc podule £150 o.n.o. Tel. (0923) 51049.

Miracle WS2000 modem with DATABEEB communications ROM £45, Oxford Pascal ROM & disc £10, BEEBUG's Toolkit + ROM £15, Microline 80 printer £20, all with original packaging and manuals, please note ROMs are not compatible BEEBUG MEMBERSHIP

Send applications address below. A cheques) on a UK BEEBUG S £16 £24 £29	for membership II membership fe bank. Members f GUBSCRIP 90 1 year 00 00	renewals, me ses, including nay also subs FION RA (10 issues Rest of EL Mido Americ Els	mbersh overse cribe to TES) UK, jrope Ile Eas as & A ewher	ip quer las, sho RISC U BFPC & Eire st Africa re	ies and orde build be in p Jser at a spea BEEBU D, Ch.I	ers for back issur ounds sterling d cial reduced rate. G & RISC £25.00 £36.00 £43.00 £46.00 £51.00	as to the rawn (for USER
E3 BACK IS Volume 1 2 3 4 5	4.00 SUE PRIC <u>Magazine</u> £0.40 £0.50 £0.70 £0.90 £1.20	Tape 5" £1.00 £1.00 £1.50 £2.00 £2.50 £3.00	Disc 2015C 23. 24 24 24 24	3.5 " .50 .00 4.75	Disc - - £4.50 £4.75 £4.75	All overseas its airmail. We official UK subscription issues, but pl there will be charge for ou which requ Note that th	ems are sent will accept orders for s and back lease note that a £1 handling ders under £10 ire an invoice. ere is no VAT ir
6 7 POST	£1.30 £1.30 AND PAC	£3.50 KING t of p&p ridual item	£.	De UK EI	stination (, BFPO + Ch urope + Eire	First Item .I 60p £1 £2	Second Item 30p 50p £1

when ordering i See table opposite.

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

BEEBUG MAGAZINE is produced by BEEBUG Ltd.

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

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

of the Publisher, BEEBUG Limited.

CONTRIBUTING TO BEEBUG PROGRAMS AND We are always seeking good quality articles and ARTICLES 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 Please submit your contributions on disc or receipt of an A5 (or larger) SAE.

cassette in machine readable form, using "View", "Wordwise" or other means, but please ensure an adequate written description is also included. If you use cassette, please include a backup copy at 300 baud. In all communication, please quote your

membership number.

BEEBUG Ltd (c) 1990 ISSN - 0263 - 7561

Printed by Newnorth-Burt Ltd (0234) 41111

Magazine Disc/Cassette

APRIL 1990 DISC/CASSETTE CONTENTS

CURVE FITTING - A useful graph plotting program, which draws a curve through a set of points.

MUSIC PROGRAMMING IN AMPLE - A program written in the Music Composition Language 'Ample", which is explained in the first of our articles on programming Hybrid's Music 5000 system.

EDIKIT (Part 4) - The final three commands added to the EDIKIT ROM - *HELP EDIKIT, *SYSINF and

*VARLIST.

WORKSHOP: WRITING & COMPILER (Part 5) Run-time library and linker to go with last month's

ORDER OUT OF CHAOS (Part 2) - A repeat of last compiler. month's program CHAOS3 quoted in part 2 of this article.

A DATAFILE SEARCH UTILITY - A utility for searching text strings in database files, which is a development of

the *FIND utility from Vol.8 No.7. ADFS DESKTOP - Create a more exciting Archimedes-

style desktop for your BBC with this utility. WEAVING PATTERNS - A fascinating application which

simulates weaving patterns.

IMPROVE YOUR MASTER AND BEEB - Emulate a Compact keyboard and cure the bug in the Master's DFS MAGSCAN DATA - Bibliography for this issue (Vol.8 No.10). with these programs.

* MOON PATROL GAME - A Bonus item to provide you

* BONUS MUSIC ITEMS - One Panda and two Hybrid with hours of fun.



