INDEPENDENT NATIONAL USER GROUP

FOR THE BBC MICROCOMPUTER

BEEBUG NEWSLETTER

VOLUME 1 NUMBER 8 DEC 1982 /JAN 1983

CONTENTS

GENERAL CONTENTS

PROGRAMS

HINTS & TIPS

MEMBERSHIP NOW EXCEEDS 12,000

BRITAIN'S LARGEST SINGLE-MICRO USER GROUP

EDITORIAL

ACORN

ROM 1.2

It is now extremely likely that the first batches of the new operating system ROM (version 1.2) will be with Acorn by the time that you read these words. However, it seems equally likely that supplies earmarked for upgrades will not now be made available until February. The first batches are apparently (predictably ?) all destined for production machines. Because of this we will be telling you how to obtain a BEEBUG-negotiated ROM in the next issue rather than in this.

Memory Upgrade Components

A number of members have written to us to ask exactly what memory chips are suitable for upgrading the BBC machine. We put this to Acorn, and received the following response:

> There are four dynamic RAM chips that Acorn have approved for use in the BBC Microcomputer. These are as follows:
>
> HITACHI HM 4816-AP3 HITACHI HM 4816-AP11
> MOSTEK MK 4516-N10 FUJITSU MB 8118-10
>
> These ICs have been rigorously tested for use with the machine at Acorn; no other ICs have undergone such tests.

If you have been supplied with some other chip - say a Mitsubishi 8118-10, as some members have reported, it does not necessarily mean that it will not work. On the other hand, it is obviously safest to go for one of the four recommended ICs; especially if you ever need to make a claim under the guarantee. Our thanks to Acorn for this release. On the question of the validity of the guarantee after DIY upgrades, raised last month, we are still pressing for a printable statement. We are optimistic that the answer may indeed be a qualified yes.

BEEBUG

Firstly we must apologise to those of you waiting for the Mk2 versions of the Seikosha and Epson screen dumps. We are working on two machine code programs to meet this need, and will have these ready for the next issue.

We now (a little belatedly) have disc systems at BEEBUG, and as a first sortie into the world of BBC discs we have produced a disc review for this issue. We can now accept programs/articles on disc in 40 or 80 track, single or double-sided formats. It is our intention, of course, to continue fully to support the cassette system. If you do send us cassettes for any purpose, please supply a back-up 300 baud copy (type *TAPE3 <ret>), because we do have loading difficulties with a small number of cassettes submitted to us.

In order to cover increasing production and distribution costs, and anticipated increases in postal charges, the price of BEEBUG membership will be increased to £9.90 per year (and £5.40 for 6 months) as from 1st Jan 1983. It is our intention to freeze this price for at least a year. The magazine is currently 60% larger than the first issue (even ignoring advertising), and further expansion is envisaged in the new year.

BEEBUG SOFTWARE MANAGER

BEEBUG requires a full-time software manager based in St Albans, to run BEEBUGSOFT.

Duties will range from the evaluation and editing of programs for the software library to the organisation of their distribution and promotion. Applicants should have adequate programming experience and a good familiarity with the BBC micro. Salary will be according to experience.

Applications should be made in writing to the editorial address, and should include a curriculum vitae as well as the names of two referees. Please also give a telephone number if possible.

Applications close on the 15th January 1983.

RETRIEVING A 'BAD PROGRAM'

by Colin Opie

How many times have you lost a program and got the ubiquitous "Bad Program" message, and how many times have you lost programs because they refused to reload?

This article looks at the whole question of program loss, and presents a Recovery Program "Rescue" which may be used to salvage "Lost Programs". In the tests which we have made on Rescue, it has performed magnificently, recovering vast sections of badly garbled programs as if by magic.

The article and program are based on the ideas submitted by members in response to a plea in BEEBUG no 5 by S.P.Brooke who had been losing a lot of programs. We acknowledge those members at the end of the article.

Program Loss

I think that the best way to approach this problem is from a purely practical point of view in the first instance, and I will begin by looking at the use of the recovery program in some detail; leaving theoretical explanantions until the end.

SIMPLE RECOVERY

First type in and save ordinarily a copy of Rescue. This program may be used to recover from many types of bad program - including faulty cassette recordings, and bad programs already resident in memory.

We will assume in the first instance that you have a program on tape which will not fully load for one reason or another. First of all you must allocate space at the top of memory away from where the Bad Program will reside, for the recovery program. To do this, first make sure that you are in mode 7, and then type PAGE=&3800 'return' for a 16k machine, (or PAGE=&7800 for 32k). If you are not familiar with the hexadecimal notation, you may like to look at BBC Basics in this issue. Type NEW and load in the recovery program.

Now reset PAGE to &E00 (type PAGE=&E00... or PAGE=&1900 for disc machines), type NEW followed by *OPT 2,0. The latter operation tells the filing system to ignore errors and to continue to load the program (it cannot however be used when loading from disc).

Now attempt to load in your program and don't worry about any error messages that appear. Eventually the last block should be found and the loading operation will therefore cease, with the message 'Bad program' more than likely appearing. (If the last block cannot be found by the machine then you may have to use one of the ideas set out later in this article).

Now set PAGE back to the value you used in order to load the recovery program (ie. &7800 or &3800). Type RUN, and press return in answer to the prompt "TOP OFFSET?".

As the recovery program is running it will list out the line numbers it thinks it can find together with the location in memory where it found them. When the program finishes it resets PAGE to &E00 (or &1900 for discs) in order to get back to your 'recovered' program. You may now type LIST or anything else you like in order to make good the retrieved parts of your lost program.

In case you feel that the operation is complicated look again at the operations in the order that you need to execute them:

```
 0. MODE 7
 1. PAGE=&7800  (&3800 for 16k)   ;Reserve space
 2. NEW
 3. LOAD "RESCUE"                 ;Load recovery program
 4. PAGE=&E00 (or &1900)          ;Reset PAGE
 5. *OPT 2,0 (only for cassette)  ;Ignore errors
 6. LOAD "BADPROG"                ;(Use the real name!)
 7. PAGE=&7800  (or &3800)        ;Point back to the recovered program
 8. RUN                           ;and run it.
 9. Press return in answer to the prompt "TOP OFFSET?"
10. LIST                          ;This should list your recovered program.
```

If your Bad Program is already in memory because the problem arose during running, or development etc., rather than in loading; then just execute instructions Ø to 4 and 8 to 1Ø.

If the program as recovered is considerably shorter than the original, you may try to get more of it back by repeating steps 1 to 4 and 8 to 1Ø (whether the original problem was on cassette or not), but at the prompt "TOP OFFSET?" you should give a number that corresponds approximately to the number of bytes (or characters) that you think you have lost. This is obviously a trial and error process, and if you set the offset too high, you may get additional garbage program lines - though these can be edited out.

THE PROGRAM

The program is listed at the end of the article. Note that for disc systems, you should alter line 22Ø to set P%=&19ØØ rather than &EØØ.

The main work of the RECOVER procedure is carried out in the REPEAT..UNTIL loop. Line 14Ø is necessary in order to remove control codes from the corrupt lines themselves. Such control codes are changed by the program into a 'hash' for easy identification in subsequent editing. Line 15Ø is a catch-all line which traps a line exceeding 25Ø characters in length, by simply splitting the line into two or more parts. You could, if you wanted to, change the program so that it always generated the same Basic line number for this process. But the disadvantage is that you may end up losing valuable program lines by doing this. The recovery program does not pick on any line in particular so as to remain (reasonably) universally applicable. Lines 11Ø and 17Ø merely print out the line numbers found and their location in memory.

```
  1Ø REM Program recovery
  2ØINPUT"TOP OFFSET ",T%
  3Ø PROCinit
  4Ø REPEAT
  5Ø   PROCrecover
  6Ø UNTIL finished
  7ØPAGE=P%
  8Ø END
  9Ø DEF PROCrecover
 1ØØ ?line=&ØD: lenpos=line+3: count=1
 11Ø IF ?line=&ØD AND line?1=&FF THEN
finished=TRUE: ENDPROC
 12Ø PRINT (line?1)*256+(line?2);
 13Ø REPEAT
 14Ø   IF line?count<>&ØD THEN count=c
ount+1
 15Ø   IF line?count<>&ØD AND line?cou
nt<32 AND count>4 THEN line?count=35
 16Ø   IF count>25Ø THEN line?(count+1)=&ØD
 17Ø UNTIL line?count=&ØD
 18Ø PRINT ~line
 19Ø line=line+count: ?lenpos=count
 2ØØ ENDPROC
 21Ø DEF PROCinit
 22Ø A%=TOP-2+T%: ?A%=&ØD: A%=A%+1: ?A
%=&FF: P%=&EØØ:REM*******CHANGE TO P%=&
19ØØ FOR DISCS
 23Ø line=P%: finished=FALSE
 24Ø ENDPROC
```

ADDITIONAL NOTES AND HINTS

1. If on trying the recovery process on a cassette that loads badly, you find that you have been unable to recover very much, try loading the bad tape again! There is always the chance that a second 'reading' will produce a better 'result'.
2. No Block zero
   The best way to recover a lost block zero is to use the T.BUG program in BEEBUG no 7. The alternative solution is to provide some other block zero and then load in the rest of the tape as normal. One method is just to save any old rubbish with the same name as your corrupt tape. Make sure that it is at least two blocks long. You load in the rubbish tape, stop it half way through the second block, then replace it with the corrupt tape. A problem with this method is that you will have to patch up the links, (using the recovery program) if they do not match. The chances are they will not! A better approach is to type in enough of the original program as required to get these first few blocks. Then save this, and reload it. Stop the recorder half way through block 2, and insert the bad tape from the start. The main problem here of course is that you may not have an accurate listing of the lost program.
3. No Final Block
   In this case you could load in as much of the program as possible and then

manually look (via indirection operators) at the memory around TOP, and insert a new 'end of program' marker in the appropriate place. - See notes at the end of the article on the theoretical basis for this. Alternatively, you should be able to use the TOP OFFSET facility described above.

4. Saving a Bad Program

If you have a Bad Program and you do not have time to sort it out, you can still save it using the following command:

```
*SAVE "name" E00 6000
```

The last hex number given can be anything you like as long as you feel that it is big enough to reach the end of your software. You can re-load it with:

```
*LOAD""
```

5. Using Backups

To reduce the likelihood of being inconvenienced by a Bad Program in the future, it may well be worth considering the following:

a) Keep adequate backups on tape as you develop your software. This is a slow process with cassettes but you know the alternative.
b) Always SAVE a copy (or two) at 300 baud for extra security at the end of the day.
c) Always verify your saved programs with *LOAD" " 8000 (see BEEBUG no 6 for details).
d) Keep your working tape and your 'master backup' tape separately.
e) Leave sensible gaps between programs on the same tape. Better still if you can afford it, use two tapes per program.
f) If you are using discs, make frequent backup copies, perhaps alternating between two discs.

THE THEORY BEHIND RESCUE

To understand what is happening we really need to jump in at the deep end and look at the internal structure of a BASIC program. Each BASIC line consists of an <0D> character (ie 0D hex or 13 decimal or 'return') followed by a 16-bit line number (most significant bit first) followed by a <line length> byte, and finally followed by the actual line:

```
Eg:         10 REM**
            20 REM
            :
            :

becomes:-
            <0D><00><0A><08><20><F4><*><*>
            <0D><00><14><06><20><F4>
            :
            :
```

Note that <0A> is hex for 10 - ie line number 10, that <20> is the ASCII code (in hex) for a space and <F4> is the internal code (or token) for 'REM'. The line length value includes the number of bytes in the whole encoded line, not just the length of the line itself.

The end of the program is signified by having the 'line number' &FFFF after the <0D> code. (In practice, setting just the most significant bit of the 'line number' to &FF will suffice).

When for example a LIST command is executed, the operating system runs through the memory checking to see if all the 'line length' codes are correct. If they are not then the message 'Bad program' appears. The task is therefore to fool the system into thinking it has a perfectly good program, even though you know that it is not as good as it used to be! In essence this is what the recovery program in this article achieves.

We are most grateful to the following for the ideas contained in this article: Colin Chalmers, A G Williams, Arnold Stewart, Cedric Marshall, Heath Rees, G L Wills, Terry Bromilow, Gavin Craig, Michael Farrell, A Suggett, Paul Mudditt, David Nichols, Mark Tilley - and to S P Brooke who posed the question originally.

DISC SYSTEM REVIEW

by Sheridan Williams

Here it is at last, our long awaited disc review. The topic of discs has given rise to more questions than any other topic recently, and we have long recognised its importance to members. However, we have been unable to do a disc review before because of the difficulty in getting hold of all the components that make up a working disc system, namely:

Disc Filing System (DFS) chips.
Machine Operating System version 1.0 or later,
(See the editorial for the supply situation on OS 1.2).
A disc interface.
The disc drives themselves.
A 'Utilities' disc.
A disc manual.
Power output socket on Beeb power supply.
Not all drives require this because they have their own built in mains supply.

Note that the Beeb external supply (switched mode) will only give 1.25A at 12v and 1.25A at 5v. Some drives require more than this when used in pairs.

Supply

There are at present just two official channels for getting a disc system (though disc drives are in ready supply). You can buy a model B with disc interface from Acorn, or you can get a model A or B upgraded at a dealer. The reason that no one offers unofficial disc upgrades is that the only supplier of the essential disc operating system (currently only in EPROM) is Acorn themselves. The BBC disc system is standard though, and once you have a working system, there is a wide range of unofficial drives that may be used with it. There is at present a further obstacle to unofficial purchase of even the drives themselves. If you buy a model B with disc interface from Acorn (or get an official upgrade), Acorn will not supply you with a manual for the operating system that they have sold you (they won't even supply a utilities disc) unless you buy one of their disc drives. Clearly it is not really on to refuse to provide a manual for the DOS unless you buy extra parts. We have discussed this with Acorn, and they are currently considering the possibility of selling the DFS manual and 'Utilities' disc as a separate item; though no final decision has yet been made.

The easiest component to get hold of is the disc drive itself, and we have assembled 5 different drives for review.

The disc drives were obtained from the following suppliers:
Microage Electronics, 135 Hale Lane, Edgware, Middlesex HA8 9QP (Tel: 01-959 7119)
Microware, 637a Holloway Road, London. N19 5SS (Tel: 01-272 6398/6237)
Acorn Computers, via Vector Marketing. Tel 0933 228953.

The Review

We review 5 drive units. Details of their specifications are given below:

Drive 1 - BBC Disc drives. Price £265 Single drive, Single-sided, 40 tracks, 100k capacity. Drives are made by Olivetti. Size: 155x241x97mm.
Drive 2 - Microware type ZL141B. Price: £155 Single drive, Single-sided, 40 tracks, 100k capacity. Drives are made by Control Data. Size: 155x220x100mm.
Drive 3 - Microage 'BBC compatible single disc drive'. Price: £235 (5% discount to

BEEBUG members off this price). Single-sided, 40 tracks, 100k capacity. Drives are made by Teac. Size: 177x289x162mm.

