£1.20

BEEBUG

FOR THE BBC MICRO

Vol 3 No 6 NOVEMBER 1984

Elite – the new supergame



- Build a graphics tablet
- Midland homebanking experiment
- Auto-keyword generator
- Cross reference lister
- Acornsoft ISO-Pascal reviewed
- Basic compilers reviewed
- Number hunt game
- Wee Shuggy game
- Latest ROM/RAM boards reviewed
- Adventure games
- And much more

BRITAIN'S LARGEST COMPUTER USER GROUP
MEMBERSHIP EXCEEDS 25,000

EDITORIAL

MORE PAGES IN BEEBUG

This is the first regular issue of BEEBUG with an extra four pages. We hope that in the next few months that this will enable us to include more reviews and other information as well as some extra programs and other features. We shall be taking advantage of the extra pages next month to add a Christmas flavour to the magazine.

DNFS ROM FROM ACORN

The DNFS ROM, supplied as part of Acorn's second processor systems, is now available separately for around £20.60 inc. VAT. This provides later, improved versions of the Disc and Network Filing Systems. The chip is accompanied by a 10 page booklet. The Acorn reference number is ANB21.

NOTICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BO

NEW RELEASES FROM BEEBUGSOFT

You should receive a copy of our new software brochure with this magazine mailing. As you will see, we have considerably expanded the range of our software, and have upgraded a number of our current titles. For example Masterfile II allows very flexible print options, and can even print documents from Wordwise or View, inserting specified data from Masterfile - e.g. insert the name, address and other particulars into circulars etc. Exmon II incorporates a full screen editor, and dual screen facilites, and we have a ROM based version of Spellcheck which runs up to 10 times faster than the disc based version. Upgrades for all these products are available at 50% discount on trade in of previous versions.

Sleuth is a totally new ROM offering dual screen single step debugging in Basic. Our Music Extension ROM is also worth close examination, and we are now offering our own brand of full specification discs which are fully tested and guaranteed, and supplied in a free case in quantities of 10, 25 and 30.

We are now taking Access and Barclaycard orders on all products - ring Penn (0494810) 6666 (multi-lines, 24 hour service). If you are taking a member's discount it is essential to quote your membership number.

In expanding our range of products we have regretfully had to phase out our software club offers. See this month's supplement for stock clearance price cuts.

PRINTMASTER ROM UPDATE

The STAR compatible version of Printmaster is now available at the same price as the existing Epson version. The new release will work with STAR printer models DELTA, DP, GEMINI and RADIX.

Computer Concepts have also stated that they believe the Epson version of Printmaster will work with the following compatible printers: KAGA TAXAN, CANON, MANNESMAN TALLY MT80.

HINT WINNERS

This month F.Duerden is the winner of the £10 prize, and S.R.Linton and D.Long both win £5 prizes. We are always please to receive your latest hints for the magazine.

MAGAZINE CASSETTE/DISC

This month we have included one extra program on the magazine cassette/disc. This is a copy of the PACK program by David Tall first published in BEEBUG Vol.1 No.9, and is intended to complement the Cross Reference program included in this month's magazine. The PACK program can be used to remove unnecessary spaces and comments from programs to save space.

BEEBUG MAGAZINE

GENERAL CONTENTS

2 Editorial
4 Acornsoft ISO-Pascal Reviewed
7 ELITE – An Outstanding New Game from Acornsoft
8 Auto-Keyword Generator
10 The Midland Home Banking Experiment
13 Build Your Own Graphics Tablet (Part 1)
16 Adventure Games
18 Domestic Accounts Extended
19 New Edition of Birnbaum's Book
20 Two Basic Compilers Reviewed
22 Points Arising
23 Improved Function Key Labels
25 Beginners Start Here
Debugging Programs (Part 1)
28 Cross Reference Lister
33 The Latest ROM and RAM Boards Reviewed
36 BEEBUG Workshop
Formatting Text
38 Watford Electronics' ROM Manager
39 Wee Shuggy
44 Number Hunt

HINTS, TIPS & INFO

12 Number of Characters per Line
12 Disc Tips
15 Faster Basic
15 Cassette Usage of Wordwise
15 Hobbit and GDUMP
15 Torch Corruption
15 Doctor Soft Mods
27 Another Use for *BASIC
27 Amusing *FX Call
27 Power up Reset
27 Basically Mistaken
27 Length of a File
27 Long Disc Files
32 Delay Loop
38 6502 Tube and the Graphics ROM

PROGRAMS

8 Auto-Keyword Generator
18 Domestic Accounts Extended
23 Function Key Labels
28 Cross Reference Lister
36 Text Formatting Routine
39 Wee Shuggy Game
44 Number Hunt Game

ACORNSOFT ISO-PASCAL

Reviewed by Matthew Rapier and Paul Spurgeon

At last a full implementation of Pascal is available for the BBC micro. We report on ISO-Pascal, a major new programming language system from Acornsoft, due to be launched in mid November.

Product:	ISO-PASCAL
Supplier:	Acornsoft
Price:	£69.00 inc VAT.

ISO-PASCAL from Acornsoft is a full implementation of the Pascal language to the latest ISO standard. Pascal is very popular as a teaching language. However, it is not just limited to use in Universities and Colleges, but is often used for systems programming and other applications on all manner of computer systems.

Compared to S-Pascal and the HCCS Pascal, this is surely a superb system. At £69 it is one of the cheapest full language systems available for the Beeb as an an alternative to Basic.

A BRIEF EXPLANATION OF PASCAL

Pascal is an example of what is termed a 'block structured' language. For those unfamiliar with block structured languages it is worthwhile giving a brief explanation of Pascal. It is impossible here to describe all the features of the language, and for further information you should consult some of the many books that explain the language more fully.

Pascal was devised by Niklaus Wirth at the Tecnological University of Zurich to embody the best principles of program structure and design. Pascal is a well defined language with a recognised standard (see Jensen and Wirth, "Pascal User Manual and Report", Springer -Verlag, New York 1975), which has deservedly established a world wide reputation.

Pascal programs are entered and modified using a text editor (akin to Wordwise or View on the BBC micro), and without line numbers. At this stage a Pascal program is simply a standard text file and this allows instructions to be laid out in a way which emphasizes the structure of the program as shown in the example:

```
WHILE p > start DO
  BEGIN
  q:=start+1;
  p:=p-1;
  WHILE q < fin DO
    BEGIN
    q:=q+1;
    IF NOT costfixed THEN
      IF cost+max<total THEN
        BEGIN
        total:=cost+max; min:=w
        END
    END
  END
```

Once the text of the Pascal program has been entered and saved on disc, it can then be compiled to machine code or similar using a Pascal compiler. It is this 'object code' file which is then executed to run the program.

One important feature of Pascal is the way in which 'compound' statements can be constructed consisting of simple statements enclosed between 'BEGIN' and 'END'. A compound statement can be used anywhere in the program in place of a simple statement. This leads to the nested block structure so typical of Pascal and encourages a well structured approach to the writing of Pascal programs. This is helped by a comprehensive set of structured control statements including not only the REPEAT-UNTIL that is part of BBC Basic, but also a WHILE-DO and most useful, a CASE statement which provides for multiple branching usually handled by the IF-THEN-ELSE or ON-GOTO type of construction in Basic.

Procedures and functions also feature strongly in Pascal, though the Pascal versions are both more powerful and flexible than in BBC Basic. They can be thought of in simple terms as a block with a name. The block of the procedure can refer to previously defined procedures and to itself,

allowing recursion. The use of parameters and local variables ensures that both functions and procedures can be written quite independently of each other and the main program.

Another characteristic of Pascal is its insistence that the data type of every variable be declared before use. Not only that, but Pascal also allows you to define your own data types. In a procedure definition the type of each parameter is explicitly stated and you can only pass variables of the correct type to that procedure. This strong data typing helps to structure the program and often highlights potential errors caused by the mis-match of data types at an early stage of program development. The option to define your own data types also allows complex data structures to be introduced into a program with minimal difficulty.

ACORNSOFT ISO-PASCAL

ISO-Pascal is supplied on two 16K ROMs for standard BBC machines and on disc for use with the 65Ø2 second processor. One ROM contains a machine code interpreter, to execute the intermediate code generated by the compiler, and also the editor and other Pascal system commands. The other ROM contains the compiler but this still needs the interpreter to run. Pascal programs are standard ASCII text files which are then compiled, not directly into machine code, but into an intermediate code (called BL code). This is then interpreted by the BL interpreter in ROM. The Pascal compiler is itself in BL code (presumably compiled from Pascal) which is why the compiler needs the interpreter.

Switching between the two ROMs is automatically controlled by the Pascal interpreter ROM. The system can be used with any filing system, but to use full error reporting a random access system like DFS or Econet is necessary. The system will work with cassette files but developing large programs in this way would be very tedious.

THE EDITOR

A text editor is provided as part of the main ROM for the creation and editing of Pascal programs, though other text editors such as View or Wordwise could equally well be used. The supplied editor is quite powerful with screen editing using the cursor and function keys, and includes a very flexible search and replace facility. You can search not only for alpha-numerics (including spaces), but also for ranges of characters, not ranges of characters and so on, making this a very useful and powerful feature.