SH3

DIR

TEST-POL

SH3 GESER

GFINDD

ATODETS

RTSDUPCE

SH3 HE RUE

DEIMI

FINDD

"DISC) + 60P P&P (30P FOR EACH ADDITIONAL ITEM) ol.5 No.1, tapes since Vol.1 No.10) available at the same prices. Music Programming In Ample

		TE (5" & 3.5	This 1 tapes SI	ILCO TO	AC
	CONTRACT	E), E4.15 (S since Vol.	.5 10. 1, 441	OVERSE	AS
	FOR F3 50 (CASSLI	Jo.1. 3.5" disc and		0.5")	Casserie
	I THIS FOR LISC SINCE VOLS	10111	V	Disc (5" or 3.5 /	£20.00
-	sok issues (5.25 Ulse -	UK ONL	Tanatta	c30.00	c39 00
В	ackies		Cassello	250.00	Looi
		Disc (5"or 3.5 /	£17.00	£56.00	ing only please.
Г	PATES	\$25.50	c33 00	anlicable, Sterl	ing only i
1	ALLASCRIPTION NAME	120.00	200.00	actane as application	o per issue of the
	SUBSCHE (5 issues)	£50.00	- of VAT and P	ostage receipt of £1.7	U por
	6 months (10 issues)	a inclusive	eort	hscription on local	AL 1435.
6	12 months (10 is	Prices are	or 3.5" disc sur	thene H	erts AL
	14.00	muted to a 5.25	dors to:	Dead St.Albans,	
	can be	commuted individual on	Jatfield	Hoad, -	
	subscriptions can	iptions and FRIIG.	11/ паше		
	Cassette subt to run. All subort	BEEDO ,			

subscription left



THE LEAR BEEBUG SPRING OPEN DAY and SHOW

Everyone is welcome at: 117 Hatfield Road, St Albans, Herts. on

Sunday 22nd April 1990

An Acorn 'Qualified Dealer'

Beebug is one of the largest Acorn Dealerships in the country. You are very welcome to visit us on our Open Day and mini-exhibition. Acorn Computers themselves will be present, along with a number of other companies.

Wide Stock Range

We have a very wide stock range, from computers, printers and software, right through to spare parts, such as keyboards and power supplies. All will be available on sale.

Beebug and RISC User Magazines

We also publish the acclaimed Beebug and RISC User magazines for the BBC Micro, Master and Archimedes/A3000 computers. Back issues will be available at special prices. The editorial team will be onhand to discuss the magazines and their contents.

Beebug DTP

Ovation, the new **DTP** system for the Archimedes will be on demonstration, along with our **scanner**, **5** 1/4 disc **buffer** and our new **memory upgrades** for the A3000.

Demonstrations

Our technical team will be present to demonstrate and discuss items of software and hardware that might be of interest to you.

Instant 0% Finance

Mercantile Credit will be attending so we can offer instant 0% Finance on all new Acorn Computers, enabling you to take your new computer away with you on the day.

Special Offers

Special reductions and clearance offers on a wide range of items with many available in our "£5 bargain bin".

HOW TO FIND BEEBUG

By Car - St Albans is easily reached from A1 A5 A6 M1 and M25.

By Train - We are 10 minutes walk from St Albans City Station on the King's Cross to Bedford line.

Companies represented include:

Acorn Computers

A number of Acorn staff will be on hand to demonstrate and discuss the complete range of computers including the A3000.

Colton Software

Pipedream 3, the efficient spreadsheet/ wordprocessor/ database for the Archimedes will be on demonstration together with View Professional for the BBC B and Master.

EMR Ltd

Mike Beecher will be attending to demonstrate the popular Studio 24 Plus MIDI Music package for the Archimedes as well as his range of MIDI add-ons for the BBC and Master series.

Star (UK) Ltd

See the business and professional range of Star printers in action including the new XB24-10 colour printer. The range of Star laser printers will also be on demonstration.

Fourth Dimension

Fourth Dimension will be demonstrating their excellent range of games for the BBC Micro and the Archimedes...

Events & visiting companies subject to change without notice.