Drive 4 - Microage 'BBC Slimline Dual disc drive'. Price:£799 (5% discount to BEEBUG members off this price). Double-sided, 80 tracks, 800k total capacity. Drives are made by Mitsubishi. Size: 153x241x94mm.

Drive 5 - Microware type ZL292. Price:£661 Two drives, Double-sided, 80 tracks, 800k total capacity. Drives are made by Control Data. These drives have their own built-in power supply. Size: 153x241x94mm.

All the drives came with a 'Utilities' disc, and those from Acorn and Microage also came with a disc filing system manual.

One of the nice features of drive 5 was that Microware supplied a switch to switch between 40 and 80 track operation on the left hand drive. This increased the cost to £690, but this seemed worthwhile from our personal point of view, since it is vital for us at BEEBUG to be able to read as many different types of disc as possible.

Another interesting fact that came to light is that both of the large capacity disc drives reviewed (systems 4 and 5) were capable of holding significantly more than the BBC micro could handle, for example system 4 could hold 1 Megabyte of data on each disc, and system 5 would hold 700k per disc. The BBC micro cannot handle this amount, so the disc drives are being under used. We will report on this in more detail in a future issue.

Comments on the Drives

All the drives had fairly short power and data cables. They were each approximately 80cm long. This means that the drives have to be reasonably close to the computer.

Drives 1-4 all used the power-out supply from the Beeb, drive 5 has its own built-in power supply.

Drive 1 This drive is supplied by Acorn for the BBC micro and is in a buff coloured case. When the flap is released the disc is automatically ejected. The discs are inserted horizontally.

Drive 2 The colour of the case is slightly lighter than the BBC drives. The disc is not ejected when the flap is released. The discs are inserted vertically.

Drive 3 This is enclosed in a light grey case, and the disc is again not ejected when the flap is released. The disc is inserted horizontally. There is a switch that locks the flap closed to prevent accidental undoing of the flap - a good idea.

Drive 4 These are also enclosed in a grey case, the remarkable fact being that the case is identical in size to Drive 4 above, but it holds two disc drives. The discs are inserted horizontally, and are ejected on releasing the flap.

Drive 5 These are contained in a cream coloured case. They are significantly larger than the Drive 4 but still not obtrusively so. The discs are inserted vertically, and are not ejected on releasing the flap. A switch can be fitted to enable the left hand drive to work in either 40 or 80 track mode.

During tests over a period of weeks, two of the five drive systems produced drive error faults:

Drive 1 (BBC) had a jammed head cam, which prevented any further use of the drive until the cam was released manually. This could be made to happen by trying to access a track greater than 40. (Eg by using 80 track discs by mistake).

Drive 2 (Microware) produced erratic &0E errors, we were unable to ascertain the cause.

BENCHMARKS

I used a series of 'benchmark' tests aimed at showing the efficiency of a series

of disc operations. The tests included:-

1. Open and Close a file 10 times.
2. Open a file, then write 100 records, each of 255 characters, then close the file.
3. Open a file, then write the 100 records, but this time in reverse order (ie. from record 100 to record 1, then close the file.
4. Open a file, then read the 100 records, created in test 2 or 3, then close the file.
5. Identical to test 4 except the records are read in reverse order.

BENCHMARK TEST RESULT TIMES

Test 1	Test 2	Test 3	Test 4	Test 5
2.8	62	192	56	99

(all times in seconds)

These benchmarks conform to the set used previously by "Personal Computer World" magazine when evaluating disc systems. Test 3 is interesting, because if the file doesn't already exist, then it will take an extra 40 seconds for the DFS to initialise it.

These times are generally rather slow, though there is a system of links (explained in the Disc System User Guide) which may be set for faster drives. Changing these links would be expected to give better timings, since the system as supplied has no links set, and corresponds to the slowest setting. This is appropriate for the Acorn-supplied Olivetti drives.

Links (Found on the keyboard pcb, just below the right hand shift key).
Links are numbered 1-8 from left to right. Links are made north-south.
From Acorn fact sheet:

Drive	Step time	Settle time	Head load	Link-3	Link-4
Tandon (fast)	4	16	0	1	1
Tandon/Shugart	6	16	0	1	0
MPI	6	50	32	0	1
Olivetti	24	20	64	0	0

All times are in milliseconds, and
1=link made, 0=no link.

Note the link settings are only checked on power-up.

Choice of Disc Media

It is important to buy the correct discs for the disc drives. In all cases these are soft-sectored discs. They should be double-density, and double-sided if you are using double-sided drives. You should also buy discs that have been certified for 80-track use if you are using an 80-track drive such as systems 3 and 6 in our review.

[WARNING: many people try to economise when purchasing discs, for example they use single sided discs because they find they work in double sided drives. This is fraught with danger, because all discs are made on the same production line, and if they had passed the double sided tests they would have been sold as such. This means that you use them at your peril. Remember that the contents of the disc can be worth perhaps hundreds of times the actual cost of the disc itself. Some people even cut the disc to make it symmetrical so that it can be inserted the other way up. This is also short sighted because the disc revolves the other way in its jacket having just lapped itself in to the cleaning material inside its case, contra-rotation will wear the disc faster.]

D.F.S. Commands

As I mentioned earlier, Acorn's policy is not to supply the disc operating instructions unless you buy their own disc drives. Some suppliers are supplying

their own manuals. However there is an easy way to find out quite a lot about each command: you just type: *HELP DFS (DFS stands for Disc Filing System).
The message that follows uses descriptions such as:
COPY <src drv> <dest drv> <afsp>
where

<src drv>	=	source drive
<dest drv>	=	destination drive
<fsp>	=	filename
<afsp>	=	filename (can include wildcards)
<drv>	=	drive number
<old fsp>	=	old filename
<new fsp>	=	new filename
<title>	=	any 7 alphanumeric characters

If the variables are enclosed in < > then they are compulsory, however if they are enclosed in ( ) then they are optional.
Disc drives are numbered 0, 1, 2, 3 where drive 0/2 and 1/3 are on reverse sides of one another.
The commands must be prefixed with an *, for example: *COPY 0 1 TEST here the file TEST will be copied from drive 0 to drive 1.

Disc System User Guide

This is the title of the manual you are supplied when you buy the BBC's official disc drives. It is very well written indeed. The first few chapters are devoted to explaining what a disc system is. Then each of the commands is explained in detail. Finally there are chapters on Random Access Files, Using the filing system in Assembler, Technical Information, and an Index. Beware though of the errors in the error codes!

Conclusion

It is not possible to make a choice or recommendation from amongst these systems, because we have had them for such a short time. We cannot, at present, say which will prove the most reliable in the long term; and the two faults that occurred should not be taken as typical. We will be reporting on our long term experiences with drives 1 and 5 in a future issue.

It is interesting to quote the figures given in the 'Control Data' information sheet on Drives 2 and 5. The mean time between failures is 8,000 hours, the service life is 5 years, no preventive maintenance is required; data reliability for recoverable errors is not more than 1 bit per 1,000,000,000 bits read.

Our thanks are especially due to Tony Latham for his help.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

MACHINE CODE AT &E00

If a machine code program is required to reside at &E00, preface it with the two bytes &0D and &FF, and make &E02 the execution address. Otherwise BREAKing will destroy the first two bytes.

Mark Tilley

MERGING VDU CALLS

Peter Vincent reminds us that VDU commands can be merged, for example:
VDU 19,0,3,0,0,0
VDU 19,1,4,0,0,0
can be written as: VDU 19,0,3,0,0,0,19,1,4,0,0,0

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

Inkey Timer

A good way to achieve a time delay in a Basic program without resetting the TIME variable, is to use the INKEY() function. Thus: 100 X=INKEY(200)
will give a two second time delay. One feature of this delay is that it can be cut short by pressing any key.

Multi-Program Storage

It is possible to store any number of programs in the BBC machine simultaneously. This is accomplished by selecting a new value of PAGE for each program. For example if you type in or load a program ordinarily on a cassette-based machine it will sit at &E00 in memory (since &E00 is the default value of PAGE).

To enter another program, just type PAGE=&2000 (say), then NEW, and type in or load, and run your second program. To get back to the first just type PAGE=&E00.

In deciding where to locate such programs, refer to the memory maps in the User Guide, and remember that they should be well above the space taken up by programs lower in memory, and you should allow additional space for variable storage of the lower program.

BREAK, OLD and NEW

Mark Tilley supplies notes on these three as follows:-

1. On pressing Break:
   PAGE is set to &E00 (&1900 with discs). NEW is performed.
2. NEW causes the following:
   &0D is stored at the location given by PAGE, &FF is stored at the next location.
   The rest of any program in memory is left intact.
   The two bytes &0D, &FF are used by Basic to signify the end of a program.
3. OLD has the following effect:
   Zero is stored at PAGE+1. This overwrites the &FF generated by NEW, and so recovers the existing program. The zero becomes the most significant bit of the first line number.
   NOTE that if the first line number was greater than 255, it would be replaced by an entirely different value.

BASIC PROGRAM COMPACTER (16k)

Program by Graham Taylor

Text by Adrian Calcraft

This is a program which will be of particular interest to owners of 16k machines, or anyone developing programs and running out of memory space; or indeed anyone who uses large amounts of documentation in REM statements during program development, but who would like a bit more memory when the final version of the program is actually running. The object of Pack is to reduce the amount of space your program occupies. It achieves this by scanning each line of a program in turn, removing unnecessary spaces, blank lines and REM's, and adjusting the values of the line length markers accordingly.

The program in its current state is designed for cassette use. If you have a disc interface fitted, please see the 'DISC NOTE' below.

Setting up Pack for the First Time

The procedure for typing in and running Pack for the first time is a little unusual, and care must be taken to follow the instructions carefully. In particular you must not add anything (except spaces if you wish) to the program that you type in in its original form. Otherwise it will not fit in the extremely small space provided for it. Once the final copy of Pack has been saved, it is extremely easy to use on all subsequent occasions. Note that Pack may not be used in conjunction with the cassette bugs fix published in BEEBUG No 4, so verify all SAVEs with *LOAD"" 8000.

The program should be set up for the first time as follows:-

1. Type in the program exactly as listed.
2. Do not RUN this program, but save a copy of it as a backup, just in case you hit a problem in the next few steps.
3. Program the function keys f0 and f1 by typing the following:

```
* KEY 0 PAGE = &C00|M  RUN|M  <return>
* KEY 1 PAGE = &E00|M        <return>
```

   Note that the vertical slash character "|" is immediately to the left of the left cursor key.
4. Take care not to press either of the function keys yet at this stage.
5. Run Pack. You will get two prompts: R? and I? Press "Y" to both.
6. Pack is now packing itself, and relocating itself down below &E00. This is smart. When it has done this it will clear the screen. If you get "No Such Line" error messages, you have probably mistyped Pack (though do not worry if you added some extra spaces, it will automatically remove these). If you get an error at this stage you should clear the whole machine and use the backup tape to correct Pack, and go back to Step 3.
7. Type PAGE = &C00 Return.
8. Alter line 10 to R%=&E00
9. Make your final functional copy of Pack by executing *SAVE"PACK2" B00 E00 This stores the compacted Pack plus the user key buffer.

Using Pack

Compared to the above performance, using Pack is relatively easy.

1. Press Break. (For discs type *TAPE<return> PAGE=&E00<return>)
   Type *LOAD "" Return, and load the compacted version of Pack (saved in 9 above).
2. Load the program to be packed.
3. Press function key f0. This automatically resets PAGE and Runs Pack.
4. You will then be prompted with R? If you answer "Y", all REM statements will be removed. If you have GO TO lines in your program which point to a REM line, you will get a No Such Line error when you run the compacted program later. But this

is easily fixed by re-inserting the lost lines.

5. You will then be prompted with I? If you anser "Y", all IF statements will be com pacted - so make sure you have put in the "THENs" rather than just leaving spaces. For example if you have the following:
   10 IF A = B C = D
   Pack will remove the spaces, and give the equivalent logic of:
   10 IF A = BC = D
   which has a totally different meaning. To avoid this you should insert a THEN thus:
   10 IF A = B THEN C = D
6. Pack will then compact according to your instructions, removing unnecessary spaces and line numbers where appropriate. Even if you answer "N" to both prompts, it will still compact the rest of the program.
7. When the task is completed the screen will be cleared, and you should press function key f1. You may now LIST, RUN or SAVE your compacted program in the normal way.
8. If you need to edit the compacted program, you should do this with extreme care, since you will need to reinsert certain spaces when using the copy editor. The reason for this is that Basic words such as AND, THEN etc preceded directly by variable names are NOT recognised by the Beeb. Thus IF ATHEN B is read as the variable ATHEN rather IF A THEN B. Providing you do not use the copy editor, no problems of this kind arise with compacted programs because the Basic words are already stored in their so-called 'tokenised' form.

How Pack Works

Pack looks at the tokenised version of a program as it is stored in memory, removes spaces, adjusts the line length byte and rewrites the line starting from where the original program was stored. It is easier to understand this if we consider a mini 1 line program. 10 PRINT A*10 If this was typed in as a statement, the computer would store the following decimal data to represent it.....

13 0 10 11 32 F1 32 61 32 49 48 13 255

BYTE 1....13	Indicates the beginning of a new line.	BYTE 8....65	ASCII code for A.
		BYTE 9....42	ASCII code for a *.
BYTE 2.... 0	High byte of line number	BYTE 10...49	ASCII code for character "1"
BYTE 3....10	Low byte of line number	BYTE 11...48	ASCII code for character "0"
BYTE 4....11	Line length. Total length of this line..11 Bytes	END OF 1ST LINE	
		BYTE 12...13	Indicates the end of the line.
BYTE 5....32	ASCII code for a space.		
BYTE 6...241	Token for PRINT.	BYTE 13..255	Status byte..255 shows end of program.
BYTE 7....32	ASCII code for a space.		

As Pack operates on these tokens it is possible to strip out spaces which the computer would normally require simply for syntax. To see these tokens as stored in the computer is very simple using the "Indirection Operators" as described in the User Guide pages 409-413 eg. To see bytes in memory at &E00..type A=&E00<return> (A=&1900 for disc systems) then type PRINT A?0 or PRINT A?1 ...etc. Alternatively the Memory Display Utility in BEEBUG no 6 will give a fuller view of Basic programs. See also "Points Arising" in this issue.