THE COMPILER

The ROM based compiler will compile a program from either a disc file or from the current memory file created by the editor. The intermediate code file in turn can be sent either to disc or stored in memory. This provides a fast development environment for small programs as both source and object code may be kept together in memory. The disc based compiler for the 65Ø2 second processor system does not have this facility, and can only compile from and to disc. Compiling a Pascal source program provides a line numbered listing of the program on the screen (which can be turned off) as compilation proceeds.

The single pass compiler produces runnable intermediate code if no errors are encountered. If an error is found then the compiler pauses, giving an error code, and if an error file is available on disc, a descriptive message. It also identifies the line in error by giving a line number, and also points to the approximate position of the error. The editor is able to search for and locate a specified line number in the Pascal source program (remember that this does not explicitly include line numbers), a feature not usually available in word processors. Not only did the compiler seem to run slowly (with or without the listing being displayed on the screen), but error messages took a surprisingly long time for the compiler to fetch from disc.

The disc version of the compiler implements the full ISO specification to level 1a, while the ROM version is to the lower level Øb (thus no support for conformant arrays, and packing has no effect on strings).

THE EXTENSIONS

Acornsoft have added many extensions

to their ISO-Pascal to allow access to the BBC micro's input and output routines. These are, of course, non-standard (in Pascal), and mostly mimic the operation of BBC Basic functions (including MODE, VDU, PLOT, POINT, SOUND, ENVELOPE, OSCLI, ADVAL, INKEY, TIME, and SETTIME). Other useful extensions provide for numeric/string conversion, the use of external file names, dynamic storage management and machine code calls. The latter allow other operating system routines to be called quite easily, but calling your own machine code is more complicated, though it is all explained in the manual with examples. Although these extensions make a useful addition to the language, they are not part of the accepted standard for Pascal.

HOW WELL DOES THE SYSTEM WORK?

Our impression was that the speed of execution of the compiled code was not always as fast as expected. In some examples that we tried, Basic was very slightly faster than Pascal. Run time error reporting is somewhat cryptic, and there are no extensions allowing for error trapping within a program. A simple trace facility can be enabled which prints out line numbers and procedure names as they are executed. Although it is fairly simple to write comprehensive debugging code into your program, it is much easier if the system does it all for you.

DOCUMENTATION

The system is delivered with a substantial manual, and the book "Pascal from BASIC" by P.J.Brown. The manual contains chapters on the editor, compilation, compiler options, language extensions, memory organisation, internal representations, machine code linking and various appendices. We were confused by the manual, with its large section devoted to the editor right at the start, and the chapter on using the compiler is spoilt by continual reference to memory organisation (high water mark etc.). Apart from this the manual is comprehensive and detailed in the information it provides.

THE DISADVANTAGES

There are two drawbacks to this Pascal system. There is firstly no provision for segmenting a large program into separately compiled modules. While not standard Pascal, it is generally acknowledged that it is good practice to decompose a program into relatively independent and self-contained modules. Thus, for example, it would be useful to write a module implementing graphics calls to the BBC's operating system. A sample program showing how procedures such as move(x,y) and draw(x,y) may be written is provided, but it is impossible to compile this as a separate library module. The best that can be done is to include text files within the Pascal source program, chaining files together and creating one monolithic program, all of which has to be compiled everytime. Quite large programs can be compiled in this way.

Secondly, as already described, ISO-Pascal compiles to intermediate BL code, which can be saved to tape or disc. This code can only be executed by using the interpreter ROM, or (with a second processor) by using the disc based run-time system. There is no way of producing a compiled program that can then be run by other users, unless they buy ISO-Pascal as well. It may be that, as with BCPL, Acornsoft will produce a stand-alone package for Pascal.

THE GOOD POINTS

The system has an excellent text editor which is ideal for program development, and the compiler has good error reporting facilities which pin-point errors, at least as far as syntax is concerned. The extensions allow simple use of all the BBC micro's facilities without resorting to any complicated programming techniques. Overall, Acornsoft have achieved a notable success in implementing a full version of Pascal on the BBC micro, notwithstanding the criticisms above, and at the reasonable price of £69 for the complete ISO-Pascal system.

IN CONCLUSION

At present the main interest in this product is likely to be from lecturers and teachers seeking to use Pascal to teach computer programming, and from individuals who are interested in programming in this powerful language. For all of these, Acornsoft's ISO-Pascal provides a major new language for the BBC micro.

ELITE – AN OUTSTANDING NEW GAME FROM ACORNSOFT

Reviewed by David Fell

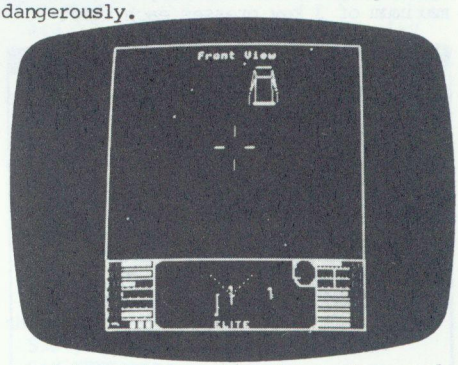
Name : Elite
Supplier : Acornsoft
Price : Tape £14.95 inc. VAT
Disc £17.65 inc. VAT
Rating : *****



"Acornsoft Elite is the first in a new generation of 3-D space games featuring interstellar travel in a distant cluster of galaxies..." said the advertising. Thoughts of an over-hyped third rate nightmare started to loom close as, with a degree of trepidation, I loaded Elite. That was just over a week ago, and I'm now convinced that Acornsoft have just released the best game ever for the Beeb.

Elite combines the elements of a number of classical games to produce a superb three dimensional graphical trading game of skill and absolute addiction, that squeezes every ounce of performance out of the Beeb. Let me explain this further; Elite combines together a three dimensional fast action combat game, a 'Monopoly' like trading game and various aspects of an adventure style exploration game. Elite takes its name from the overall goal of the game, which is to achieve a rating of Elite (the highest accolade available which, as yet, has not been reached by any mortal). To achieve this you need to be a successful space trader (to purchase the necessary weapons) and to achieve a high level of flying and combat skills (your rating is based upon the latter, and your survival is dependent upon both).

In Elite you pilot a Cobra MK3 space ship equiped with normal and hyperspacial drives. Combat, normal flight and docking are all performed in real time, with some stunning graphics achieving an amazing impression of realism. Docking at an orbiting space station is essential if you are to be able to trade. The trading itself is quite complex due to the sheer quantity of information involved: 2000 planets selling 17 different items; political situation; planetary produce, etc. If all this sounds too dull you can always give up the life of the honest trader and become a bounty hunter or space pirate, but this involves living life dangerously.



Combat takes a while to master, and requires sheer determination and three dimensional perception to survive. Your armoury, apart from three missiles and a small laser, will depend upon accruing money from trading. For a real demonstration of combat, go to your nearest dealer, and he should have an Elite demonstration disc. As a warning, it is said that there are spaceships 'out there' that no one has ever seen!

CONCLUSION

Elite is undoubtably a masterpiece of programming that I would recommend anyone who has a Beeb to purchase as soon as possible. Even now, just a week after it's launch, Elite has already firmly established itself as a cult game for the Beeb that seems to create its own self perpetuating fame. There is a monthly competition for players of Elite, and full details are included with the game.

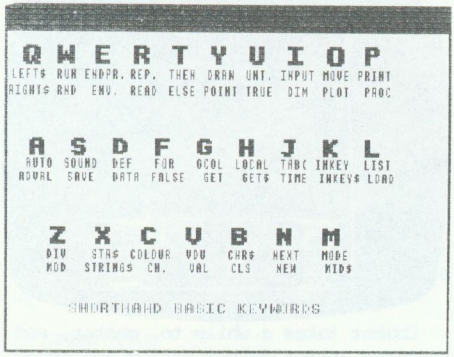
Tested on Basic I & II and O.S. 1.2

AUTO—KEYWORD GENERATOR (16k)

by John Bower

Many microcomputers, including the Electron for example, allow you to enter Basic keywords with just a few key strokes directly from the keyboard. John Bower describes a short utility to add this most useful feature to the BBC micro, to save both time and effort when typing programs into your Beeb.

The purpose of this program is to allow the user to type in a whole keyword with ease and speed. Wouldn't it be nice to be able to type in the keyword ENVELOPE by pressing only three keys instead of eight, or to type in ENDPROC with just two keys instead of seven? This utility will allow you to do not only that, but will allow you to type in up to 52 Basic keywords with a maximum of 3 key presses each time.



Using this program you can enter any of the 52 Basic keywords using either two or three key strokes in each case. The first 26 keywords are generated by pressing Tab followed by one of the 26 alphabetic keys, and the other 26 by using Ctrl/Tab together first. The keywords generated in this way are shown in the diagram - those using Tab alone at the top and those using Ctrl/Tab at the bottom of each pair. Even if you don't use all 52 keywords generated in this way, you may still find some of the Tab combinations particularly useful and convenient. You can also decide which 52 keywords you want to include. You may also find it helpful to copy the keyword diagram and place it above the function keys of your machine for reference.

To use the program, just type it in and save it on cassette or disc (you can omit all the assembler comments if you wish - that is text following '\'). If you are using a cassette system you will need to change the address for the location of the machine code. This is at line 110 where &A00 should be changed to &D00. A further change is required to allow this program to run properly using Basic I. The information is contained in the program as a comment at line 360, and and involves changing the first instruction on line 370 from LDA #&74 (for Basic II) to LDA #&70 (for Basic I). This is because the keyword look-up table starts in a different place.

Once saved you may then run the program and test it out. To do this just press the Tab key once, and then press the 'A' key. You should now see the keyword AUTO appear on the screen. Now press Tab followed by all the other keys between A and Z to check that they all work. If some do not give the expected results, check line 600 to make sure that the values of the tokens are correct.

Next you can test if the second set of keywords work. To do this hold down the 'CTRL' key,and at the same time press 'TAB' (represented as Ctrl/Tab). Now release both keys and press a key between A and Z. You should now get more keywords, but different to those before.

If they all work (corresponding to the diagram), you can customise the program to suit your own needs. You can do this by changing lines 600, 610, 630 and 640. If for example you wish to substitute the keyword OPENOUT for GET, you can do this by finding the token for OPENOUT (&AE) from the User Guide (page 483) and putting that in place of

the token for GET (&A5) in line 640. It would also be prudent to replace the keyword GET by OPENOUT in the REM statement at line 630. The REM statements at lines 600 and 630 serve as reference guides to the current token settings in lines 610 and 640.

Once you are satisfied with the keywords that you have set up, then you can save the machine code by typing:

```
*SAVE KEYWORD A00 AFF
```

and re-run the program by typing:

```
*RUN KEYWORD
```

Cassette users should save their code by typing:

```
*SAVE KEYWORD D00 DFF.
```

You can, if you wish, use any name you choose rather than KEYWORD. To disable the effect at any time, type in the command *FX13,4 and to re-enable the code type *FX14,4. If you press Break you can restart the code by typing CALL &A00 (or CALL &D00 on cassette).

```
  10 REM PROGRAM SHRTHND
  20 REM VERSION B0.3
  30 REM AUTHOR  John Bower
  40 REM BEEBUG  NOVEMBER 1984
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 FOR pass=0 TO 2 STEP 2
 110 revector=&A00:REM &D00 for casset
te
 120 store=revector+&C0
 130 OSBYTE=&FFF4:OSRDCH=&FFE0
 140 P%=revector
 150 [OPT pass
 160 \ revector write character routine
 170 LDA &20E:STA &70
 180 LDA &20F:STA &71
 190 LDA #start MOD256:STA &20E
 200 LDA #start DIV256:STA &20F
 210 RTS
 220 .start \ save key press & registe
rs
 230 STA &74:TXA:PHA:TYA:PHA
 240 \ check for TAB key
 250 LDA &74:CMP #9:BEQ cntrl:JMP fini
sh
 260 .cntrl \ check for cntrl key/esca
pe
 270 LDA #&81:LDX #&FE:LDY #&FE:JSR OS
BYTE:CPX #&1B:BNE get2ndky:JMP finish
 280 .get2ndky \ get A to Z key/escape
 290 TXA:PHA:JSR OSRDCH:CMP #&1B:BNE l
owcase:PLA:JMP finish
 300 .lowcase \ mask for lower case &
adjust A-Z offset
 310 AND #&DF:SEC:SBC #65:CMP #0:BMI f
in2:CMP #26:BCS fin2:TAY
 320 \ if cntrl used advance data poin
ter
 330 PLA:CMP #&FF:BNE offset:TYA:CLC:A
DC #26:TAY
 340 .offset \ get token value from st
ore
 350 LDA store,Y:STA &79
 360 \ load ROM address of 1st BASIC I
I keyword token (for BASIC I use LDA #&
70)
 370 LDA #&74:STA &80:LDA #&80:STA &81
:LDY #0
 380 .search \ search ROM for token
 390 LDA (&80),Y:CMP &79:BEQ found
 400 INY:CPY #0:BNE search:INC &81:JMP
 search
 410 .found LDX #8 \ copy keyword
 420 .loop CPY #0:BNE loop1:DEC &81
 430 .loop1 DEY:LDA (&80),Y:CMP #&24:B
CC display
 440 CMP #&7F:BCC ok:INX:JMP display
 450 .ok STA &84,X:DEX:JMP loop
 460 .display \ put into keyboard buff
er
 470 INX:LDA &84,X:STX &82:TAY:LDA #&8
A:LDX #0:JSR OSBYTE:LDX &82:CPX #8:BEQ
over
 480 JMP display
 490 .over LDA #0:STA &74 \ cancel key
 press
 500 .finish \ restore regs & key press
 510 PLA:TAY:PLA:TAX:LDA &74:JMP (&70)
 520 .fin2:PLA:LDA #7:STA &74:JMP fini
sh
 530 ]
 540 NEXT pass
 550 FOR location=store TO store+51:RE
AD token:?location=token:NEXT
 560 REM store selected keyword tokens
 570 CALL revector
 580 END
 590 REM keyword tokens with TAB+ A-Z
 600 REM AUTO,CHR$,COLOUR,DEF,ENDPROC,
FOR,GCOL,LOCAL,INPUT,TAB(,INKEY,LIST,MO
DE,NEXT,MOVE,PRINT,LEFT$,REPEAT,SOUND,T
HEN,UNTIL,VDU,RUN,STR$,DRAW,DIV
 610 DATA &C6,&BD,&FB,&DD,&E1,&E3,&E6,
&EA,&E8,&8A,&A6,&C9,&EB,&ED,&EC,&F1,&C0
,&F5,&D4,&8C,&FD,&EF,&F9,&C3,&DF,&81
 620 REM keyword tokens with TAB+CNTRL
+A-Z
 630 REM ADVAL,CLS,CHAIN,DATA,ENVELOPE
,FALSE,GET,GET$,DIM,TIME,INKEY$,LOAD,MI
D$,NEW,PLOT,PROC,RIGHT$,READ,SAVE,ELSE,
TRUE,VAL,RND,STRING$,POINT,MOD
 640 DATA &96,&DB,&D7,&DC,&E2,&A3,&A5,
&BE,&DE,&91,&BF,&C8,&C1,&CA,&F0,&F2,&C2
,&F3,&CD,&8B,&B9,&BB,&B3,&C4,&B0,&83
```

THE MIDLAND HOME BANKING EXPERIMENT

by Peter Rochford

Homebanking, using a specially adapted television set or a microcomputer, has been talked about for many years. The rapid increase in interest and ownership of low-cost microcomputers has now made this a reality for some. Peter Rochford describes his own experiences with this new service using a BBC micro.

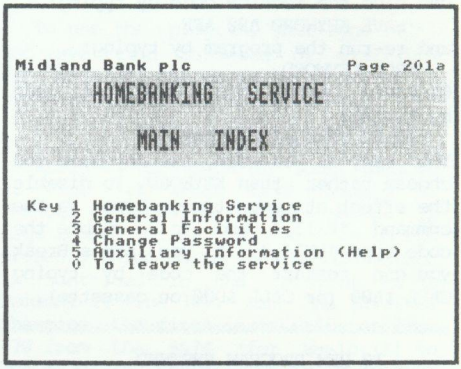
Some six months ago the Midland Bank quietly launched an experimental homebanking service for its customers. The Midland is the first of the 'big four' major banks to show any practical interest in electronic banking at the domestic user level.

The experiment was originally set to run for one year with a view to it being extended if proved successful. Those like myself, who applied to participate in the experiment, were sent a letter outlining how the system would work and details of what equipment (which includes a BBC micro with suitable Prestel type modem) would be needed by the user. During the experimental stage, no charge would be made in return for the feedback obtained from users and the resultant experience gained by the Midland Bank.

Once accepted and registered, a user is sent a rather impressive folder containing the phone numbers relevant to the system and an instruction manual. The manual is really excellent, providing easy step-by-step guides to logging on and accessing the many services available. Included in the manual are numerous facsimile screen displays and examples to help you find your way round the system.

Even at this experimental stage, the facilities and information provided are remarkably comprehensive. Obviously you can check the balance of your account(s) and look at your current or previous statement. What really impresses though, is the way in which you can use the system to interrogate your account to find individual details quickly and simply.

For example, debits can be found without having to scan through the entries of all the pages of your



statement. Instead, you can tell the system the number of the cheque you are looking for, or the value of the cheque, and it will find it for you, assuming of course that it has been entered in your account. As an alternative, you can enter start and end cheque numbers and all cheques between and including these numbers will then be displayed. You are also notified of the numbers of cheques that have not yet been cleared.

Credits can be searched for by just asking the system to search for a credit of a specific value. All credits which correspond to this value will then be displayed.

If you use a lot of standing orders homebanking is definitely for you. You can obtain a list of all your current standing orders and then go on to ask for a detailed look at any one you choose, or indeed all of them. You are given the name of the beneficiary, the bank and account into which it is paid, the frequency of payments, value of payments, when the next payment is due, and other details.

As well as examining your accounts, a recently added facility allows you to

transfer funds between your accounts on-screen. There is a set limit to the amount you can transfer in any one day and a limit to the number of transfers you can effect in one day.