Use of Variables

A%...Flag for REM Packing	Z%...Current byte
B%...Flag for IF packing	L%...Address of input program
L%...Line start of input program	R%...Address of output program
M%...Line length of input program	C%...Rem flag
P%...Line pointer of input program	D%...Data flag
R%...Line start of output program	Q%...Inquotes flag
S%...Line length of output program	I%...If flag
Y%...Line pointer of output program	Pack is designed for Basic programs only.

```
   1Ø PRINT"R?" :A%=GET :PRINT"I?" :B
%=GET:L%=&EØØ:R%=&CØØ :S%=Ø:M%=Ø
   2Ø R%=R%+S%
   3Ø L%=L%+M%
   4Ø ?R%=&D:R%?1=L%?1:IF L%?1=&FF TH
EN CLS:END
   5Ø R%?2=L%?2 :M%=L%?3:S%=M%: IF A%
=&59 AND L%?4=&F4 THEN3Ø
   6Ø C%=FALSE:D%=FALSE:Q%=FALSE:I%=F
ALSE:Y%=4
   7Ø FOR P%=4 TO M%
   8Ø Z%=L%?P%:IF Z%<>&F4 THEN 1ØØ
   9Ø C%=TRUE:IF A%<>&59 THEN D% =TRU
E:C%=FALSE
  1ØØ IF Z%=&DC THEN D%=TRUE
  11Ø IF Z%=&22 THEN Q%=NOT Q%
  12Ø IF B%=&59 THEN 14Ø
  13Ø IF Z%=&E7 THEN I%=TRUE
  14Ø IF NOT I% GOTO 16Ø
  15Ø IF Z%=&8C OR Z%=&E5 OR (Z%=58 A
ND NOT Q%) THEN I%=FALSE
  16Ø IF Q% OR D% OR I% THEN PROCC:GO
TO 18Ø
  17Ø IF L%?P%=&2Ø OR C% OR (P%=M% AN
D P%?L%=&3A) THEN S%=S%-1 ELSE PROCC
  18Ø NEXT
  19Ø IF S%<5 GOTO 3Ø ELSE R%?3=S%:GO
TO2Ø
  2ØØ DEFPROCC:R%?Y%=L%?P% :Y%=Y%+1:E
NDPROC
```

DISC NOTE

As it is currently developed Pack is unsuitable for use with discs. We expect to publish a disc version in a coming issue, and for now disc owners could type *TAPE then PAGE=&EØØ<return> followed by NEW before using the program from cassette. You must also remember not to press 'Break'. If you are thinking of developing your own disc version of Pack, you will need to store it at the top end of memory, because the space below &EØØ currently used to store Pack is corrupted by disc use.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

OSBYTE &87

Mr K. Turner has pointed out that the OSBYTE call with A=&87 works on his Ø.1 OS. It says in the User Guide on page 432 that it is only implemented in version 1.Ø.

BBC BASICS PART 3

by David Graham

This month I shall be using the ideas of memory maps developed last time to show how to increase the space available for programs in a model A or B by 75Ø bytes, and secondly to show how to load the user keys, and user defined characters without disturbing programs already resident in the machine. This will entail the use of *LOAD and *SAVE. Lastly a brief explanation of hexadecimal system will be given.

Both the above techniques are best understood by looking at the memory map on page 5Ø1 of the New User Guide. This gives a more precise allocation of the first 3584 bytes of memory in the Beeb than the one which we gave last month. Our simplified map on page 14 of BEEBUG no. 7 allocated the whole space to "Operating System Use" - though in fact the area is used for a multitude of purposes. The three most interesting subdivisions within this are the three at the top - namely:

	&EØØ	3584
Space for user supplied routines		
	&DØØ	3328
User defined character definitions		
	&CØØ	3Ø72
User defined function key definitions		
	&BØØ	2816

The column of addresses to the right are ordinary decimal numbers, and by subtracting them, you can see that each of the three memory areas is 256 bytes wide. This number of bytes - 256 - is called a page (nothing to do with "paged ROMs") - so each of the three memory areas is of one page capacity.

When you set up the user defined keys (see BEEBUG no. 4 for full details), information representing your key sequence is placed in the bottom of these three pages; and when you define new characters (see BEEBUG no. 2 page 17 for further details) the data for this is stored in the centre page.

There are two reasons why knowing about this can be of immediate practical use:-

Increasing Available Memory

On a cassette based model A or B, the memory available for Basic programs begins at location 3584, immediately above the highest of the three pages described above. In a Model A there is considerable pressure on memory, especially if you wish to use graphics modes 4 or 5, because these take 1Øk of memory. Since Basic only starts at 3584 the amount of memory actually available for Basic is 256Ø bytes.

16384 - 3584 - 1Ø24Ø = 256Ø

(since 16k = 16 x 1Ø24 = 16384, and 1Øk = 1Ø x 1Ø24 = 1Ø24Ø)

256Ø bytes is an extremely small amount of usable memory.

If you are not using the three pages of memory mentioned above, then you can, by a simple process, re-allocate these for Basic use. You do this by changing the value of the so-called "pseudo variable" called PAGE (see BEEBUG no. 2 page 15 or the New User Guide page 317 for further details). If you type PRINT PAGE "return" you will normally get the result 3584 (64ØØ on a model B with disc interface). This is the normal default value of PAGE, and PAGE defines where the machine starts to store its Basic programs.

To increase the available memory by three pages, simply type PAGE=2816. Now any Basic programs typed in from the keyboard, or loaded in from tape will be stored lower down, so saving much space. You cannot of course use the user keys or the user defined characters. Moreover you should not attempt to load the "Bugs Fix", because

that normally resides in the highest of the three pages. Of course, if you want to use the user keys, but not the user defined characters, you could set PAGE to 3Ø72 etc. Note that pressing Break automatically resets PAGE, so you will get a "bad program" message until you reset it to 3Ø72 (or whatever value you have used).

You may come across two other reasons for changing PAGE, though we only have the space to mention these briefly at present. It is common practice to set PAGE to a higher value than 3584 in order to reserve space for machine code subroutines that will not fit in the single reserved page at 3328. Secondly, it is possible by altering PAGE to allow the machine to store any number of Basic programs simultaneously. Each can be run by resetting PAGE to the appropriate value before executing RUN.

Transparent Loader

Knowing where the user defined key definitions and the user defined character definitions are stored in the machine allows us to load them from tape without disturbing a resident Basic program. This is the principle behind the Transparent Loader article in BEEBUG no. 7.

The idea is to use the commands *SAVE and *LOAD (see New User Guide pp 392-3 for further details). *SAVE allows you to store the contents of a block of memory, and *LOAD allows you to load it back in, either where it came from, or elsewhere. One of the possible formats of *SAVE is as follows:

*SAVE"KEYS+CHRS" BØØ DØØ

This will save as a file called "KEYS+CHRS" the contents of memory locations 2816 to 3328 - that is to say, the two pages storing the function keys and user defined characters. You will notice that the expressions BØØ and DØØ reflect the &BØØ and &DØØ in the memory map on page 5Ø1 of the User Guide. These are so-called hexadecimal equivalents of the numbers 2816 and 3328. For more on this, see the hexadecimal notes at the end of this article.

The way to use *SAVE in this context is first to set the user keys and characters that you require (using Basic), then perform a *SAVE as above. Then they may be transparently loaded any time you wish using *LOAD without disturbing Basic programs already resident in the machine. It is worth working through a concrete example just to see how this operates in practice.

First program some of the user keys to suit your purposes (see User Guide page 141 or BEEBUG no. 4 page 5 for details) and define some user characters using the VDU 23 command (see BEEBUG no. 2 pp 9 & 17). Alternatively running the following program will define 3 of the keys and character 224.

```
1Ø  REM sets KEYS Ø,1 and 7 and CHARACTER 224
2Ø  *KEY Ø  RUN|M
3Ø  *KEY 1  L.|M
4Ø  *KEY 7  MO.7|M
5Ø  VDU 23, 224, &6C,&FE,&FE,&FE,&7C,&38,&1Ø,Ø : REM HEART
6Ø  END
```

If you RUN this, function key fØ will RUN a program, Key f1 will LIST, and Key f7 will change to Mode 7. Character 224 is defined as a heart, so that subsequent commands of VDU 224 or PRINT CHR$ 224 will print the shape (providing you are not in mode 7!). You can save the key and character settings (without the Basic program) by typing:

*SAVE"KEYTEST" BØØ DØØ

Now whatever Basic program you have in your machine, you can just type *LOAD"KEYTEST" to reset the keys and character(s) previously set - all without disturbing the resident Basic. See BEEBUG no. 7 page 23 for a more elaborate transparent loader. You cannot of course use the transparent loader if you have set PAGE down at 2816 to save memory.

Hexadecimal Notes

The hexadecimal numbering system provides an alternative to the more usual decimal system, and is well suited to use in connection with computers. Computers in fact use so-called binary arithmetic, and hex can be considered to be a more humanised version of binary (using base 16 instead of binary's base 2 system). The table below gives the first few hex numbers with their decimal equivalents.

DEC	HEX	DEC	HEX	DEC	HEX
Ø	Ø	1Ø	A	2Ø	14
1	1	11	B	21	15
2	2	12	C	22	16
3	3	13	D	23	17
4	4	14	E	24	18
5	5	15	F	25	19
6	6	16	1Ø	26	1A
7	7	17	11	27	1B
8	8	18	12	28	1C
9	9	19	13	29	1D

Hex is a so-called base 16 system, meaning that after the count has got beyond 15, you start to use 2 digits. Decimal is base 1Ø, and we use two digits after 9, and 3 digits after 99 and so on. As you can see, hex makes up the missing numbers by using letters.

The Beeb has a handy hex to decimal calculator built in, and if you type PRINT ~n where n is any decimal number, you will get the result in hex. To do the reverse type PRINT &n, where n is in hex. Note that the "~" character (called 'tilde') is two to the left of the left cursor control key.

The following short program will convert decimal numbers into hex:

```
1Ø REM Decimal to Hex Converter
2Ø INPUT "Input decimal number", number
3Ø PRINT "Hex equivalent = " ; ~ number
4Ø GOTO 2Ø
```

You can get some idea as to how convenient hex is when dealing with computers when you realise that each byte of memory can store 1ØØ hex (256 dec) different values, and that a page of memory is 1ØØ bytes in hex. That is why the three pages under discussion above began at round numbers in hex - ie. at CØØ, DØØ and EØØ. This is a good deal easier to remember than their decimal equivalents; and you can just type PAGE=&EØØ in place of PAGE=3584. (In BBC Basic the ampersand (&) is used to signify that a hex number follows). 1k in decimal is 1Ø24; and in hex it is again a round number - 4ØØ; and 16k, which is 16384 in decimal is &4ØØØ.

Lastly, there is a hex readout on the Beeb that you may have watched hundreds of times without appreciating its full significance. I refer to the loading and saving data printed to the screen each time you use the cassette system. The number counts that appear are in hex, and the count given is the number of blocks, and each block contains &1ØØ bytes of data. Thus a program of 8 blocks would take up 2k of memory, and so on.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

ABBREVIATED NEXT

Mr A.R.Jones has discovered that when several NEXTs are required, then commas can be substituted. Eg.

```
10 FOR A=1 TO 2:FOR B=1 TO 2:FOR C=1 TO 2:PRINT ;A;B;C:NEXT,,
```

PROCEDURE/FUNCTION LIBRARY
FILLED CIRCLE DRAWING (16k/32k) by Bobby Hesselbro

In graphical programs there are frequently extensive requirements for filled-in circles of different sizes. Some program authors find that the delay while drawing is so substantial that it is necessary to blank the screen while the drawing takes place. Filling with triangles, using angles increasing to 360 degrees is very slow if good definition is required. Utilizing a 'mirror' function (and drawing two semi-circles simultaneously) such as in Example 1 the speed is greatly improved, ranging from 0.9 sec to 2.8 sec depending on size. The example employs steps of 9 degrees, a reasonable minimum for good appearance. The parameters for PROCcircle in Example 1 are:

X,Y - Co-ordinates of circle centre.
R - Radius
F$ - A string to specify filled or unfilled drawing.
Use "F" for filled and "U" for unfilled.

The procedure begins at line 1000, and is called by a short test routine on lines 5 to 70.

```
    5 REM Ex1 - Mirror method
   10 MODE 1:REM MODE 5 for 16k
   20 FOR radius%=500 TO 25 STEP -25
   30 TIME=0
   40 GCOL0,radius%/25+1
  :PROCcircle(500,500,radius%,"F")
   50 PRINTTAB(35,30-radius%/25)
  ;TIME/100
   60 NEXT
   70 END
 1000 DEFPROCcircle(X,Y,R,F$)
 1010 VDU29,X;Y;
 1020 PROCinit
 1030 FOR ANGLE=S TO45 STEPS
 1040 XX=FNX(ANGLE-S)
 1050 YY=FNY(ANGLE-S)
 1060 X=FNX(ANGLE)
 1070 Y=FNY(ANGLE)
 1080 MOVE 0,0:MOVE XX,YY:PLOTK,X,Y
 1090 MOVE 0,0:MOVE XX,-YY:PLOTK,X,-Y
 1100 MOVE 0,0:MOVE -XX,YY:PLOTK,-X,Y
 1110 MOVE 0,0:MOVE -XX,-YY:PLOTK,-X,-Y
 1120 MOVE 0,0:MOVE YY,XX:PLOTK,Y,X
 1130 MOVE 0,0:MOVE YY,-XX:PLOTK,Y,-X
 1140 MOVE 0,0:MOVE -YY,XX:PLOTK,-Y,X
 1150 MOVE 0,0:MOVE -YY,-XX:PLOTK,-Y,-X
 1160 NEXT
 1170 ENDPROC
 1180 DEFFNX(ANGLE)
 1190 =R*COS RAD ANGLE
 1200 DEFFNY(ANGLE)
 1210 =R*SIN RAD ANGLE
 1220 DEFPROCinit
 1230 IF F$="F"THEN K=85 ELSE K=5
 1240 NS%=5
 1250 S=45/NS%
 1260 ENDPROC
```

A still faster procedure which does not seem to be in use yet involves first drawing the inscribed square and then filling in the rest with lines, using only Pythagoras. This procedure is illustrated in Example 2; the time required is nearly proportional to the diameter of the circle (1 sec for 500 diameter); the time saving is particularly great on smaller circles. The appearance is nearly perfect, except that the shape approaches a square for sizes below 20. When using logical operators in drawing, it is important that no point is plotted twice; lines 70 and 80 in Example 2 take care of this in modes 1 and 4. In other modes it will be necessary to employ separate loops for horizontal and vertical lines to achieve this. Acorn published (L Thomas, Acorn User, July/Aug 82, p.7) a beautifully simple circle-drawing procedure which can be speeded up by 30% by drawing two semi-circles simultaneously. Even so, this faster version is nearly three times slower than the square method.

The parameters for PROCcircle in Example 2 are:

XC%,YC% - Co-ordinates of circle centre.
R% - Radius
colour% - Logical colour
op% - Plotting mode for GCOL (0-4).
Use 0 for normal plotting.