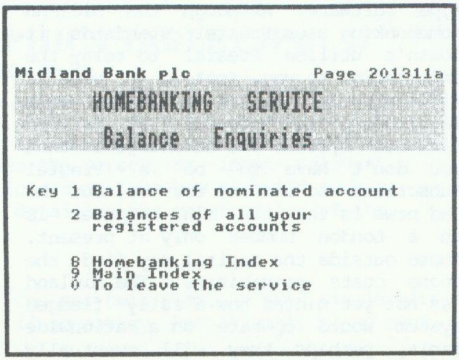
Apart from the details of your account, there are a whole host of information services at your disposal on such subjects as foreign currency, interest rates, exchange rates, autobanks, personal loans etc. You may also order a new cheque book or statement and any of the currently available literature on bank services.

A mailbox is provided for sending messages to other users of the system and a noticeboard where the bank provides details of changes to the services provided.

Because of the security aspect, logging-on to the Midland Bank computer does mean remembering several codes. Once you have dialled up the computer and obtained the welcome page on screen, you are asked to enter your user number. This is a permanently allocated 6 digit number which you can divulge to other users on the system allowing them to contact you by mailbox. Next you enter a password which is not shown on screen as you type it in. Finally, before entering the system, you have to enter your customer identity. This is another password, but unlike the first, can be of variable length between four and ten characters. Both the password and customer identity can be changed on-screen as and when you wish whilst connected to the system.

Having entered all the correct codes you then gain access to the system. You are now presented with the main menu page, but to obtain access to your accounts you must type in the relevant account numbers.

Should you make a mess of typing in any of the codes, you are allowed just two further attempts before the system will disconnect you. Further attempts by re-dialling and trying to log on, even with the correct codes, will be fruitless. Access can only be restored by writing to the homebanking unit who will also re-notify you of your codes.



In general, using the system is quite simple. There are lots of menus to guide you and in most cases it requires only a single key press to obtain the service you want. It is virtually impossible to 'get lost' in the system as there is always the facility to jump back to a menu from any page. The screen displays are very well laid out and plenty of use is made of colour to highlight various features.

To aid you in finding out about everything on offer, there is a 'rolling demonstration' that takes you through most of the services available, you just sit back and watch. This is a very helpful, not to mention intriguing facility. However, it takes quite a while so beware the cost to your telephone bill.

At present, there are certain restrictions on the system out of normal banking hours. This amounts to preventing you searching your account for credits and debits and obtaining detailed information on standing orders. This is quite significant as your telephone charges are higher during the day and you do not have the convenience of 24 hours a day full service which should surely be one of the main benefits of any homebanking system. Your balance and latest statement are always available, as are the general information and ordering services.

To use the service you must be a Midland Bank customer. Apart from that qualification, you will need your Beeb, a 1200/75 baud modem and Prestel

type software. Although the Midland homebanking uses Prestel standards it doesn't utilise Prestel to relay the information to you. Instead it uses a mini-computer, linked to the two bank mainframes, to provide a private Viewdata system. The advantage is that you don't have to be a Prestel subscriber to use the service, but the bad news is that the mini-computer is on a London number only at present. Those outside the capital may find the phone costs prohibitive. The Midland has not yet hinted how a fully fledged system would operate on a nationwide basis, perhaps they will eventually utilise Prestel.

My own personal feelings after using homebanking for six months are of great enthusiasm. I have found it well worth the cost of the phone calls and it has enabled me (much to my bank manager's joy!) to manage my account much better. How much benefit you derive from using homebanking will depend on how much use you normally make of your bank account. Businesses that need to keep close tabs on their finances I am sure will find it indispensable.

```
MIDLAND BANK plc                    Page 2411a
          BALANCE ENQUIRY
 Mr. A. B. Smith        C/A
 Account No 01234567  Branch 404788

 The balance of the above account at
 close of business on   23 May 83

 was          610.47 CREDIT

 Key 8 Homebanking Index
     9 Main Index
     0 To leave the service
```

```
MIDLAND BANK plc                    Page 2413a
          STANDING ORDERS  PAGE 1 OF 1
 Mr. A. B. Smith        C/A
 Account No 01234567  Branch 404788

                                          When
 Beneficiary                 Amount       Paid
 ABBEY NATIONAL B/S           20.00 Mthy
 INSURANCE                   134.40 Yrly
 GOLF CLUB                     5.50 Yrly
 COUNTY COUNCIL               27.84 Mthy
 A.SMITH BUDGET A/C           95.00 Mthy

 Key
     3 Details of individual order
     8 Homebanking Index
```

If the full range of services can be made available 24 hours a day and extended to cover more transactions, such as paying bills, obtaining loans etc, then homebanking will become a most useful facility for many people. Even so, the cost of the service is bound to be a significant factor as far as home users are concerned.

At the time of writing this article (July 84), the Midland Bank homebanking unit have indicated that the experiment is to be extended for a further year and that there is spare capacity on the system for anyone wishing to become a user of the service. For further details and to check availability at present, you should contact them at the address shown below.

Electronic Banking Development Unit,
Midland Bank plc,
27/32 Poultry,
London. EC2P 2BX.
Tel: 01 606 9911 Ext 3113.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

NUMBER OF CHARACTERS PER LINE

If you wish to calculate screen width dependent on the current mode, then the following formula will enable you to do this (C% will hold the number of characters on exit, and M% holds the mode number on entry):

```
IF M%<6 C%=2^(2-(M% MOD 3))*20 ELSE C%=40
```

DISC TIPS - Peter Chambers

When using either DNFS or Watford DFS with a *RUN !BOOT file, remember to include a CLI instruction before any OSRDCH attempt, as interrupts will be disabled. When using A=0, Y=0 for OSARGS, the control block will be found in the second processor, despite what the Advanced User Guide suggests.

BUILD YOUR OWN GRAPHICS TABLET (Part 1)

by Ben Miller-Smith

The analogue port of the BBC micro (model B) provides an easy way of interfacing a simple graphics tablet to the machine, and this can then be used to draw and copy diagrams onto the screen, or input relevant coordinates into a program. This article describes how to build such a graphics system, the first part detailing the construction of the tablet itself.

A graphics tablet connected to the analogue port provides an ideal way of entering a variety of visual images into the micro for display on the screen, and storage on cassette or disc. Drawings can be entered freehand, or existing diagrams traced out as required. Several such graphics tablets are commercially available, but at a price (see review in BEEBUG Vol.2 No.8), whereas the number of components required is minimal and the home construction cost is very small.

MATERIALS REQUIRED

(a) A Chipboard or Contiboard base, approximately 24" x 18", with a smooth (e.g. Melamine) finish. Available in 8mm or 12mm thicknesses (either suitable) from any DIY store at approximately £1.50 for a 24" x 36" panel, which you will need to cut in half to obtain the correct size.

(b) One metre of 5mm x 20mm (approx.) wood strip, to make the arms. If such strips are not available from your DIY store as straight timber, look for hardwood mouldings that are generally available in a variety of cross-sections, including rectangular. Approximate cost is about 50p.

COMPONENTS REQUIRED

(a) Two 10k linear panel-mounting potentiometers with at least 1" shafts. Available from Tandy shops (Part No. 271-1715 at 79p each), or Maplin Supplies (Part No. FW02C 'Pot Lin 10k' at 43p each).

(b) Three metres of four (or more) core cable, reasonably flexible. Available from most shops selling telephones, or from Maplin as Part No. XR66W '4-wire Phone Cable' in any reasonable length at 21p per metre. You can use twin flex, or even single wire, if necessary, but it won't be so neat.

(c) One 15-way D plug (i.e. with pins) to fit the BBC analogue port, preferably with a covering shell and cable strain-relief fitting. Available from Maplin as Part No. BK58N 'D-Range 15 way Plug' at £1.45 (optional cover BK60Q 'D-Range 15 Way Cover' at £1.20), or many computer shops.

(d) Four rubber feet for the base-board. Available from Maplin as Part No. FW19V 'Feet Cab', at 8p for 4.

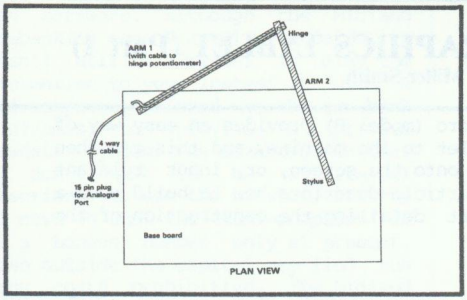
You will also need to have available a soldering iron and solder, drills in a range of sizes, some hand tools for wire cutting, and some adhesive.

Maplin components are available from their shops in Westcliffe, Hammersmith, Birmingham, Manchester and Southampton, or mail-order from their main office at Rayleigh in Essex (Tel. 0702-552911).

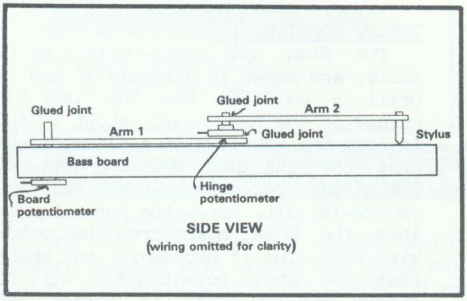
TABLET CONSTRUCTION

The plan and cross-section of the tablet are shown in diagrams 1 and 2. Drill a hole in the top left hand corner of the base board, about 1 inch in from the edges, with a diameter such that the threaded mounting collar of the first potentiometer is a tight (screw-in) fit. Screw the potentiometer into the base board from the bottom, with some glue if necessary to ensure that the whole potentiometer will not rotate when the shaft is turned. The shaft of the potentiometer will protrude through the board, ready to accept the first arm.

Cut a 13 inch length of the arm material and 0.5 inches from one end drill a hole to accept the board



potentiometer's shaft as a tight fit. Aim for the shaft/arm joint to be stiff enough so that in normal use the arm will always turn the shaft without slipping, but under abnormal stress (like dropping the tablet!) the joint will slip - however, if the joint is, or becomes, too slack a suitable adhesive can be used at the joint during final assembly. At the other end of the arm, glue the base of the second ('hinge') potentiometer to the top surface, such that the distance between the two shafts is 12 inches approximately. If the second potentiometer has not got a flat base (probably because the back end of the shaft protrudes slightly), drill a clearance hole in the arm so that the potentiometer can sit flat. The four rubber feet, referred to in the list of components, or just suitable blocks of wood, can be fitted to the underside of the four corners to provide a firm and level base.



The second arm is prepared with a similar tight-fitting shaft hole Ø.5 inches from one end, and a wood screw (or bolt) as a pointer at the other. Drill the hole for the pointer so that the effective length of the two arms is the same (i.e. arm 1 shaft centre to shaft centre equals arm 2 shaft centre to pointer). If it's not quite right, don't worry - corrections can be made in the software. Aim for the height of the pointer to be such that the second arm will be supported horizontally by its mounting on the hinge potentiometer shaft at one end, and the pointer resting on the base board at the other. Do not have the pointer tip too sharp or you may tear things being copied.

Complete the mechanical assembly. Press the first arm down the board potentiometer's shaft until it is flat on the base board, but can turn freely. Ensure that the potentiometer will not meet its 'end-stops' during normal use (rotate the shaft with respect to the arm if required). Glue the arm/shaft joint if necessary. Similarly, mount the second arm on the hinge potentiometer's shaft, ensuring a tight fit (glue if required), and free movement of the shaft between end stops. The pointer on the end of the arm linkage should now be able to reach almost all points on the base board, and all the action should be smooth with no jerkiness, sloppiness or slipping between the arms and the potentiometer shafts. This is most important and should be tested before final gluing.

ELECTRICAL CONNECTIONS

Four wires connect the Graphics Tablet to the computer Analogue port. VREF (pin 14) and ANALOGUE GROUND (pin 8) are connected to the two potentiometers in parallel, on the outer potentiometer tags. The wiper (centre) connection of the first (base-board) potentiometer is connected to CHANNEL 1 (pin 15), and the second (hinge) potentiometer wiper to CHANNEL 2 (pin 7).

In practice, carefully solder three wires to the hinge potentiometer (using multicore solder), twist them together, and run them back along arm 1 (attaching them with blobs of glue if necessary) and over the edge of the board (or down through an extra hole) to reach the first potentiometer, making the same two connections to the outer tags of this potentiometer, and connecting a fourth wire to its wiper. At the other end of at least 3 feet of cable solder the wires to the Analogue

port plug according to the table below, preferably using a soldering iron with a fine tip, and some sleeving over the joints to prevent accidental short-circuits.

D-Plug	Connection	Signal Name
Pin 7	Hinge potentiometer wiper (centre tag)	Channel 2
Pin 8	Both potentiometers 'bottom' tag	Analogue Ground
Pin 14	Both potentiometers 'top' tag	Vref
Pin 15	Board potentiometer wiper (centre tag)	Channel 1

Test the connections as follows. Clockwise rotation of either arm should give an increasing analogue voltage input to the computer. Run the following little program and observe the displayed voltages as the arms are moved (ignore any jitter in the least significant digits).

```
10 MODE 7
20 REPEAT
30 PRINT TAB(0,5);SPC(40)
40 PRINT TAB(0,5);ADVAL(1)
50 PRINT TAB(20,5);ADVAL(2)
60 TIME=0:REPEAT UNTIL TIME>50
70 UNTIL FALSE
```

If either potentiometer is working in reverse, interchange the connections on its 'top' and 'bottom' tags (i.e. the two outer tags). If both are reversed it is probably easier to interchange the connections on pins 8 and 14 of the Analogue input plug.

This completes the detailed construction of the graphics tablet. Next month we will conclude this project by looking at how to calibrate the graphics tablet and at a program which provides a set of basic drawing routines to use with your new add-on.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINT

FASTER BASIC - P.J. Vincent

When performing certain operations in Basic, some programming techniques are more efficient than others (timewise). Generally speaking, REPEAT-UNTIL is faster than IF-THEN-GOTO, but FOR-NEXT is faster than REPEAT-UNTIL. A GOSUB is slightly faster than a procedure, contrary to what the User Guide suggests, and it is quicker to use global variables instead of local variables, again different to what the User Guide says. Turning off the ADC conversions though has very little effect upon timings.

CASSETTE USAGE OF WORDWISE - F. Duerden

If tape users of Wordwise want to produce several similar letters with just the top section different, then this can be achieved by means of the GF"name" command. If this is included in the DH section, and a number of files, all with the same name but different contents are present on tape, then Wordwise will produce a number of copies of the current piece of text, except that the headers will be different.

HOBBIT AND GDUMP - Tim Pearkes

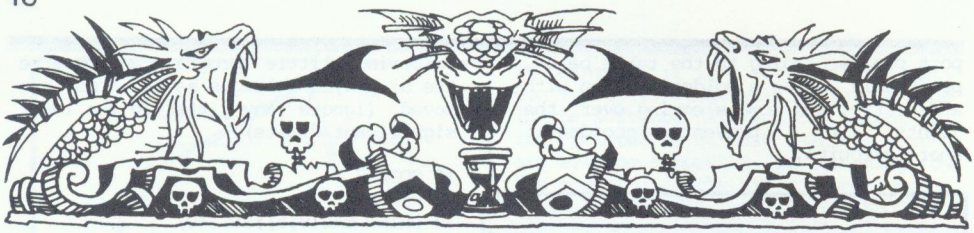
Users of the Hobbit system may be interested to know that GDUMP will not function correctly with the Hobbit enabled. Disable it, and GDUMP then functions correctly. Several ROMs are apparently affected by the Hobbit system in this way.

TORCH CORRUPTION - John Satchell & John Sowden

When chaining between BBC Basic programs, or hitting Escape while running such a program, the Torch Z80 (version v71) corrupts location &80.

DOCTOR SOFT MODS - Bjoern Floetten

There is an extra hidden command in Doctor Soft's 747 Flight Simulator; pressing 'T' will move the plane forward by 1 nautical mile.



ADVENTURE GAMES

Dungeon Master 'Mitch' invites you to join him in his adventure world

Welcome to the Dungeon mortal. Let me clear a space on this coffin lid for you to sit down. Ignore the dragon - he has already eaten so you are safe for a while. Well have you finally tired of zapping the Invaders? Perhaps now's the time to accept you're never going to get to the fourth screen on Killer Gorilla or see the sixth screen of Swoop. Never mind, there are other worlds to conquer where your brain can compete on equal terms with those nimble-fingered twelve year olds who can clear a Snapper screen while eating a hamburger with their other hand.

For those gentle readers who have never entered the world of Adventures let me briefly explain tne delights. Before the coming of graphic screens and sound generators, programmers on mainframe computers were restricted to playing Noughts and Crosses and Hangman. Luckily, some bright spark realised that the text handling power of the computer, coupled with the imagination of a fantasy author, could produce a brilliant new dimension to adventure stories. Unlike a book, where the reader is unable to help or hinder the progress of the hero, the computerised adventure story allows the reader to decide the next move (e.g. shall he enter that dark doorway or not?). A wrong decision could result in an early and bloody demise. Well written adventures are an exciting escape into a world of magic and fantasy which fascinate both young and old. Too many adventure writers forget that the text must be skilfully used to draw the player into his world to amuse, intrigue and stimulate the imagination - boring text always means a boring game.

Bearing in mind that well written text is the first essential of an adventure, the use of sound and graphics is creeping in. The BBC micro is restricted in its memory size compared with others and so far the introduction of graphics has normally been at the expense of the descriptive text. However text compression techniques are making it possible to include quite large graphic sections in some of the new games. Potential authors among you should remember that graphics should be treated with caution however, as pretty pictures are no substitute for exciting prose. Teletext graphics are becoming more popular as their use does not require the use of a graphics mode with its subsequent loss of memory. The latest game from Epic Software, 'Wheel of Fortune', has managed to combine a full sized game with a half screen teletext picture at every location without losing much in its text. However Bug Byte's 'Twin Kingdom Valley' seems to have placed its eggs firmly in its graphics mode basket with full screen graphics combined with limited lack-lustre text which does little for the imagination. Other games such as 'Vampire Castle' from Micrograf use small novelty teletext effects to introduce some spice into a mainly text game. I note that Level 9 computing intend to introduce graphics into their future games. [The grapevine says that these will start to appear soon. - Ed.]