The procedure begins at line 3000, and is called by a similar test routine on lines 5 to 70.

```
   5 REM Ex2 - Square method
  10 MODE 1:REM MODE 5 for 16k
  20 FOR radius%=500 TO 25 STEP -25
  30 TIME=0
  40 PROCcircle(500,500,radius%
,radius%/25+1,0)
  50 PRINTTAB(35,30-radius%/25);TIME/100
  60 NEXT
  70 END
3000 DEFPROCcircle(XC%,YC%,R%,colour%,op%
3010 LOCAL X%,Y%,x%,y%
3020 GCOLop%,colour%
3030 VDU29,XC%;YC%;
3040 y%=R%/SQR(2):x%=y%
3050 MOVE x%,y%:MOVE x%,-y%:PLOT 85,-x%,y%
3060 MOVE -x%,y%-4:MOVE -x%,-y%
:PLOT 85,x%-4,-y%
3070 FOR X%=x%+4 TO R% STEP 4
3080 Y%=SQR(R%*R%-X%*X%)
3090 MOVE X%,Y%:DRAW X%,-Y%
3100 MOVE-X%,Y%:DRAW-X%,-Y%
3110 MOVE Y%,X%:DRAW-Y%,X%
3120 MOVEY%,-X%:DRAW-Y%,-X%
3130 NEXT
3140 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

RENUMBER QUERY

David Wright asks why RENUMBER 1000 and then BREAK, OLD changes the first line number to 232. This is because BREAK resets the first two bytes of the program to zero, and the second of these is the high byte of the first line number. OLD cannot restore this so the first line number is reduced modulo 256.

SOFTWARE REVIEW

Program: FUN GAMES
Price: £10.00 inc VAT.
Reviewer: D.E.Graham

Supplier: BBC

As with all the BBC's programs for the Beeb, a great deal of effort has gone into the packaging, and this is really impressive. It will be a source of disappointment to many that the programs within do not always reach the same high standards.

The cassette contains four games programs - and this is all to the good. The games are generally amusing diversions, but would not be described as gripping or addictive, and they are generally unambitious graphically; though Breakout is an exception.

Breakout: This uses mode 7 graphics, but is quite well executed - and fast - quite a bit faster than the breakout published this issue. Colours are used effectively, and higher bricks produce higher pitched blips. There is however, no way to exit the program - except by using Break.

Dodgems: Quite a nice program in which you guide a car around a simple maze trying to avoid the computer's car which tracks you down. The graphics are simple but adequate; though there is again no way to stop the program, except Break: and you are left with a dazzling flashing blue and yellow screen each time you play. The only way to remove it is to play again.

Flash: The object here is to remember an ever growing sequence of colours (selected from four only). These are flashed in sequence on the screen accompanied by a tone; and you must repeat the sequence. The idea is a good one but the execution, although adequate is really very unexciting visually. There are just four large blocks of colour.

Snake: Another mode 7 game. The object is to guide the snake around the screen to pick up a letter placed randomly on the screen. When you have eaten that one, another one appears - ad infinitum. Again a little more imagination would have improved things a lot. The snake is just a long band of white; it has no colour or texture, and there is very little strategy involved in playing the same. This is really a far cry from Computer Concept's Snake or BEEBUG's Magic Eel, which are both highly compulsive, have a series of screens of increasing difficulty, and call for some degree of strategic play. Against this, of course there are four games on the BBC cassette.

One final point - the programs are loaded in with a header program which tells you to turn on the cassette etc, and this is designed to operate in mode 7 - but has no mode 7 command - so that if you are in say mode 4 you get a rather odd screen - and if you are in mode 5, the instructions are almost illegible. One further remark on the header is that when loading Breakout you are given no screen loading messages of any kind, and you have no idea what is happening.

I also took a look at two other BBC cassettes, "Painting", and "Music", and have generally similar criticisms to make about them. The BBC's reputation for excellence in other spheres is not apparently matched, as yet by the quality of its software: though these are early days for them.

Program: MONSTERS
Price: £9.95 inc VAT
Reviewer: D.E.Graham

Supplier: Acornsoft
Requirements: Model-B

This is another extremely well written game from Acornsoft which compares for sound and graphics with their Planetoid (formerly Defender) and Snapper. The colour

graphics is arguably even more striking than these two, and a copy of their well-circulated advertisements will show you the scenario. You control (with up, down, left and right) a man plagued by monsters. You must take him up the ladders and dig traps for them (you dig with "D"). When they fall in you fill them in with "F" and they drop through with a resounding thud. Of course you need to develop a strategy to prevent the other marauding monsters from getting you while you are trapping the first. On the third page you get a more grisly green monster accompanying the red ones, and to trap these, you must dig holes on two levels, so that he has a double fall. You must also finish each screen in a limited time or you will run out of oxygen. The game is certainly very well conceived and very addictive. My only adverse comments are on the extreme difficulty of catching the green monsters, and on the use of the "D" and "F" keys - to operate these you really need to take your hand off the up-down keys, and this makes play difficult if you are suddenly attacked when quietly filling in a monster. The price of £9.95 is high. Acornsoft would argue that you have to pay for quality, but on the volume of sales one assumes that Acornsoft has, they could certainly be a little cheaper.

WORDWISE WORD PROCESSOR REVIEW

Supplier: Computer Concepts Tel 09277 69727.
Price £39.00 + £1.50 +VAT.
Reviewer: David Graham

Note - Wordwise requires a Series 1 operating system.

Wordwise is a full word processor package, and is available ex-stock. We have not had our copy very long, so this review is of necessity somewhat limited in scope - though we are at least using it to produce this review.

Wordwise is supplied in EPROM, and comes with simple fitting instructions. It was only a matter of minutes before we had the chip installed in one of the spare 'paged ROM' sockets in the Beeb; and on switching on, the system was instantly up and running. To enter Wordwise from Basic you type *WORDWISE <return> (or the abbreviation *W.). It then announces its presence, and asks if it should clear the workspace. When you answer, it presents a menu of options. These are concerned with entering and editing text, formatting it, viewing it in its formatted state, printing it out, and saving and loading it to disc or cassette.

Wordwise operates in Teletext mode, so as to keep as much memory as possible for text storage, and makes full use of cursor keys and the red function keys. Entering text is simple - you just type it in continuously, without any need to press return at the end of a line. In fact it is not necessary to know where the line breaks are at this stage, so you can keep your eyes on the text you are typing from (or the keyboard).

For editing purposes you can step very quickly through the text using a combination of the cursor control keys and CTRL and SHIFT. You may enter new text wherever the cursor is positioned, and the existing text at the cursor may either be overwritten, or automaticallly moved down.

Wordwise provides a number of the most useful commands found on comparable word processors - such as block move of text, right justification, change from lower to upper case and back, string search and replace etc. This latter command permits all occurrences of a given word to be replaced with any other - eg replace "ROM" with "read only memory". Incidentally this last feature may be used for the editing of Basic programs. These may be read in as a text file, and the names of variables or procedures changed, or keywords searched for.

Once the editing is complete, the text may be viewed in its fully formatted state in mode 3 (if there is sufficient memory left), and can then be output to tape, disc or printer.

From the limited tests that we have been able to perform, Wordwise appears to be a most welcome addition to the software available for the BBC machine. It will certainly make the production of BEEBUG a lot easier.

STOP PRESS

Subsequent to the writing of this review we have begun to negotiate a special discount on the Wordwise package for BEEBUG members. If all goes well, you will find details of this on the inside back cover of this issue.

SOUND AND ENVELOPE DESIGN (PART 2)

by David Graham

Last month I introduced Ian Soutar's Envelope Editor program, and began to look at ways of using it to edit the pitch envelope part of the Beeb's ENVELOPE command; and I hope that you noticed the 'stop press' item on page 2 explaining how to get the full program into 16k. This month I shall look at ways of generating and editing the amplitude envelope part of the ENVELOPE command. At the end of the article there is a brief note of major variables and procedures used in the program, the full listing of which was given last month.

Notes on T

Before looking at the amplitude envelope, there is one further point to make about the variable T mentioned last month. This gives the time base for both envelopes in one hundredths of a second. Most examples used last month had this set at 1 - the fastest rate possible. If you set it to 2 and play a sound (by pressing L) you will hear the actual increments in sound, because the change is already slow enough to be distinguished by the ear. In fact I cannot see for the moment why one should ever require T to take values much greater than 1.

If you add 128 to the value of T (technically speaking, setting bit 7 of T high), this does not change the time at all, but introduces another effect. With T in the range 128 - 255, the pitch envelope does not repeat. You can soon see the effect of this by running the program with all the data as supplied, but with T changed to 129. You just get one pitch cycle followed by a constant tone for the rest of the amplitude envelope.

Amplitude Envelope

Since we are investigating the amplitude envelope, we will first set the pitch envelope to give sounds of a constant pitch. The following will achieve this:

T	PI1	PI2	PI3	PN1	PN2	PN2	
1	Ø	Ø	Ø	6	3	3	Pitch = 1ØØ

As a first working example of an amplitude envelope, it is worth re-creating the example given in the User Guide page 247. To achieve this, enter the following values:

AA	AD	AS	AR	ALA	ALD	
3Ø	-4	Ø	-5	12Ø	8Ø	Duration = 7

You should get a curve very similar to the illustration on page 247; showing how closely the Envelope Editor program mirrors the envelope shape as conceived at Acorn. This particular envelope is of interest for two reasons. It is the general shape produced by some natural percussively produced sounds. And secondly, it incorporates the concepts which the ENVELOPE parameters were conceived to reflect (and which make it quite hard for a novice to use - though the Editor program goes some way to overcoming this). To see how the design of the parameters reflect this particular application, we have only to see how the envelope is defined.

The first thing to note is that while the pitch envelope can take a maximum of 3 phases, the amplitude envelope has four. They are termed 'Attack', 'Decay', 'Sustain' and 'Release' from left to right. The initial attack phase takes the amplitude fairly rapidly up from zero to the value 126 (at a rate of rise given by AA). It then drops at a rate given by AD(= -4) until it reaches the sustain level given by ALD (= 8Ø). Audio output then keeps the same value during the sustain

phase (because AS, the change rate for this phase is zero). The sustain phase is maintained until Duration (the fourth parameter of the SOUND command) as counted from the start of sound output, is exceeded. The amplitude is 'released', and decays at a rate given by AR (= -5).

As you can see from this, the amplitude envelope is quite a bit more complex to use than its pitch counterpart. Because of this it is difficult to develop a rule of thumb to employ when creating amplitude envelopes; though for many purposes it might be useful to start off from this basic shape, and edit it to suit your requirements. You can streamline things by incorporating this data in the program itself. To do this alter the data in line 70 to read:

```
70 DATA 1,0,0,0,6,3,3,30,-4,0,-5,120,80
```

and alter line 30 to read:

```
30 p%=150:d%=10
```

If you press 'L' you will hear that the amplitude of the tone does follow the graph. You can then begin to edit the amplitude for various effects. Suppose firstly that you wish to create the sound of a small bell, or clock chime. Before generating a suitable amplitude, we must get the tonal quality or timbre more or less right. A simple way to create the right sort of effect is to use two SOUND commands, whose pitches differ by a small amount, but which are both controlled by the same ENVELOPE command. Therefore insert into the program the line 135 SOUND 2,1,p%+1,d%. This puts sound channel 2 under the control of envelope 1 (the editor uses envelope 1), and gives it a pitch one unit greater than the first sound channel. If you now run the editor with the data above to give a plain pitch envelope at pitch 150, and an amplitude envelope as described above, then you should hear a bell-like sound. You will note that there are two principal defects. It lasts too long before fading away - so change Duration to 4 or 5; and it does not start abruptly enough. The so-called attack rate (AA) should therefore be increased to the full 127.

If you want to make an electronic organ sound on the other hand then you need a much slower attack rate. Keep the two SOUND statements as they are, but change Pitch to 100, and Duration to 10. An attack rate of AA = 10 to 20 gives the desired effect. Reducing it further gives something more reminiscent of a violin - but does not sound much like one because we do not have the right timbre. To simulate a variety of timbres, you need to be able to mix the notes from the three sound generators at different amplitudes - you therefore need to use several envelopes simultaneously, and this is beyond our present scope.

If you remove the sustain phase altogether, you get a much more dull thud. Try pitch = 3. Duration = 10

AA	AD	AS	AR	ALA	ALD
127	-5	-5	-5	120	60

White Noise

Bangs, crashes and other effects can be produced quite easily by using the channel zero option in the SOUND command. To try this, delete line 135 if you have entered it, and alter 130 to:

```
130 SOUND 0,1,p%,d%
```

If you run the program now, you will find that the value of 'pitch' (0 to 7) defines the 8 available sound effects (see User Guide pages 348 and 349, or BEEBUG 1 page 11). Entering pitch =0 gives a 'division by zero' error in the graph plot routines, but you get the same sound with pitch = 8 (without the attendant error).

The three varieties of white noise are given by setting 'pitch'=4,5 or 6. If you have set the amplitude envelope as in the first examples (with an attack rate of 50) you will get a rather odd sound. To make it sound like a bang, you need a fast

attack - so try AA = 127. This gives the right attack rates but the sustain obviously needs removing - so make AD, AS and AR each equal to -5, change ALD to 60, and change T to say 3 or 5. This now gives a reasonable explosion, though you do need a better speaker to reproduce it properly. Changing pitch values between 4 and 6 will change the deepness of the note. To summarise, here is a set of values suitable for explosions:

Change line 130 to: SOUND 0,1,p%,d%

Data:	T	PI1	PI2	PI3	PN1	PN2	PN3	
	5	0	0	0	6	3	3	Pitch=4,5,or 6
	AA	AD	AS	AR	ALA	ALD		
	127	-5	-5	-5	120	60		Duration=10

though you should experiment around these parameters.

Once again there are vast rewards for the experimenter, and these articles are intended to help to familiarise people both with the Editor program, and the commands themselves. There is of course so much more that can be done with the reasonably powerful commands available on the Beeb. As a first area of experiment, you might like to try introducing small variations into the pitch envelope when the sound is on channel zero.

Notes on the Envelope Editor Program

Procedures

PROC A	Draws amplitude envelope
PROC C	Handles pitch and amplitude parameters on screen
PROC P	Draws pitch envelope
PROC PL	Plays the sound

Major variables

E%( )	Holds current envelope parameters
l%( )	Holds envelope parameter names and data ranges
p%	Value of pitch (set in Line 30)
d%	Value of Duration (set in Line 30)

The main program loop is at Lines 80 and 90. You may alter the preset ENVELOPE values by altering the 13 pieces of data in line 70. These are the values of T onwards - ie T, PI1, PI2....etc. (See User Guide for ENVELOPE syntax). If you want to find out the envelope values that you have edited into the program at any stage, they are simply as follows, and may be read directly from the screen - all except the first parameter N - which is set to 1.

ENVELOPE 1, T, PI1, PI2, PI3, PN1, PN2, PN3, AA, AD, AS, AR, ALA, ALD

Remember that you can always get back to the preset data values by pressing Escape, and re-running the program. Note that if you are typing in this program take care not to confuse the lower case L (appearing as l% in line 20 and elsewhere) with a figure one. There are no appearances of the expression l% in this program - it will always be 1).

We will give a £50 prize to the most interesting repertoire of sound effects. These should be submitted in program form on cassette or disc. There should be no accompanying literature. The program must therefore be self contained, with your name and address in REM statements at the beginning. (It is easier and cheaper, if sending a cassette, just to send the cassette and not its library case). We regret that cassettes/discs cannot be returned in this instance. The closing date for the competition is 15th February 1983. Please mark you envelope "SOUND COMPETITION", and post to BEEBUG, PO Box 50, St Albans, Herts. AL1 2AR.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

LOADING TIMES

Charles Lyne says that the approximate loading time from cassette can be given in seconds by PRINT &****/&4E, where **** is the program length printed on the screen after a sample load.

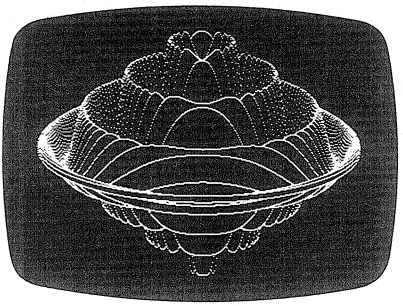
BOOK REVIEWS

Title: Practical Programs for the BBC Computer and Acorn Atom
By: David Johnson-Davis Price £5.95
Reviewer: David Graham

As its title suggests, this book is not concerned with teaching you to program, or with teaching you to use the computer, but with the presentation of a number of programs. The range of programs offered is good, and includes games, graphics, number and word handling, and rather surprisingly an SPL compiler. With each program there is discussion of the objects and principles involved (though nothing on the programming principles). The program listings are also broken down into useful subsections with boxed functional headings. This makes them relatively easy to follow.

The final section of the book, chapter 5, introduces the subject of compilers, and gives a listing for an experimental Simple Programming Language compiler for the BBC machine, plus a number of SPL programs that may be run on it. This all looks very interesting, and well worth some closer study. But if you are not interested in compilers, you are left with only 65 pages of the book; and since some of that space is taken up with the Acorn Atom version of each program presented, the book becomes a little expensive per useable page. It does, however, contain some good ideas, such as the brief program given below which plots the very nice 3D object in the photograph.