So far Acornsoft have resisted the move towards graphics. Their traditional text-only games such as 'Castle of Riddles' and 'Philosophers Quest' have instead relied on amusing descriptions to create the scene. While this is perfectly acceptable (and preferred by some - do you need picture story books?), it's a pity they didn't use lower case lettering which is easier on the eye. This apart they can still hold their own against some of the glossy competition which uses coloured text for everything making it

impossible to read on some monitors and black and white televisions. Remember - rubbish written in pretty colours is still rubbish.

The Grand Wizard of Acornsoft has just sent me his latest offering called 'Gateway to Karos'. This adventure is another in the same Acornsoft mould - no graphics or sound. The format has, however, been changed slightly to include lower case letters and some coloured text. The location of the game is the mythical land of Karos where your task is to find the Talisman of Khoronz plus any other jewelled bauble you happen to stumble upon and then return safely through the magical gateway. As with earlier games Karos is populated with demons and magic users, oil lamps, screwdrivers and ruby brooches. There are 250 locations to explore and 65 objects to manipulate which should keep most glory hunters busy for quite some time. The inhabitants may be questioned by you to obtain clues and your questions, plus other commands, may be complex statements instead of the usual two word instructions such as 'FILL BOTTLE' (yes the bottle is back again!). In addition to falling rocks, and man eating wolves your condition gradually deteriorates unless you eat and drink occasionally. As usual that damned lamp starts to flicker just as you need it most so don't forget to turn it off when you don't need it.

I have enjoyed all Acornsoft's adventures and this is no exception as it sticks closely to their winning formula of lots of relatively easy problems which you know you will solve if you persist. Some other companies' adventures are too devious for their own good, and they have failed in the art of setting the problems at the level designed to satisfy master and novice alike. Karos is like a tin of Heinz beans; it is a safe buy, and you know before you hand over your money what you are getting, which although may not be a brave new leap forward, will not leave you feeling cheated as with some over-hyped games. I'm not sure that the trend towards complex command analysers in adventures is worth the trouble for Karos, like others, misinterprets some long commands. I found I quickly returned to two word commands as it is usually quicker and safer. In addition I found that posing clever questions to the games inhabitants was a waste of time, as simply typing "ASK" seems to get the same reply - which incidentally is usually a baffling cryptic quote. The game certainly has hours of fun packed into it and so far I've found no mazes - thank goodness! I'd like to strangle the programmer who first introduced these as a feature.

Included with the game is an envelope containing answers to the various problems. This seems a better solution to the earlier idea of the postcard in Philosophers Quest which could be used to request clues from Acornsoft. I bet they didn't know what they had let themselves in for with that. So far I have resisted the temptation to open the envelope, but I hope they have laid it out better than Level 9 have done with their clue sheet. It appears to be impossible there to find only the answer to your problem without seeing the answers to everything else. As a final word on Acornsoft, whatever happened to 'Kingdom of Hamil' which they released last year? I've seen precious little of it in the computer press and I note that its name is excluded from the 'Also Available' list of titles on Karos' rear cover. Perhaps it has disappeared up one of its own twisty mazes or got stuck in the mud below the wishing well. [We actually have a copy of this game, but don't have the time to solve it! - Ed.]

For those heroes and heroines who have already crawled into some dark opening inside their micro and are currently beating their brains out against some unsurmountable puzzle, why not scribe the details on a piece of old parchment, mark it ADVENTURE DUNGEON, keeping it apart from any other queries you have and drop it into the dark opening of a pillar box to me here, at BEEBUG. I'll welcome any problems , hints or suggestions, so if you have any spare spells let me know.

P.S. Write in ink not blood, it scares the postman and agitates the dragon!

DOMESTIC ACCOUNTS EXTENDED

by David Fell

In the BEEBUG Vol.2 No.10 (April 1984) issue, we published a program called Home Accounts - Annual Budgets (or DOMAC for short), and this attracted quite a lot of feedback from members. One question that was frequently raised was that of dumping the screen to a printer. This update to the DOMAC program provides a couple of extensions to this program, including a quick universal screen dump.

Adding the new sections to the program is a fairly easy task. Load in your original version (which should not have been renumbered), and type in the new lines from the section that you require; for example, if all you want is the screen dump, type in only the lines from the screen dump section. Note that some of the lines in the original will be overwritten - this is intentional.

SCREEN DUMP

This is the main update to the original program, and consists of a few minor alterations to the main program, plus a section of code at the end of the program that pokes the screen dump to memory. There is an error detection check built into this, which will stop the program before any damage is done if you have made an error when typing in the DATA statements. The screen dump can be called at any time from within the program by holding down the Shift key and pressing Escape. The program will make a short tone, dump out the current screen to the printer, and then return to the main menu.

TUBE * COMMANDS

The original DOMAC program was 6502 Tube compatible with the possible exception of the * commands section, which tended to overwrite the program with the * command. Either set PAGE to &D00 or greater if running in the Tube, or make the short alterations listed below. Basic I users should not add the last two lines of the * command section. An asterisk (*) typed in at the menu now also calls the OS command section.

ACCURACY

Due to the way in which the program stored its numbers, there were occasional rounding errors. Whilst the degree of error here is probably less than that in estimating the figures entered, it is still irritating. The correction is quite simple; change the factor of 100 to a factor of 1000 in the following lines where 100 is used for division or multiplication: 1750, 1890, 1950, 2080, 3040, 3090, 3120, 3150, 3910, 3990. Those of you with Toolkit may like to use the search and replace option to do this.

AND FINALLY...

Thankyou to all the members who have written in with comments and suggestions on this program, and also to TPL for providing assistance to members who wanted screen dumps before I wrote the little routine here.

Note - This month's magazine cassette / disc contains the complete updated DOMAC program.

SCREEN DUMP SECTION

```
  110 MODE4:HIMEM=&5700:PROCinit:PROCti
tle
  130 MODE4:HIMEM=&5700:PROCcolours:PRO
Ccolour(1)
 2105 PROCreadassem
 2280 DEF PROCerror1 SOUND1,-15,10,10
 2285 IF INKEY-1 CALL&5700:ENDPROC
 2330 DEF PROCerror2 SOUND1,-15,10,10
 2335 IF INKEY-1 CALL&5700:E%=FALSE:END
PROC
 5000 DATA 202657203957EEA8
 5010 DATA 57A928CDA857D0F3
 5020 DATA A9008DA85720E7FF
 5030 DATA EEA957A920CDA957
 5040 DATA D0E120985760A900
 5050 DATA 8DA8578DA957A91A
 5060 DATA 20EEFFA90220EEFF
 5070 DATA 60A91F20EEFFADA8
 5080 DATA 5720EEFFADA95720
 5090 DATA EEFF206E57A98720
```

```
5100 DATA F4FFE000D00E2083
5110 DATA 57A98720F4FFE000
5120 DATA D002A920A90120EE
5130 DATA FF8A20EEFF60A911
5140 DATA 20EEFFA90720EEFF
5150 DATA A91120EEFFA98020
5160 DATA EEFF60A91120EEFF
5170 DATA A98720EEFFA91120
5180 DATA EEFFA90020EEFF60
5190 DATA A90120EEFFA90C20
5200 DATA EEFFA90320EEFF60
5210 DATA EAEA000000000000
5230 DEF PROCreadassem
5240 RESTORE 5000
5250 P%=&5700:T%=0
5260 FORI%=0TO21
5270 READ A$
5280 FORJ%=0TO7
5290 V%=EVAL("&"+(MID$(A$,J%*2+1,2)))
5300 ?(P%+J%+I%*8)=V%
5310 T%=T%+V%
5320 NEXT,
5330 IF T%<>23519 PRINT"Checksum error
.":END
5340 ENDPROC

* COMMANDS SECTION

 190 R=GET:IF R<>42 R=R AND &DF
 260 IF R=71 OR R=42 PROCos
3220 INPUT"*"OS$:OSCLI OS$
3230 VDU14

REM SECTION

  10 REM Program DOMESTIC ACCOUNTS II
  30 REM VERSION B2.0M
  40 REM BEEBUG
```

NEW EDITION OF BIRNBAUM'S BOOK

Reviewed by Mike Williams

Assembly Language Programming for the BBC Microcomputer (2nd edition) by Ian Birnbaum, Macmillan, price £8.95.

Back in 1982 when the Beeb was still a mysterious adventure for most (that's if you were lucky enough to be able to get hold of one) there appeared a book that immediately became a best seller in its field. That book on assembly language programming has now progressed to a new second addition. So what has changed since we reviewed this book in BEEBUG Vol.1 No.9?

Certain changes stand out straight away, the cover is brown, not green, the whole book (apart from the program listings) has been typeset and (hurrah, hurrah) the book now has a full index. It is also marginally longer (by 12 pages) yet looks slimmer because of the thinner paper used. The price remains unchanged from the original.

The effect of typesetting has been to lose the somewhat distinctive appearance of the first edition and this also seems to have resulted in a more frequent occurence of diagrams that are separated from the text that refers to them. On the other hand the program listings are now much clearer, and advantage has been taken of the LISTO command to produce better structured and more readable listings.

Essentially the content of the book is as before, with a progressive introduction to assembly language programming based on equivalent Basic concepts, and backed up by plenty of short example programs. I personally remain unconvinced of the use of Basic as an introduction to assembler as it propogates a high level language view of low level programming, but it is clearly an approach which has proved popular and succesful.

The chapters on indexed and indirect addressing are indeed models of excellence, and the text has been revised to take full advantage of the EQUS and EQUB commands of Basic II. The chapter on subroutines and interrupts has been extensively re-written to cover the extra features of O.S.1.2 such as events, and extra example programs have been included to illustrate the new material.

Solutions to all the exercises set in the book are provided as before and all the programs are now available on a single cassette, not two, for £9.00.

Without doubt this book must remain one of the best introductions to assembly language programming for the BBC micro that has been published.

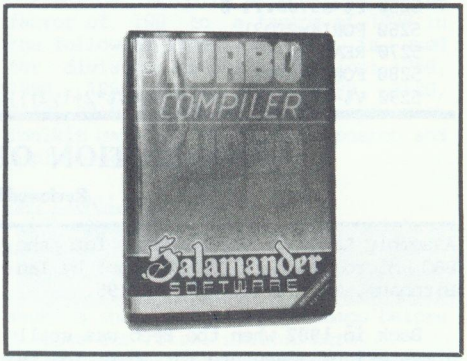
TWO BASIC COMPILERS

Reviewed by Eric Glover

Although Basic is easy to learn and convenient to use, machine code is often preferred for its speed and compactness (think of all those machine code games). In consequence, a compiler which could convert Basic to machine code would seem to be high on everyone's priority list. Eric Glover looks at two compilers for Basic that have recently been released. Could one of these be the solution to your programming problems?

```
Product  : Turbo Compiler
Supplier : Salamander Software,
           17 Norfolk Road, Brighton,
           Sussex, BN1 3AA
Machine  : BBC models A & B
Price    : £9.95 tape inc.VAT +50p p&p

Product  : BBC Basic Compiler
Supplier : ACK Data, 21 Salcombe Drive,
           Redhill, Nottingham, NG5 8JF
Machine  : BBC 32K
Price    : £14.95 tape, £19.95 disc
           inc.VAT
```



INTRODUCTION

When a program in BBC Basic is run, the Basic interpreter, built into the micro, interprets each instruction and carries out its function. Much time is spent on interpretation. For example, every instruction in a loop has to be interpreted each time it is encountered. Basic cannot 'remember' its meaning from the previous time.

As a result, interpreted Basic is slow compared to machine code, but programs written in Basic can be made to run much faster. If a Basic program is translated (or compiled) into machine code prior to execution, then the resulting program will run nearly as fast as one written directly in machine code, but with all the convenience of programming in Basic.

Once converted to machine code, the program can be saved in this format for execution at any future time using the command *RUN. The machine code version will probably also take up less memory space than the original Basic program. In addition, programs in machine code are very difficult to read, and this format provides a degree of program protection not possible with Basic.

SOME TERMS EXPLAINED

When using a compiler, the original Basic program is called the 'source' program, and the compiled machine code is known as 'object' code. With both the compilers reviewed here, the source code is prepared by typing it as usual into memory from the keyboard. Although convenient, this limits the size of the Basic program that can be compiled (or the size, and hence capability, of the compiler) as both must share memory together. Not only that, but memory space must also be found for the object code as well. This is not essential - most professional compilers take the source code from disc, and in turn send the object code back out to disc, thus allowing the compiler to occupy most of the available memory, and the Basic program to be very much larger.

THE LIMITATIONS

Ideally you should be able to compile any Basic program, but in reality it is impossible to fit a full compiler into memory, together with

both source and object code. Consequently compilers tend to accept only a subset of BBC Basic. Amongst the features often excluded are string variables, arrays, and real numbers. The two compilers reviewed here are no exception to this trend.

SALAMANDER SOFTWARE - TURBO COMPILER

This compiler is well packaged with a good clear manual. A tape version and a disc version of the compiler are included on the cassette as well as a demonstration graphics program. The manual provides full instructions for the use of the compiler and its transfer to disc if required. It is written in compact machine code (only 2K!) and sits below PAGE. The default addresses for the start of the source program and object program are &1500 (&2100 disc) and &2200 (&2800 disc) respectively, but they may be easily changed. Compilation is achieved by the *TURBO command, but there is no assembler listing. If an error is detected, the faulty line will be indicated (in hexadecimal!). If compilation is successful, the object program may be CALLed at the appropriate address, and also saved using *SAVE.

The facilities offered by the Turbo compiler are barely adequate to justify the term "compiler", and in many respects Turbo is little more than a high level assembler. The major constraints on the Basic source code include:

1. Except for line numbers and memory addresses, only positive hexadecimal single byte integers are allowed in the range 0 to 255, negative numbers being handled as complements.

2. Only the integer variables A% - Z% may be used.

3. Each Basic line may only contain one statement.

4. String variables are not allowed, but the string indirection operator '$' is implemented in the usual Acorn fashion (as is '?' but not '!').

The following Basic keywords are implemented as normal or with only minor restrictions: CALL (returns 6502 register values in A%, X%, Y%), CLG, CLS, AND, EOR, OR, END, GET, GOSUB, IF, INKEY, LET, MID$, MODE, REM, RETURN, SOUND and VDU.

Basic keywords implemented with quite major restrictions are: CHR$ (may only be used in PRINT statements), FOR-NEXT (must not be nested), PRINT (only strings may be PRINTed), and THEN (only a line number may follow).

The effect of several other Basic statements (ADVAL, ASC, BGET#, BPUT#, CLOSE#, DRAW, ENVELOPE, INPUT, LEN, MOVE, OPENIN, OPENOUT, PLOT, POINT, RND, and TIME) may be obtained by the use of the VDU statement, the indirection operator '?', or OSBYTE/OSWORD calls. Division is carried out by repeated subtraction.

As you may realise, a full list of Basic statements not supported by the Turbo Compiler would be rather long. These drawbacks may be less serious if it is regarded as a tool for compiling machine code subroutines, which may then be CALLed from interpreted Basic. Providing the user has a knowledge of O.S. routines (as given in the BBC User Guide) and a degree of perseverance it could prove useful in this respect. But writing a large program wholly for compilation by Turbo could well prove to be frustrating if not impossible with so many restrictions.

ACK DATA - BASIC COMPILER

Only the disc version of this compiler is reviewed here, but the tape version would seem to be identical in function. The compiler itself is written in Basic and occupies about 9K. Again you type in or load your source program as for a normal Basic program. The compilation is started by pressing function key f1. If all goes well, an assembler listing of the compiled code is produced, which may then be run by CALLing &1902 (&E02 for cassette systems). If, however, there is an error in the source program, then an error message usually appears indicating the offending line (if you are lucky). Occasionally, the compiler will go into an endless loop until Escape is pressed, whereupon it will confess that it can't compile the program.

The source program starts at &3900 and the compiler itself resides at &5300. These locations appear to be fixed. A source program of up to 6.5K (!) can be compiled with the following restrictions:

1. Arithmetic is two byte integer only (0 to 65535 or -32768 to 32767). Numbers may be in decimal or hexadecimal.

2. Only the variables A% - Z% may be used.

3. There are no string variables or string handling facilities.

4. Nested FOR-NEXT and REPEAT-UNTIL loops are not permitted.

The following BBC Basic keywords and operators are "supported": ADVAL, AND, CALL, CLEAR, CLG, CLS, COLOUR, DEF, DIV, DRAW, END, ENDPROC, ENVELOPE, EOR, FOR, GCOL, GET, GOSUB, GOTO, HIMEM, IF, INKEY, INPUT, LET, LOMEM, MOD, MODE, MOVE, NEXT, OR, PAGE, PLOT, POINT, POS, PRINT, PROC, REM, REPEAT, RETURN, RND, SOUND, SPC, SQR, STEP, TAB, THEN, TO, TOP, UNTIL, VDU, VPOS, '+', '-', '*', '/', '&', and '?'.

I say "supported" because although one is told that "the purpose and effect of these words is the same as described in the BBC User Guide", this is not quite true. For example, I could not call procedures recursively or use strings in INPUT statements. Neither of these restrictions was mentioned. Generally, however, the compiler worked well provided that spaces were avoided in the source program. Although more sophisticated Basic programs could be compiled with this compiler than with the previous one, its main use, I suggest, would be the same, the development of compiled machine code routines to be CALLed from within a Basic program.

CONCLUSIONS

Both compilers were robust, easy to use and did not overwrite the source program, although the ACK compiler annoyingly renumbered it 0, 1, 2...etc. Since the main attraction of using a compiler is the speed of the compiled code (the speed of compilation is irrelevant as it only has to be compiled once) it is important to say that both performed well in this respect. The ACK compiler was able to compile the first five PCW benchmarks which then executed roughly 15 times faster than with Acorn's Basic interpreter (itself very fast). Owing to its limited arithmetic facilities, Salamander's compiler could not compile these benchmarks except in a modified form, but the results were just as quick in general.