```
 10 REM FROM :
 20 REM PRACTICAL PROGRAMS FOR THE
 30 REM BBC COMPUTER AND ACORN ATOM
 40 REM BY DAVID JOHNSON-DAVIS
 50 MODE4:VDU29,640;512;:XS=4:YS=4
 60 A=640:B=A*A:C=512
 70 FORX=0TOA STEPXS:S=X*X:P=SQR(B-S)
 80   FORI=-P TO P STEP 6*YS
 90     R=SQR(S+I*I)/A
100     Q=(R-1)*SIN(24*R)
110     Y=I/3+Q*C
120     IFI=-P THEN M=Y:GOTO150
130     IFY>M M=Y:GOTO160
140     IFY>=N GOTO170
150     N=Y
160     PLOT69,-X,Y:PLOT69,X,Y
170     NEXT:NEXT
180 END
```



Title: Let Your BBC Micro Teach You to Program
By: Tim Hartnell Price £6.25
Reviewers: David Graham and Sheridan Williams

This book has a quite different aim from David Johnson-Davis' book. It is intended primarily for people with very little experience of the BBC machine, and introduces most of the Basic keywords from first principles. It appears to do this reasonably competently, and also to contain a good number of programs that readers can experiment with, and which illustrate the principles under discussion. The book is quite nicely produced, and would appear at first sight to represent reasonable value. Moreover there is a great need for such a book. In spite of all this, we

feel that we cannot recommend it; and this is a pity. Our strong reservations arise because the book in our view generally provides a very bad example of programming practice. For example the implementation of procedures on the BBC Micro really makes subroutines, with their attendant lack of structural clarity, a thing of the past. Yet very many of the programs in this book use subroutines in the place of procedures. Immediately after the section on procedures there are a few programs which do make good use of procedures, but this is not the case generally. Furthermore there are a number of occasions where procedures are actually exited with a GOTO! This is not good practice, and can in certain situations cause the machine to issue error messages. Apart from the generally unclear structure of programs, little use is made of the long variable names on the BBC machine - even on shorter programs. The likely reason for these weaknesses is that many of the programs may have beem adapted from those designed for less complex machines - and with one program, "Personal Accounts", it is specifically stated that it has been adapted from a program for the ZX8Ø.

In short, although the approach is a good one, and individual features of Basic clearly introduced, a book which claims to be teaching you to program the BBC Micro should in our view make far greater use of the advanced features of that machine in the programs given as examples.

STOP PRESS NEW ASSEMBLY LANGUAGE BOOK

We have just received a review copy of ASSEMBLY LANGUAGE PROGRAMMING FOR THE BBC MICRO by Ian Birnbaum (Macmillan £8.95). This 300 page book appears to be just what potential assembly language programmers are looking for. It seems clear, thorough, methodical, and is BBC specific. It also contains some useful machine code programs, including a monitor and string sort. From a brief examination, it appears to be far more useful than even classics such as Zaks' Programming the 6502. D.E.G.

ANALOGUE UPGRADE

by Rob Pickering

Continuing with our series on upgrading a model A micro: this month we describe how to fit the Analogue To Digital Converter. This will allow joysticks to be used with the computer, and enable the computer to read in, or measure, small analogue voltages.

Rather than giving a description of what an Analogue to Digitial (A-D) converter is and what it can do, this article will be concerned purely with the fitting and testing of the device.

If you have already added the RS423 upgrade described last month, then it won't take you long to fit the A-D converter. Removing the case and the motherboard then replacing them afterwards constitutes most of the work. Since this was described fully last month, we will refer you to the relevant sections of BEEBUG no 7 in order to avoid repetition.

WARNING

As stated last month, carrying out any user-upgrade is likely to terminate your guarantee. If you have already made any alterations to your computer then this won't worry you. Also, the upgrade requires very delicate soldering on the main Printed Circuit Board (PCB) which is a serious matter at any time and should only be undertaken by those well skilled in delicate soldering.

IDENTIFYING THE COMPONENTS

There are only three components in the upgrade kit. One socket and two Integrated Circuits. Given below are the labels by which the components are identified on the circuit board, followed by a description.

SK6 ..... 15-pin D-type socket. Right-angle PCB mounting.
I.C.73 ... 28-pin D7ØØ2C
I.C.77 ... 14-pin 74LSØØN

In identifying the components you may find that the code on top of your I.C.s is not quite the same as above. However, the codes should contain "D7ØØ2" and "74LSØØ" to stand any chance of being the correct components. If this is not the case then I suggest that you get in touch with your supplier and ask for advice.

TOOLS: You will need a soldering iron of less than 25 Watts and with a fine tip; a cross-head screwdriver size No.4; a pair of long-nosed pliers are a good idea though not essential. An I.C. insertion tool will save a lot of trouble, though not essential and not worth buying just for use on this occasion.

TAKING IT APART

First you need to open up the computer, and remove the motherboard. Follow the instructions 1-6 and 8-1Ø in BEEBUG no 7 pages 25 and 26.

FITTING THE SOCKETS AND I.C.s

(1) Insert SK6 (The 15-pin 'D'-type socket) into the set of holes marked "SK6" and situated to the left of the DIN plugs, and hold it in position.
(2) Solder the 15 pins of SK6 very carefully (from the underside). Although it may not be very obvious, there are fine copper tracks running between some of the pins. Unless you are very careful with the soldering iron, you can bridge these wires with solder and short them out.
(3) You can now insert the two I.C.s. Turn the motherboard back so that it is component-side-up. Look in the back-left region of the board, about 12cm in from the left and the back, and locate a large empty socket marked I.C.73 and a much smaller socket marked I.C.77. Insert both I.C.s into the corresponding sockets, the same way round as the rest of the I.C.s on the board. That is with

the small 'U'-shaped dimple towards the back of the board. Be careful as usual not to bend the pins out of shape, or to use too much force, and remember to keep static electricity away from them.

(4) Now that all the components are in place you can reassemble everything. You should already have turned the motherboard back so that it is the correct way up, ready to be put back in place.

RE-ASSEMBLY

To re-assemble the computer, follow the instructions 13 to 20 given in BEEBUG no 7 page 26.

TESTING

The easiest way to test the A-D converter after fitting is to simply plug in some joysticks and try it out. My thanks to John Yale for the following program designed to test a pair of joysticks of type ANH01 as supplied by Vector and Acorn dealers. It gives visual representation of the digital values produced by the joysticks, but also gives a sort of graphical display as well. This program should help to give you a grasp of programming to control the joysticks. Line 60 uses the ADVAL (see User Guide) command to read in two voltage levels from each joystick. One represents horizontal and the other vertical positions. Traces are produced on the screen for each joystick, and measured values are also printed out. Lines 160 and 180 use the ADVAL function on channel zero to test whether the fire buttons are being pressed.

```
  5 REM JOYSTICK TEST PROGRAM
 10 ON ERROR GOTO 200
 20 MODE 4
 30 VDU23;8202;0;0;0;
 40 REPEAT
 50  FOR I=1 TO 4.
 60   V=ADVAL(I) DIV 64
 70   Y=800-128*I
 80   MOVE0,Y
 90   GCOL 0,1
100   DRAW V,Y
110   GCOL 0,0
120   DRAW 1280,Y
130   PRINTTAB(0,8+4*I);"Channel ";I;V;SPC(8)
140  NEXT I
150  PRINTTAB(10,30);
160  IF ADVAL(0) AND 1 THEN PRINT"FIRE"
 ELSE PRINTSPC(4)
170  PRINTTAB(25,30);
180  IF ADVAL(0) AND 2 THEN PRINT"FIRE"
 ELSE PRINTSPC(4)
190 UNTIL FALSE
200 MODE 7
210 REPORT:PRINT" AT LINE ";ERL
```

SUPPLY

The analogue upgrade kit is available from a number of suppliers - see advertisers and discounts page in this issue. I would like to take this opportunity to thank Customised Electronics Limited (known as CEL) for their much appreciated help in supplying the kit for the A-D converter, as well as last month's RS423 kit.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

LOUD AUDIO

The audio output from the BBC micro is generally rather quiet, and many of the subtleties of the excellent sound facilities on the Beeb can be missed because of this. There are two ways to make it louder:

1. Turn it up. There is a preset volume control fairly close to the loudspeaker connector under the keyboard. This may be turned up a little.

2. Use an external speaker. The audio amplifier in the Beeb is an LM386. This is easily capable of half a watt of audio with a decent sized 8 ohm speaker. You could connect a socket so as to break the audio lead to the internal speaker, and the socket could be mounted in the RESET switch hole at the rear of the machine. Very loud audio can be obtained in this way.

VIDEO CONTROLLER CHIP NOTES – ACCESSING THE 6845

by John Yale

The new User Manual explains how to use the VDU 23 command to alter the registers in the 6845 CRT Controller chip (page 385), but gives no information on how many registers there are or what they do. This article provides brief information for those users interested in finding out more about this interesting I.C. We should stress that this article is for the experimenter and that it is not normally necessary to address the 6845 directly, since the operating system provides all the control required for most applications via VDU calls.

The Motorola 6845 data sheet reveals that there are 18 registers, 14 of them write only and two read only. The registers are named RØ to R17 and may be written to on the BBC machine with the VDU command:

```
VDU 23,Ø,R,X,Ø;Ø;Ø;
```

where R is a number in the range Ø to 17 corresponding to the register, and X is the value of the byte of data to be written.

Some of the registers may be read, but no BASIC command is provided for this, so the memory mapped IO must be addressed directly or via a SHEILA call. The 6845 resides at &FEØØ and &FEØ1, and to read register R into the Variable X the following code may be used:

```
?&FEØØ = R
X = ?&FEØ1
```

The first line writes the register number into the 6845 address register whilst the second reads the specified register from the register file. Only registers R14 to R17 may be read.

The following Program will allow any of the registers to be written and the readable register pairs are displayed. R12,13 is displayed as it does seem to be readable even though the data sheet indicates that it is not.

```
10 REM CRTC READ/WRITE
20 REM BY JOHN YALE
30 ON ERROR OFF
40 MODE 7
50 INPUT'"What mode ",mode
60 MODE mode
70 PROCREAD
80 ON ERROR GOTO 30
90 INPUT'"What register ",R
100 IFR<Ø THEN150
110 ON ERROR GOTO 80
120 REPEAT
130 INPUT'"Value ",X
140 VDU 23,Ø,R,X,Ø;Ø;Ø;
150 PROCREAD
160 UNTIL FALSE
170
180 DEF FNread(R)
190 LOCAL X
200 ?&FEØØ=R
210 X=?&FEØ1
220 ?&FEØØ=R+1
230 =256*X+?&FEØ1
240 DEF PROCREAD
250 FOR R1=12 TO 16 STEP 2
260 PRINT"  R";R1;"="~FNread(R1);
270 NEXT R1
280 ENDPROC
```

After RUNning the program, enter the mode you wish to investigate. The contents of the three readable register pairs R12,13 R14,15 and R16,17 will be printed out. You will then be prompted for a register number, and the value to be written to it. At any stage pressing Escape will return you to the previous prompt in the program, finally exiting at the "Mode ?" prompt. Note that there is no error checking on inputs, and that pressing return on empty data will read in a zero; and that entering data zero into register zero turns off the VDU drivers, and thus blanks the screen. (Press 'Break' to recover).

Now that you are all set to investigate, what do the registers actually do? The following descriptions taken from the data sheet should help to start you off. You may find registers 12,13 and 2 to be the most directly useable.

Horizontal Total Register (R0)

This 8-bit write only register determines the horizontal sync frequency. It is programmed with the total number of displayed characters plus the non-displayed characters minus one. This register is normally set to 63 for 20 or 40 column modes, and 127 for 80 columns.

Horizontal Displayed Register (R1)

This 8-bit write only register determines the number of displayed characters per line. The number programmed must be less than the contents of R0. The character size is not changed by this command, the lines get shorter to accommodate fewer characters, and the start of lines will slant across the screen, unless the VDU driver is changed.

Horizontal Sync Position Register (R2)

This 8-bit write only register sets the position of the horizontal sync pulse in character times. When the programmed value is increased the display is shifted to the left. When the programmed value is decreased the display is shifted to the right. Changes of plus or minus up to 5 can be used for screen shift effects. The sum of the contents of R1, R2 and R3 should be less than R0. Typical values for R2 are 51 for Mode 7, 98 for Mode 0, and 49 for Mode 5.

Horizontal Sync Width Register (R3)

This 4-bit write only register controls the width of the horizontal sync pulse from zero to 15 character times to compensate for different monitor's requirements.

Vertical Total Register (R4) and
Vertical Total Adjust Register (R5)

The frequency of the vertical sync is set by both R4 and R5. The calculated number of character line times to get exactly a 50Hz vertical refresh rate is not an integer. The integer number of character line times minus one is programmed into the 7-bit write only vertical total register (R4). The fraction of character line times is programmed in the 5-bit write only vertical total adjust register (R5) as a number of scan line times.

Vertical Displayed Register (R6)

This 7-bit write only register specifies the number of displayed character rows on the screen, and is programmed in character row times. Any number smaller than the contents of R4 may be used, and may be used to blank off an area of the screen.

Vertical Sync Position (R7)

This 7-bit write only register controls the position of the vertical sync pulse in character line times. When the programmed value is increased, the display position of the screen is shifted up. Any number equal to or less than the vertical total (R4) may be used. Any offset which has previously been specified in a *TV command will be used to adjust the value sent to R7.

Interlace Mode Register (R8)

This 2-bit write only register has the following effect.

Normal sync mode (0 or 2): Only one field is available, and each scan line is refreshed at the vertical sync frequency.

Interlace sync mode (1): The frame time is divided between even and odd alternating fields. The horizontal and vertical timing relationship (VS delayed by half a scan line) results in the displacement of scan lines in the odd field with respect to the even field. In the Interlace sync mode the same information is displayed in both fields.

Interlace sync and Video mode (3): Alternating lines are displayed in the even and odd fields.

Care must be taken when using either interlace mode to avoid an apparent flicker effect. This is due to the doubling of the refresh time for all scan lines since each field is displayed alternately. This will be more apparent on some monitors than others due to the different persistence phosphors used. In addition there are restrictions on the programming of the CRTC registers for interlace operation:

a. The horizontal total register (R0) must be odd (i.e. an even number of character times)
b. For interlace sync and video only, the maximum scan line address R9 must be odd (i.e an even number of scan lines)
c. For interlace sync and video only, the vertical displayed register R6 must be even. The programmed number must be half the number required.
d. For interlace sync and video only, the cursor start register R10 and cursor end register R11 must both be even or both odd depending on which field the cursor is to be displayed in.

Maximum Scan Line Address Register (R9)

This 5-bit write only register determines the number of scan lines including spaces that make up a character.

Cursor Start Register (R10) and
Cursor End Register (R11)

These registers allow a cursor of up to 32 scan lines to be placed on any scan line of the character block. R10 is a 7-bit write only register used to define the start scan line and the cursor blink rate. The least significant 5-bits determine the cursor start scan line counting from the top of the character. Bits 5 and 6 of R10 control the cursor operation as shown in the following table:

Bit-6	Bit-5	Cursor Mode
0	0	Non-blink
0	1	Non-display
1	0	Blink,1/16 field rate
1	1	Blink,1/32 field rate

R11 is a 5-bit write only register which defines the last scan line of the cursor.

Note that the cursor off command given on page 77 of the manual: VDU 23;8202;0;0;0; is equivalent to: VDU 23,0,10,32,0,0,0,0,0,0 and is thus sending the value 32 to R10, ie the cursor non-display command.

Start Address Register (R12-H,R13-L)

This 14-bit register pair controls the first address output by the CRTC after vertical blanking. It consists of an 8-bit low order register (R13) and a 6-bit high order register (R12). The start address register determines which portion of the RAM is displayed on the screen. Hardware scrolling is accomplished by

modifying the contents of this register. By changing the contents by the line length - eg 40 or 80, a line feed, up or down, is effected instantly, with no effort from the CPU. A larger change could be used for a complete page change. Since it is so fast, it should be possible to change pages for an animated display. Note though that the two registers provide only a 14-bit address. The additional 2 high bits are generated from the CRTC row address generator.

Cursor Address Register (R14-H,R15-L)

This 14-bit read/write register pair is programmed to position the cursor anywhere in the display using the same addressing as for R12,13.

Light Pen Register (R16-H,R17-L)

This 14-bit read only register pair captures the refresh address output by the CRTC on the positive edge of a pulse input to the LPSTB pin (available on the Analogue input connector). Since the light pen pulse is asynchronous with respect to refresh address timing, an internal synchroniser is designed into the CRTC. Due to delays in this circuit the value of R16 and R17 will need to be corrected in software.

Acknowledgements: We are most grateful to Bazyle Butcher and 'ACCumulator' (The magazine of the 'Amateur Computer Club') for the inspiration for these notes.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

Noisy Audio

According to a letter from Acorn (Ref AAW/MS) to one of our members:-

"The loudspeaker noise is caused by the fact that the computer's sound is generated in a digital fashion. An adjustment to this is available from a dealer.

The recommended modification is a ..... 10k resistor between pin 8 IC20 and ground (hole just below pin 8)."

[We think that the real cause is an unterminated audio input line. Ed]

Program tested on 0·1 and 1·1 O.S.

BREAKOUT (16k/32k)

by Adrian Calcraft

The object of this game is to destroy a brick wall by bouncing a ball at it. The wall disintegrates a little every time the ball hits it, but it takes concentration and skill to remove the whole wall. A bat is controlled by the player, who must position it (left/right) to ensure the ball is kept in play. Only three serves are given per game, plus a bonus of one serve for each complete wall demolished.

Striking the ball whilst the bat is in motion causes the ball to spin from its normal path, allowing more control for the skilled. A random spin also occurs when the ball hits the upper boundary once having broken through the wall.

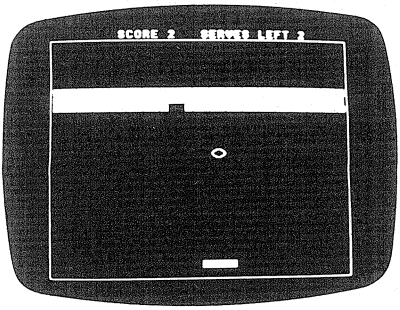
The game uses Mode 4, which leaves precious little space for the program on 16k machines, consequently the variables and procedure names have all been cut to a bare minimum. When typing the program into a 16k machine, remove all REMs and ensure that no extra spaces are inserted, or you may run out of room. But do not remove spaces in IF statements, because the "THEN" has been left out to save space. Also the instructions in the program will only fit into 32k machines, so those with 16k please ignore lines:- 140 and 1230 to 1330.
Bat controlsare "Z" for left, and "X" for right, and the control of the program is by lines 180 to 270.

For those interested here is a list of variables and procedures

A% Used in FNCR to check when ball hits the bat or a brick.
B% Count of current score, on this wall only.
C% Used with A%.
D% The number of brick segments to be hit by the ball (1 or 2).
F% Final spin. The total spin given to the ball by the bat.
H% Horizontal direction indicator (1 or -1).
I% The highest score on the machine.
J% The accumulated spin, transfered to F% when bat hits the ball.
L% Level of play. Used to determine how many walls to build.
M% The score needed at this level, on this wall only, to destroy the current wall totally. See B%.
N% Used to determine the wall colour. Red, blue and green used.
P% X co-ordinate of ball's position.
Q% Y co-ordinate of ball's position.
S% Number of serves remaining.
T% Total score, not including the score on this wall (B%).
V% Vertical direction indicator (1 or -1).
X% X co-ordinate of bat's position.
Y% Y co-ordinate of bat's position.
Z% Used as indicator for ball out of play.
X1% X co-ordinate of bat's last position.
CR% Correction value, used to enable double bounce when entering wall diagonally.
B$ String variable representing ball.
BT$ String variable representing bat.
W1$ String variable representing 1st /3rd /5th rows of bricks.
W2$ String variable representing 2nd /4th rows of bricks.

ProcST Start proc. Instructions, only for 32k machines.
ProcNW New wall proc. This proc is used to set up a new wall at the beginning of a game, or after the wall is destroyed.
ProcBT Bat movement proc. This proc is used repeatedly to check for movement of the bat by player, to increment the spin count and redraw the bat.
ProcBL Ball movement proc. Repeatedly used to move ball, check for a bounce, change direction indicators, update counters and check for ball out of play.
ProcW Wall count proc.Used to redisplay score and check for a destroyed wall.
ProcNBL New Ball proc. After ball out of play, this re-serves ball.
ProcBX Draw Box proc. Sets up screen at beginning of a game.
ProcNG New Game proc. Displays your score and high score.
FNCR Check for crash. This function enables the use of the POINT keyword, by calculating the ball's position relative to the graphics origin.

```
   5REM BREAKOUT BY ADRIAN CALCRAFT
  10MODE4
  20VDU19,1,1,0,0,0,19,0,4,0,0,0
  30VDU 23,226,7,28,48,96,96,48,28,7
  40VDU 23,227,224,56,12,6,6,12,56,224
  50B$=CHR$226+CHR$227
  60VDU 23,224,0,254,254,254,254,254,2
54,254
  70VDU 23,225,0,255,255,255,255,255,2
55,255
  80VDU 23,0,11,0;0;0;
  90*FX 11,7
 100W1$=CHR$224+CHR$225:W2$=CHR$225+CH
R$224
 110W1$=STRING$(18,W1$):W2$=STRING$(18
,W2$)
 120BT$=STRING$(4,CHR$225)
 130I%=0:CR%=0
 140PROCST
 150F%=0:B%=0:Z%=TRUE:S%=1:L%=2:T%=0:M
%=72:J%=0:N%=1
 160ONERROR GOTO310
 170REM
 180PROCNW
 190REPEAT
 200REPEAT
 210PROCBT
 220PROCBL
 230PROCW
 240UNTIL Z%=FALSE
 250PROCNBL
 260UNTIL S%<0
 270ENVELOPE1,1,1,10,10,1,10,5,127,0,0
,-5,26,126
 280SOUND2,1,20,100
 290FOR A=1TO10000 :NEXT A
 300VDU 19,1,1,0,0,0
 310PROCNG:GOTO150
 320REM
 330DEFPROCBX
 340GCOL0,1
 350MOVE55,20:DRAW1225,20:DRAW1225,923
:DRAW55,923:DRAW55,20
 360PRINT TAB(2,9);W1$
 370PRINT TAB(2,10);W2$
 380PRINT TAB(2,11);W1$
 390IF L%>3 PRINT TAB(2,12);W2$
 400IF L%>4 PRINT TAB(2,13);W1$
 410X%=2:Y%=29
 420PRINT TAB(10,2)"SCORE"
 430PRINT TAB(20,2)"SERVES LEFT"
 440PRINT TAB(X%,Y%);BT$
 450P%=RND(33):P%=P%+3:V%=-1
 460Q%=RND(11):Q%=Q%+15:H%=1
 470ENDPROC
 480REM
 490DEFPROCBT
 500X1%=X%
 510IF INKEY(-67) X%=X%+2:J%=J%+2:PRIN
T TAB(X1%,Y%)"  ":GOTO540
 520IF INKEY(-98) X%=X%-2:J%=J%-2:PRIN
T TAB(X1%+2,Y%),"  ":GOTO540
 530J%=0
 540IFX%<2 X%=2
 550IFX%>34 X%=34
 560PRINT TAB(X%,Y%);BT$
 570*FX 15,0
 580ENDPROC
 590REM
 600DEFPROCBL
 610IF F%>4 F%=4
 620IF F%<-4 F%=-4
 630IF Q%<29 PRINT TAB(P%,Q%)"  "
 640P%=P%+H%+F%:Q%=Q%+V%
 650IF P%>36 P%=36:H%=-1:VDU 7:F%=-F%
 660IF P%<2 P%=2:H%=1:VDU 7:F%=-F%
 670IF Q%>30 Z%=FALSE:GOTO760
 680IF Q%<4 Q%=4:V%=1:PRINT CHR$(7):IF
 RND(3)=3 F%=F%+(H%*3)
 690D%=0:IF F%<>0 F%=F%-H%
 700IF FNCR(P%,Q%)<>0 D%=D%+1
 710IF FNCR(P%+1,Q%)<>0 D%=D%+1
 720IF D%>0 B%=B%+D%:V%=-V%:IF Q%=29 F
%=J%:B%=B%-D%
 730IF D%>0 VDU 7:CR%=CR%+1 ELSE CR%=0
 740IF CR%>1 H%=-H%:V%=-V%
 750IF Q%<29 PRINT TAB(P%,Q%);B$
 760ENDPROC
 770REM
 780DEFFNCR(P1%,Q1%)
 790A%=1280*((P1%+0.5)/40)
 800C%=1023*((31.5-Q1%)/32)
 810=POINT(A%,C%)
 820REM
 830DEFPROCW
 840PRINT TAB(16,2);B%+T%
 850PRINT TAB(32,2);S%
 860IF B%<>M% GOTO900
 870 N%=N%*2:IF N%=8 N%=1
 880VDU19,1,N%,0,0,0
 890PROCNW
 900ENDPROC
 910REM
 920DEFPROCNW
 930L%=L%+1:S%=S%+1:T%=T%+B%
 940B%=0:M%=M%+36:IF M%>180 M%=180
 950ENVELOPE 1,2,5,5,5,10,10,30,10,10,
10,10,100,150
 960SOUND 1,1,25,58
 970NOW%=TIME :REPEAT UNTIL TIME >NOW%
+400
 980CLS
 990PROCBX
1000ENDPROC
1010REM
1020DEFPROCNBL
1030S%=S%-1:IF S%<0 GOTO1090
1040FOR PIT%=50 TO 0 STEP-1 :SOUND 1,-
10,PIT%,1 :NEXT
1050NOW%=TIME+180 :REPEAT:UNTIL TIME>=
NOW%
1060VDU 7
```




```
1070P%=RND(33):P%=P%+3
1080Q%=RND(11):Q%=Q%+15
1090Z%=TRUE:V%=-1
1100ENDPROC
1110REM
1120DEFPROCNG
1130CLS:ONERROR GOTO1330
1140PRINT TAB(8,8)"YOU SCORED",B%+T%
1150IF B%+T%>I% I%=B%+T%
1160PRINT TAB(8,10)"HIGH SCORE",I%
1170PRINT TAB(16,12)"ANOTHER GAME? Y/N
"
1180REPEAT
1190F%=GET:UNTIL F%=&59 OR F%=&4E
1200IF F%=&4E GOTO1330
1210FOR F%=1TO1000:NEXT
1220ENDPROC
1230REM
1240DEFPROCST
1250PRINT TAB(5,4)" HOW TO PLAY"
1260PRINT TAB(5,6)"THE OBJECT IS TO DE
STROY"
1270PRINT TAB(5,8)"THE WALL,BY BOUNCIN
G THE"
1280PRINT TAB(5,10)"BALL AT IT"
1290PRINT TAB(5,12)"BAT CONTROLS ..LEF
T Z..RIGHT X"
1300PRINT TAB(5,28)"HIT A KEY WHEN REA
DY TO START"
1310F%=GET
1320ENDPROC
1330REM
1340MODE7
1350*FX 11,50
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

CASSETTE VOLTAGES

Alastair Lack says that he has discovered from Acorn that the recommended input voltage to the computer from cassette is 500 mv ptp. The output voltage is 180 mv ptp (his measurement).

LOCAL USER GROUP INDEX UPDATE

BEEBUG is happy to act as an information point for local groups. If you would like to be on our index, just drop a line to us and mark the envelope "Local user groups". Don't forget to quote your membership number.

This listing is an update to the last full listing issued in BEEBUG no 6.

Bangor & District
Dilwyn Jones
Fodol Farm,Hafod Lane
Caernarfon Road
Bangor
Gwynedd LL57 4BU
[Meet Bangor Community Centre Rm 1A at 7.30pm Thursdays, fortnightly from 4/11/82]

Ipswich
Karl Brandenburg
19, Oxford Road
Ipswich IP4 1NL

Liverpool
STEM
117 Grove St
by Myrtle Street
Liverpool.
[Meetings on 3rd Thu of month from 7.30pm]

Medway Towns
Dave Laws
25 The Ridgeway
Chatham Kent
0634 42855

Salisbury
Alastair Lack
Tel: 0722 77303

Staffordshire
A. Wiseman
7 Farm Close
Perrycrofts
Tamworth Staffs
0827 69587

South Wales
Nicholas Goodwin
22, Gendros Drive
Gendros, Swansea

EDUCATIONAL GROUPS/SUB-GROUPS

MUSE
(Microcomputer Users in Education)
The Secretary
Richard Green
22 Tennyson Avenue
Hull HU5 3TW
Membership £10.00pa (£9.00 by standing order). MUSE publishes six journals a year and has a considerable software library which will be making available educational BBC programs. MUSE has regional and Area groups throughout the country.

London
W.E.Hunt
143, Montague Road
Leytonstone
London E11 3EW
(Currently working on mathematics, and science programs)

SPECIALIST INTEREST GROUPS

A postal BBC micro ADVENTURE/FANTASY Club has now been formed. The club will provide a central library of Adventure and Fantasy games written by members for the use of other members. Send an SAE for further details to:
BBC Micro Adventure Club
29 Blackthorn Drive
Larkfield
Kent ME20 6NR

Program tested on 0·1 and 1·1 O.S.

SPACE CITY (16k)

by J. Banks

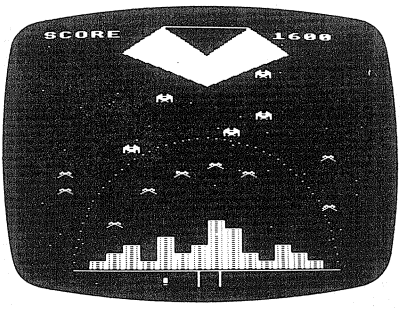
This is an ingenious little program using four colours in mode 5. It will run in a 16k machine even with full instructions. Invading aliens spring from a hovering craft, and descend by random steps on to the unprotected city. You control the customary laser base, but it differs from the usual variety in that you can control the lateral position of the laserbolt in mid-flight. Dextrous use of this facility makes it just possible to prevent a successful attack on the city by the first fleet. The second fleet however soon arrive, and they move even faster. There is on-screen scoring and full instructions are given in the program. If you wish to change the colours then you should experiment with line 110. See the User Guide for details of the VDU 19 call.

Main Procedures

NAME	USE
CITY	Draws the city in a given colour.
INV	Moves the aliens around the screen and tests for an alien landing on the city.
BASE	Enables the gun base to be moved left and right, and also initiates the firing of a missile.
MSL	This moves the missile and checks to see if an alien has been hit.
HIT	If an alien has been hit then this procedure checks to see which one, adjusts the score and resets certain coordinates (eg. the gun base).

VARIABLES USED

X%( ) and Y%( ) Invaders coordinates.
MX% and MY% Missile coordinates.
A% and U% Invader type to be printed.
I% and J% Various loop counters.
H% Height of invaders shot from mothership.
SC% Score.
C% Count of invaders hit or passed city without landing.
FL% Flag for missile fired.
E% End game, invaders landed.
S% Loop counter for stars.
Z Loop counter for city dome.
B% Buildings (X) coordinates.
T% Reads data, height of buildings.
HT% Height of buildings.
R% Invader to be moved.
M% Move invader left right or stay.
P% Test for hit.
G% Keyboard scan for missile/base instructions.




```
 10MODE7
 20PRINTTAB(12,2)"SAVE SPACECITY"''"
STOP THE INVADERS FROM LANDING"''"Z AND
X KEYS MOVE MISSILE LEFT AND RIGHT"'"
 BEFORE AND AND LAUNCHING."''"   PRESS
RETURN KEY TO LAUNCH MISSILE."
 30PRINTTAB(8,16)"THE HIGHER YOU HIT TH
EM"''"          THE MORE YOU SCORE"''"
   PRESS SPACE BAR TO START"
 40REPEATUNTILGET=32
 50MODE5
 60VDU23;8202;0;0;0;
 70DIMX%(19),Y%(19)
 80VDU23,224,255,165,255,165,255,165,25
5,165
 90VDU23,225,0,36,126,90,36,66,129,0
 100VDU23,226,36,102,255,231,255,189,153
,165
 110VDU19,2,6,0,0,0,19,1,3,0,0,0,19,3,5,
0,0,0
 120H%=640:SC%=0:C%=0:MX%=608:MY%=32:FL%
=0:E%=2:CLG
 130FORS%=1TO40:VDU18,0,2,25,69,RND(1279
);RND(1023);:NEXT
 140PRINTTAB(0,1)"SCORE"
 150PROCCITY
 160FORZ=0TO PI STEP PI/50:VDU25,69,640+
(512*COS(Z));60+(512*SIN(Z));:NEXT
```

```
  170VDU25,4,960;896;25,4,800;1010;25,85,
640;768;25,85,480;1010;25,85,320;896;18,0
,0,25,4,480;1006;25,4,800;1006;25,85,640;
890;
  180IFH%<352 H%=352
  190FORI%=10TO19:X%(I%)=640:Y%(I%)=992:N
EXT
  200FORI%=0TO9:X%(I%)=I%*128:Y%(I%)=H%:S
OUND0,-15,4,1:VDU18,3,1,25,4,640;890;25,5
,X%(I%);Y%(I%);225,25,4,640;890;25,5,X%(I
%);Y%(I%);:DELAY=TIME+20:REPEATUNTILTIME>
DELAY:NEXT
  210REPEAT
  220PROCBASE
  230PROCINV
  240PROCBASE
  250VDU4:PRINTTAB(14,1);SC%:VDU5
  260UNTILE%=1ORC%=20
  270IFC%=20 C%=0:H%=H%-64:GOTO180
  280VDU4:PRINTTAB(3,8)"PRESS SPACE BAR"'
"      TO START"
  290PRINTTAB(3,12)"OR F TO FINISH"
  300ON INSTR(" F",GET$) GOTO120,310 ELSE
 300
  310MODE7:PRINTTAB(16,16)"GOODBYE":END
  320DEFPROCCITY
  330VDU25,4,128;60;25,5,1152;60;25,4,600
;60;25,5,600;0;25,4,680;60;25,5,680;0;
  340FORB%=3TO16:READT%:FORHT%=1TOT%:VDU5
,25,4,B%*64;64+HT%*32;224:NEXT:NEXT
  350RESTORE
  360ENDPROC
  370DATA1,2,3,1,4,2,3,6,5,2,1,3,2,1
  380DEFPROCINV
  390R%=RND(20)-1:M%=RND(3)-2
  400IFR%>9 A%=226ELSEA%=225
  410VDU18,2,2,25,4,X%(R%);Y%(R%);A%
  420X%(R%)=X%(R%)+M%*64
  430Y%(R%)=Y%(R%)-32
  440IFX%(R%)>1216 X%(R%)=1216
  450IFX%(R%)<0 X%(R%)=0
  460IFY%(R%)=32 C%=C%+1
  470SOUND1,-10,50,1
  480VDU18,1,1,25,4,X%(R%);Y%(R%);A%
  490IFPOINT(X%(R%)+16,Y%(R%)-36)=2ANDY%(R%)
   <320 E%=1:PROCCITY
  500ENDPROC
  510DEFPROCBASE
  520VDU18,2,2,25,4,MX%;MY%;33
  530*FX15,0
  540IFINKEY(-67) AND MX%<1200 MX%=MX%+32
  550IFINKEY(-98) AND MX%>32 MX%=MX%-32
  560IFINKEY(-74) OR FL%=1 PROCMSL
  570VDU18,1,1,25,4,MX%;MY%;33
  580ENDPROC
  590DEFPROCMSL
  600MY%=MY%+32:FL%=1
  610IFMY%>960 MY%=32:FL%=0
  620P%=POINT(MX%+16,MY%-16)
  630IFP%=10RP%=3 PROCHIT
  640ENDPROC
  650DEFPROCHIT
  660FORJ%=0TO19
  670IFMX%<>X%(J%)ORMY%<>Y%(J%)THEN740
  680IFJ%>9 U%=226ELSEU%=225
  690VDU25,4,X%(J%);Y%(J%);U%
  700SOUND0,-15,2,2
  710SC%=SC%+Y%(J%):C%=C%+1
  720Y%(J%)=-32
  730MY%=32:FL%=0
  740NEXT
  750ENDPROC
```

POINTS ARISING

Mini Text Editor

If you are using this on a disc-based machine, you will need to alter line 31022 of the program published last month to 31022 N% = PAGE + 1 This change only affects the printout without line numbers, and moreover does not alter its correct operation using a cassette based system.

Memory Display Utility

Doug Blyth the (real) author of the Memory Display Utility program given in BEEBUG No. 6 has written in with two ways of using his program to examine a Basic program in memory.

Method 1. Renumber the dump routine with high value line numbers, say starting at 30000. Then make a cassette file using *SPOOL "Dump"commands as described in BEEBUG No. 3 page 16. Load the user program to be examined and merge Dump with it and GOTO 30000.

Method 2. Load Dump. Execute PAGE=&1400. Load user program. Execute PAGE=&E00 (on a cassette-based machine). Run and look at memory from &1400 onwards. This method seems the easiest. In passing, when resetting PAGE on any occasion, if on checking its value with PRINT PAGE it is found to be different from the value set, this will be because the OS masks the value with &FF00 so that Page points to a page boundary.

New Page for Disc Systems.

Last month we gave a note on how to configure a disc-based machine for cassette use by typing *TAPE before loading.

To leave more space in the machine for programs you can, as we said, type PAGE=&E00. This re-allocates the disc workspace for program use. We should also have added that you should type NEW after the PAGE change, so as to cover all eventualities.

Program tested on 0·1 and 1·1 O.S.

ARTIST

(for a model B - or 32k with analogue interface)

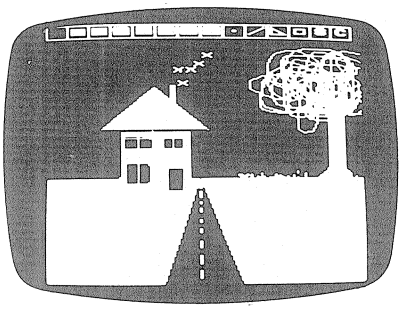
by J. Yale

A very nice drawing and painting package that operates from a single joystick (type ANH01) using a palette of colours and shapes chosen from the top of the screen. It draws points, lines, triangles and rectangles in any colour selected.

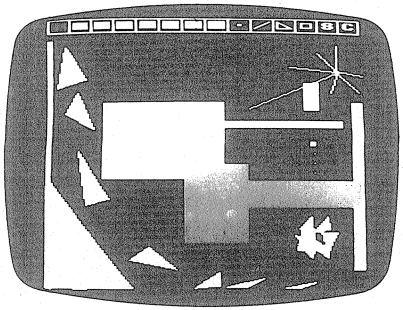
This sketching program uses one of the joysticks to position lines, triangles or rectangles on the screen. Once the program is running, the keyboard is not used at all, new colours and shapes being selected from a menu along the top of the screen using the cursor and fire button. The following functions are available, selected by the shape shown.

Point	Points are plotted at the cursor position as long as the fire button is pushed. Keeping it pressed draws continuously.
Line	Alternate pushes of the fire button leave a marker on the screen, or draw a line to the last marker which is then removed.
Triangle	Two markers must be positioned before the third press draws a triangle.
Rectangle	Position a marker at one corner of the rectangle, and then press at the opposite corner.
S	Save the screen to cassette or disc. This takes about 6 minutes on cassette, or 2 secs on disc! With cassette saving set the recorder to 'record' before pressing the joystick fire button. There is an option at the start of the program to read in a saved screen.
C	Clear the screen to the currently selected colour.

To investigate different plotting modes change the value of logic% in line 250. 1=OR 2=AND 3=EOR 4=INVERT. Note that rectangles do not plot correctly in EOR or INVERT due to overlapping of the two triangles making up the rectangle. Note also that due to a less publicised bug in OS 0.1, you will occasionally see unexpected lines drawn in the bottom right hand quarter of the screen if you are using the old operating system.



```
10 REM'ARTIST' by John Yale
20 REMSept 1982
30 REMBBC Model B with Joystick
40 ON ERROR GOTO 2100
50 MODE 7
55 *OPT1,0
60 PROCintro
70 MODE 2
80 VDU 23;8202;0;0;0;
90 PROCconstants
100 PROCinit_screen
110 REPEAT
120 REPEAT
130 PROCjoystick
140 UNTIL ADVAL(0)AND 1
150 IF y%>top_picture% THEN PROCfuncti
on ELSE PROCplot
160 UNTIL FALSE
170 END
180
190 DEF PROCconstants
200 LOCAL I
```



```
  210 points%=1 :line%=2 :triangle%=3
  220 rectangle%=4
  230 top_picture%=960
  240 cell_width%=88 :ncell%=14
  250 logic%=0 :colour%=7
  260 last_time%=TIME
  270 shape%=points%
  280 x%=640:x1%=x%:x2%=x%
  290 y%=top_picture%DIV 2
  300 y1%=y%:y2%=y%
  310
  320 REM Make all colours non-flashing
  330 FOR I=8 TO 15
  340 VDU 19,I,I-8;0; :NEXT I
  350
  360 REM Define special characters
  370 REM Point,Line,Triangle,Square,Cur
sor,Marker,Solid
  380 VDU 23,224,0,0,0,24,24,0,0,0
  390 VDU 23,225,1,2,4,8,16,32,64,128
  400 VDU 23,226,128,192,160,144,136,132
,130,255
  410 VDU 23,227,0,126,66,66,66,66,126,0
  420 VDU 23,253,0,68,40,16,40,68,0,0
  430 VDU 23,254,16,16,16,254,16,16,16,0
  440 VDU 23,255,255,255,255,255,255,255
,255,255
  450 ENDPROC
  460
  470 DEF PROCinit_screen
  480 LOCAL I
  490 REM Draw the cells
  500 GCOL 0,7
  510 MOVE 0,top_picture%+4
  520 DRAW ncell%*cell_width%,top_pictur
e%+4
  530 FOR I=0 TO ncell%
  540 MOVE I*cell_width%,top_picture%+4
  550 DRAW I*cell_width%,1020
  560 NEXT I
  570 DRAW 0,1020
  580
  590 REM Fill in the cells
  600 REM First the colours
  610 FOR I=0 TO 7
  620 PROCprint_cell(I,I,255)
  630 NEXT I
  640 REM Then the special shapes
  650 FOR I=8 TO 11
  660 PROCprint_cell(I,7,216+I)
  670 NEXT I
  680 PROCprint_cell(12,7,ASC("S"))
  690 PROCprint_cell(13,7,ASC("C"))
  695 IF load THEN *LOAD"SCREEN"
  700 PROCcursor(x%,y%)
  710 ENDPROC
  720
  730 DEF PROCprint_cell(N,colour,char)
  740 REM Print 'char' in cell 'N' in gi
ven colour
  750 VDU 5
  760 MOVE N*cell_width%+16,top_picture%
+48
  770 GCOL 0,colour
  780 PRINT CHR$(char);
  790 VDU 4
  800 ENDPROC
  810
  820 DEF PROCcursor(x%,y%)
  830 REM Print the cursor,centre x%,y%
  840 REM Invert the colour there
  850 GCOL 4,0
  860 MOVE x%-24,y%+12
  870 VDU 5,254,4
  880 ENDPROC
  890
  900 REM Put or remove a marker
  910 DEF PROCmarker(x%,y%)
  920 GCOL 4,0 :MOVE x%-24,y%+12
  930 VDU 5,253,4
  940 REM Restore state
  950 GCOL logic%,colour%
  960 MOVE x2%,y2% :MOVE x1%,y1%
  970 ENDPROC
  980
  990 DEF PROCjoystick
 1000 REM Read the joystick and plot
 1010 REM new cursor if it has moved
 1020 oldx%=x%:oldy%=y%
 1030 x%=(x%+ADVAL(2)DIV 45)DIV 2
 1040 y%=(y%+ADVAL(1)DIV 64)DIV 2
 1050 IF x%DIV8=oldx%DIV8 AND y%DIV4=old
y%DIV4 ENDPROC
 1060 PROCcursor(oldx%,oldy%)
 1070 PROCcursor(x%,y%)
 1080 ENDPROC
 1090
 1100 REM Wait for the button release
 1110 DEF PROCbutton_release
 1120 REPEAT UNTIL (ADVAL(0)AND 1)=0
 1130 ENDPROC
 1140
 1150 DEF PROCfunction
 1160 REM Service the selected function
 1170 REM given by cursor position
 1180 LOCAL cell
 1190 SOUND 1,-15,50,2
 1200 cell=x%DIV cell_width%
 1210 IF cell<8 THEN colour%=cell
 1220 IF cell=8 THEN shape%=points%
 1230 IF cell=9 THEN shape%=line%
 1240 IF cell=10THEN shape%=triangle%
 1250 IF cell=11THEN shape%=rectangle%
 1260 IF cell=12THEN PROCfile
 1270 IF cell=13THEN PROCclear_screen
 1280 IF cell>7 THEN count%=0
 1290 PROCbutton_release
 1300 ENDPROC
 1310
 1320 REM Plot the current shape
 1330 DEF PROCplot
 1340 PROCcursor(x%,y%)
```

```
1350 IF shape%<>points% SOUND 1,-15,100
,2
1360 MOVE x2%,y2% :MOVE x1%,y1%
1370 GCOL logic%,colour%
1380 ON shape% GOTO 1390,1400,1410,1420
1390 PROCpoints :GOTO 1440
1400 PROCline :GOTO 1440
1410 PROCtriangle :GOTO 1440
1420 PROCrectangle :GOTO 1440
1430
1440 PROCcursor(x%,y%)
1450 IF shape%<>points% PROCbutton_rele
ase
1460 x2%=x1% :y2%=y1%
1470 x1%=x%  :y1%=y%
1480 ENDPROC
1490
1500 REM Points
1510 DEF PROCpoints
1520 IF last_time%>TIME-20 PLOT 5,x%,y%
ELSE SOUND 1,-15,100,2:PLOT 69,x%,y%
1530 last_time%=TIME
1540 ENDPROC
1550
1560 REM Line
1570 DEF PROCline
1580 count%=count%EOR 1
1590 IF count% PROCmarker(x%,y%) ELSE P
ROCmarker(x1%,y1%) :PLOT 5,x%,y%
1600 ENDPROC
1610
1620 REM Triangle
1630 DEF PROCtriangle
1640 count%=(count%+1)MOD 3
1650 IF count%<>0 PROCmarker(x%,y%)ELSE
PROCmarker(x2%,y2%):PROCmarker(x1%,y1%)
:PLOT 85,x%,y%
1660 ENDPROC
1670
1680 REM Rectangle
1690 DEF PROCrectangle
1700 count%=count%EOR 1
1710 IF count% PROCmarker(x%,y%) :GOTO1
750
1720 PROCmarker(x1%,y1%)
1730 MOVE x%,y1% :PLOT 85,x%,y%
1740 MOVE x1%,y1%:PLOT 85,x1%,y%
1750 ENDPROC
1760
1770 DEF PROCclear_screen
1780 REM Clear screen to current colour
1790 GCOL 0,colour%
1800 MOVE 0,0 :MOVE 0,top_picture%
1810 PLOT 85,1280,0
1820 PLOT 85,1280,top_picture%
1830 ENDPROC
1840
1850 DEF PROCintro
1860 FOR Y=10 TO 11
1870 PRINT TAB(10,Y);CHR$132;CHR$157;CH
R$131;CHR$141;"ARTIST   ";CHR$156
1880 NEXT Y
1890 PRINT TAB(5,15);"Do you want instr
uctions (Y/N)";
1900 IF (GET AND &DF)=ASC("N") THEN 207
2
1910 CLS
1920 PRINT'''"The cursor is moved with t
he left"
1930 PRINT"hand joystick."
1940 PRINT'"Pressing the fire button wi
ll plot"
1950 PRINT"a point."
1960 PRINT'"Positioning the cursor on t
he top line"
1970 PRINT"and pressing the fire button
 has"
1980 PRINT"the following effect:"
1990 PRINT'"COLOURS      - Change plot c
olour"
2000 PRINT "DOT          - Doodle mode"
2010 PRINT "LINE         - Plot lines"
2020 PRINT "TRIANGLE     - Plot triangle
s"
2030 PRINT "RECTANGLE    - Plot rectangl
es"
2040 PRINT"S            - Save screen"
2050 PRINT "C            - Clear screen"
2060 PRINT'''"Push any key to continue"
2070 REPEAT UNTIL GET
2071 CLS
2072 PRINTTAB(5,15)"Load picture from f
ile (Y/N)?  ";
2074 load=((GET AND &DF)=ASC("Y"))
2080 ENDPROC
2090
2100 REM Error routine
2110 MODE 7
2120 REPORT:PRINT" @ ";ERL
2130 END
2140
2150 DEF PROCfile
2160 *SAVE"SCREEN" 3280 7FFF
2170 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

WHAT MODE? (Not OS 0.1)

You may not have noticed that OSBYTE 135 besides reading the character at the cursor position, returns the current graphics mode in the Y register. Therefore to set the variable mode% to the current mode:

```
A%=135
mode%=(USR(&FFF4) AND &FF0000) DIV &10000
```

DISCOUNTS

BEEBUG has arranged discounts for members at a number of retail outlets who supply computer books, software, hardware and services. We are continually negotiating further discounts.

We have heard that some suppliers are selling 120ns 4816 memory chips for Beeb upgrades. The Acorn specification is for a 100ns device (eg 4816-3), though we have not heard of anyone having problems with 120ns chips.

H. Banton 8 Princess Road Urmston Manchester M31 3SS Tel:061 747 7014	Printed cards to label the user-definable keys. £1.80/10, £2.40/20 £4.60/50 10% member discount.
Happy Memories Gladestry Kington Herefordshire HR5 3NY (Tel: 054422 618)	Computer Hardware 10% off all 'one-off' prices. Quantity prices may be negotiable.
Kingsley TV Services 40 Shields Road Newcastle upon Tyne (Tel:.0632 650653)	£10 discount off their range of Monitors and TV/Monitors.
Microage 135, Hale Lane Edgware Middx HA8 9QP	5% discount off disc drives mentioned in the "Disc Review" in this issue.
Mine of Information (Mail Order) 1 Francis Avenue St Albans Herts (Tel: 0727 52801)	Computer Books 5% discount
Opus Supplies Birchtrees 10 Beckenham Grove Shortlands Bromley BR2 0JU	Computer furniture 20% discount. Send for brochure.
Technomatic Ltd 15 Burnley Road LONDON NW10 (Tel: 01-452 1500)	Hardware, software and books, 5% discount
Wallace J & S 9 Barn Close Crewkerne TA18 8BL	Flexible plastic dust covers. £3.95. 25% discount to members.
Watford Electronics 33, Cardiff Road Watford Herts. (Tel: Watford 40588)	5% off most items.

It is advisable to telephone before placing an order, to check availability. Members should simply quote their membership number with their order, though members taking discount will not necessarily be given credit card facilities, (you must check this). We are not acting for these companies, nor receiving payment from them, and cannot be held responsible for their services.

BEEBUG SOFTWARE LIBRARY

GAMES 1 STARFIRE (32k) This is a very well written star-wars type game, with first class sound and graphics. The object is to
£3.50 get the enemy ships within your sights and destroy them with your lasers before they use up too much of your protective force field. You should try this at warp 3! This version now incorporates faster and smoother key response. BEEBUG 7 tells you how to incorporate Joysticks to Starfire.

GAMES 2 MOON LANDER (16k) Land the module in the crater with a speed less than 15km/hr. The program has fuel and speed screen
£3.50 readouts, with left-right and vertical thrust controls. The craft leaves a vapour(!) trail so that you can see your trajectory. There are two degrees of difficulty.

3D NOUGHTS & CROSSES (32k) 3-dimensional noughts and crosses played on a 4 x 4 x 4 board. It has a colour 3-dimensional graphics display, and the machine plays a pretty good game.

GAMES 3 SHAPE MATCH (16k) A two player game with nice visuals - a joint pattern recognition and memory test. Remember the
£3.50 positions of the shapes and match them up to win.

MINDBENDER (16k) A Beeb implementation of a "Mastermind" type game ("Mastermind" is a trademark of Invicta Plastics). You can choose the number of columns (1-10), and the machine assigns any of 7 coloured letters to each column. You make your guesses, and the machine tells you your score each time. There is a special facility to give the same combination to two or more players, so that the game can be used competitively in a completely fair way. This is a very good test of your logical powers, especially with 10 columns.

GAMES 4 MAGIC EEL (32k) Excellent fast moving SNAKE-type arcade game. Guide the colourful Eel around the screen gobbling up
£3.50 everything it comes across, it gets longer and faster with every bite. When you have eaten through the first frame, this is replaced with harder ones - faster with more obstacles. Fast and compulsive - don't be put off by the low price. This is a highly addictive game for all ages.

GAMES 5 CYLON ATTACK (32k) By the writer of Starfire. Fast moving invader-type game. Cylon ships hover in formation before
£3.50 dive-bombing and firing at your laser base. Quick reactions are needed to shoot as many down as possible before they re-group. Masterful use of sound and graphics.

GAMES 6 ASTRO TRACKER (32k) This is just about the ultimate in arcade games of the Asteroid variety. A brilliantly written
£5.75 machine code program giving excellent visual effects with fast moving action simultaneously throughout the whole visual field.

You control a spacecraft using left, right and forward thrust, plus hyperspace and a fire button. The idea is to fire at the asteroids before they home in on you. As they break up, so the game gets faster, and two types of alien craft appear. They make uncannily purposeful, and yet unpredictable, attacks on you - with one sort firing directly at you, and the other sort - the astro-tracker, tracking and divebombing you. If you survive the first screen, there are plenty more; with the action getting faster and faster. A very good game indeed.

UTILITIES 1 - DISASSEMBLER (16k) Read the machines ROMs (and EPROMs) with this nicely written disassembler. It gives you the
£4.00 full 6502 mnemonics. There is also a column with ASCII codes of each byte, allowing you to spot embedded text immediately. To use it just enter the start address, and page mode is engaged, displaying 20 lines or so at each press of the space bar.

REDEFINE (16k) A very useful graphics tool. Redefine allows you to build up user defined graphics characters. You enter points on an 8 x 8 grid using the cursor control keys, the program constantly displays the character so you can see it build up, and also the VDU 23 command that you need to use to create that character in your program. The program will literally save you hours.

MINI TEXT ED (32k) A mini word processor substitute. It uses the machine's Basic editing facilities to allow text to be entered, edited, saved, loaded, and printed to the screen or printer. The printer routine contains a subroutine to remove the Basic line numbers. Note that this will not, as it stands, right-justify, and will not automatically close up the text after editing. It nevertheless provides a very useful facility.

APPLICATIONS 1 - SUPERPLOT (32k) Produces tailored screen representations of any function entered. This can be achieved in
£5.20 any of the three major coordinate systems: Cartesian, Polar, or Parametric. SUPERPLOT comes complete with a 7-page instruction booklet. Explore the world of graphic representation.

Here is just one unsolicited response from a member: "As a mathematician, I find SuperPlot very exciting - and as a teacher, I find it commendable in that it doesn't attempt to replace a teacher, but to augment teaching in a unique way. It is really very good". - G.P. North Yorks.

APPLICATIONS 2 - MASTERFILE (32k) This is a general purpose file management program. Its uses are manifold; for example you
£5.75 can file: A BEEBUG magazine index; Names and Addresses of friends; School Class lists; Book lists; Client/Customer lists; Record collection etc. The program can hold up to 550 records with only one field, but more practically it can hold as many as 100 records with 5 fields.

Features incorporated in the program are - Save file on cassette; Load back a previously recorded file; Display individual records on the screen or printer; Search file for a particular match; Sort file on any item or items; Printout of address labels.

The program comes complete with a dummy data file on cassette for experimentation purposes, and an extensive manual.

This extremely versatile program would cost well over £20 commercially and in fact we at BEEBUG use it for a variety of purposes. We are offering it to members at the special price of £5.75 inc VAT. [A separate disc version of MASTERFILE will be available in the near future.]

All prices now include VAT. Please add 50p postage and packing to all orders regardless of size. Post to BEEBUG Software, 374 Wandsworth Road, London SW8 4TE.
The above postal rates are for the UK only. Overseas enquiries welcome.

IF YOU WRITE TO US

Back Issues (Members only)

All back issues are kept in print (from April 1982). Send 90p per issue PLUS an A5 SAE to the subscriptions address. This offer is for members only, so it is essential to quote your membership number with your order. (In difficulty please phone 01-720 9314).

Subscriptions Address

BEEBUG
Dept 1
374 Wandsworth Rd
London
SW8 4TE

Subscriptions

Send all applications for membership, subscription renewals, and subscription queries to the subscriptions address.

Membership costs: £5.40 for 6 months (5 issues)
: £9.90 for 1 year (10 issues)
European Membership £16 for 1 year.
Elsewhere - Postal Zone A £19, Zone B £21, Zone C £23

Software (Members only)

Available from the subscriptions address. (Except Wordwise, see Editorial for details).

Contributions and Technical Enquiries

Please send all editorial material to the editorial address opposite. If you require a reply it is essential to quote your membership number and enclose an SAE.

Editorial Address

BEEBUG
PO Box 50
St Albans
Herts
AL1 2AR

Ideas, Hints & Tips, Programs, and Longer Articles

Substantial articles are particularly welcome and we will pay for these! But in this case, please give us warning of anything that you intend to write. In the case of material longer than a page, we would prefer this to be submitted on cassette or disc in a format similar to that produced by our Mini Text editor (See program in this issue or Utilities 1 cassette) or as a Beeb datafile providing that you use the bugs fix on page 21 of the July issue, or you have the 1.0 (or later) operating system. If you send on cassette please record two copies, one of them at 300 baud. We can also accept Wordwise (menu option 1) files on cassette or disc.

BEEBUG NEWSLETTER is edited and produced by Sheridan Williams and Dr David Graham.
Production Editor Phyllida Vanstone.
Thanks are due to Rob Pickering, John Yale, Adrian Calcraft, and Colin Opie for assistance with this issue.

Printed in England by Staples Printers St Albans Limited at The Priory Press. ISSN 0263-7561