Overall, the ACK compiler is a more comprehensive offering than that of Salamander Software, and it produced more sophisticated machine code programs. For example, when printing a string it would set up a loop and access each character by indirect indexed addressing, whereas the Turbo compiler would simply load the accumulator with each character directly until it reached the end. The difference in capabilities of the two is reflected in their prices, but both exemplify the soft options of compiler design (rudimentary parser, no symbol table etc.) and are rather disappointing. However, given a little ingenuity and some expertise, both could be put to some good effect, e.g. shifting aliens. Despite their relative cheapness, it is difficult to give other than lukewarm approval to compilers that impose so many limitations upon the Basic programmer. As such, we look forward with anticipation to the future release of Computer Concepts' new Basic compiler.

POINTS ARISING

Z80 SECOND PROCESSOR REVIEW (BEEBUG Vol.3 No.4)

We have been asked to point out that M-Tec Basic, which was referred to in the comparison with the equivalent Torch ZEP100 system, is now bundled in with the Perfect software at no extra charge. The price of £126.50 only applies when M-Tec Basic is purchased as a separate software item.

IMPROVED FUNCTION KEY LABELS (16K)

by Graham Crow

In BEEBUG Vol.2 No.9 we published a program to print out function key labels on an Epson printer. Graham Crow has now improved this program to produce even neater function key labels on an Epson FX-80.

By using superscript mode for the text, it is possible to achieve a whole line of bold heading and still have 9 lines of text. Using reverse feed, the border is printed in single strike mode, while the text is in double strike. Several examples of excellent keystrips produced by the program have been selected to illustrate this article.

Type the program in and save it on cassette or disc. The program already contains data to produce a keystrip for ASTAAD2 published in BEEBUG Vol.2 No.9. If you run the program, then this will be the keystrip printed.

You can easily add further lines of data for other keystrips of your choice and simply change the value after the RESTORE in line 130 to point to the start of the data for the keystrip you want to print. This allows you to maintain one program that is capable of printing any one of several keystrips.

The program also has double broken lines above and below the function key label. Cutting between these will ensure that the keystrips fit your Beeb exactly.

For each keystrip you will need 9 lines of data, the first to contain the keystrip title, and the remainder will each require 10 data items to be printed on that line, as shown in the example provided.

Before you print a keystrip with this program, you will need to make sure that the buffer on your printer is switched off because user defined characters on the printer use the same memory space as the buffer. This can be done by removing the cover for the DIP switches and making sure that switch 1-4 (switch number four in the row of eight switches) is in the off position. Then you can run the program and see the results for yourself. Remember though, to turn the buffer back on when you have finished.

```
WORDWISE
SHFT +CTRL | HASH SIGN | UNDERLINE ON | UNDERLINE OFF | EMPHASISED ON | EMPHASISED OFF | CONDENSED ON | CONDENSED OFF | ENLARGED ON | ENLARGED OFF | STANDARD PAGE
           | 0C27,82,0, 35,27,82,3 | 0C27,45,1 | 0C27,45,0 | 0C27,69 | 0C27,70 | 0C15 | 0C18 | 0C27,87,1 | 0C27,87,0 | LH5LL76PL72 TS2B52EPJO
           | INSERT/ OVERWRITE | EMBEDDED COMMAND START | EMBEDDED COMMAND END | MARKER | MOVE CURSOR TO CHAR | WORD COUNT TO CHAR | DELETE UP TO CHAR | DELETE MARKED SECTION | MOVE MARKED SECTION | COPY MARKED SECTION
```

```
  10 REM PROGRAM CRIB
  20 REM VERSION B0.1
  30 REM AUTHOR  G.M.Crow
  40 REM BEEBUG  NOVEMBER 1984
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60:
 100 MODE 7:ON ERROR GOTO 210
 110 PROCinit
 120 PROCredefine("*")
 130 RESTORE 1880:REM change this line
to change CRIBS
 140 PROCheading
 150 PROCborder
 160 PROCreverse(176)
 170 PROCtext
 180 VDU1,27,1,64,3:REM printer off
 190 END
 200:
 210 REPORT:PRINT" at line ";ERL:END
 220:
1000 DEFPROCinit
1010 superscript=TRUE:condensed=FALSE
1020 C$=STRING$(7," ")
1030 VDU2,1,27,1,64:REM printer on
1040 REM *FX6
```

→

```
 1050 REM enable linefeeds (delete if a
lready set by DIP switch)
 1060 PROCspacing(24):REM 24/216" lines
 1070 ENDPROC
 1080:
 1090 DEFPROCheading
 1100 cutline$=STRING$(132,"=")
 1110 PROCfont(4):REM condensed
 1120 PRINT'cutline$;
 1130 PROCfont(49):REM enlarged/emphasi
sed/elite
 1140 READ heading$
 1150 PRINT SPC(4);heading$
 1160 ENDPROC
 1170:
 1180 DEFPROCborder
 1190 t$="*********** ********** "
 1200 m$="*          *          "
 1210 T$=C$+STRING$(5,t$)
 1220 M$=STRING$(6," ")+STRING$(5,m$)+"
*"
 1230 PROCspacing(18):REM 18/216" lines
 1240 PROCfont(4):REM condensed
 1250 PROConeway:PRINT T$
 1260 FOR J=1 TO 9:PROConeway:PRINT M$:
NEXT
 1270 PROConeway:PRINT T$
 1280 ENDPROC
 1290:
 1300 DEFPROCtext
 1310 FOR J%=0 TO 8:FOR I%=0 TO 10
 1320 IF condensed THEN PROCfont(20):RE
M condensed/double-strike
 1330 IF superscript THEN VDU1,27,1,83,
1,0
 1340 READ text$
 1350 IF text$="line" THEN I%=10:printl
ine$=T$:GOTO 1430
 1360 IF I%=0 THEN printline$=LEFT$(tex
t$+C$,6)+" ":GOTO 1430
 1370 IF I%/2=INT(I%/2) THEN text$=LEFT
$(text$,12) ELSE text$=LEFT$(text$,11)
 1380 REM 12 chrs(evens),11 chrs(odds)
 1390 text$=STRING$(6-INT(LEN(text$)/2)
,CHR$32)+text$+C$:REM centre string
 1400 IF I%/2=INT(I%/2) THEN text$=LEFT
$(text$,12) ELSE text$=LEFT$(text$,11)
 1410 text$=text$+" "
 1420 printline$=printline$+text$
 1430 NEXT
 1440 PROConeway:PRINTprintline$;
 1450 NEXT:PRINT''
 1460 VDU1,27,1,84:REM superscript off
 1470 PROCfont(4):REM condensed
 1480 PRINT cutline$
 1490 ENDPROC
 1500:
 1510 DEFPROCspacing(n)
 1520 VDU1,27,1,51,1,n
 1530 REM lines are n/216" apart
 1540 ENDPROC
 1550:
 1560 DEFPROCfont(n)
 1570 VDU 1,27,1,33,1,n
 1580 REM n specifies mix of type modes
 1590 ENDPROC
 1600:
 1610 DEFPROConeway
 1620 VDU1,27,1,60
 1630 REM prints left to right only for
 best alignment
 1640 REM acts only on current line
 1650 ENDPROC
 1660:
 1670 DEFPROCreverse(n)
 1680 VDU1,13,1,27,1,106,1,n
 1690 REM reverse feeds paper by n/216"
 1700 ENDPROC
 1710:
 1720 DEFPROCredefine(char$)
 1730 REM Printer manual sections 3-36
onward
 1740 char=ASC(char$)
 1750 REM copy ROM characters into RAM
Character Generator
 1760 VDU1,27,1,58,1,0,1,0,1,0
 1770 REM specify character
 1780 VDU1,27,1,38,1,0,1,char,1,char
 1790 VDU1,139:REM attribute 'a'
 1800 REM numbers for new shape
 1810 VDU1,0,1,0,1,32,1,0,1,80,1,0,1,80
,1,0,1,32,1,0,1,0
 1820 REM select download character set
 1830 VDU1,27,1,37,1,1,1,0
 1840 ENDPROC
 1850:
 1860 REM*****************************
*********************************
 1870:
 1880 DATA "ASTAAD2   BEEBUG   March 19
84"
 1890 DATA,,,,,,,,,
 1900 DATA plus,SCREEN,SCREEN,SCREEN,SO
FT CHARS,INFILL,DUMMY,REVERSE,,,
 1910 DATA SHIFT,SAVE,LOAD,DUMP,ON/OFF,
ON/OFF,ROUTINE,COLOURS,,,
 1920 DATA,,,,,,,,,
 1930 DATA line
 1940 DATA,,,,,,,,,
 1950 DATA,ASTAAD,DRAW,DRAW,REPEAT,MOVE
,DRAW,LINE,DELETE,DELETE,DRAW
 1960 DATA,TEXT,ARROW,POLYGON,POLYGON,,
,,LINE,AREA,CIRCLE
 1970 DATA,,,,,,,,,
 1980:
 1990 REM*****************************
*********************************
```

BEGINNERS THIS WAY

DEBUGGING PROGRAMS (Part 1)

by Ben Miller-Smith

This month Ben Miller-Smith presents the first of two articles on coping with errors in programs. This can be a difficult task for the beginner, despite the excellent error reporting on the BBC micro. This month's article concentrates on some of the problems that may arise from errors introduced when typing in a program.

1. INTRODUCTION

The BBC computer and its Basic interpreter provide quite comprehensive and powerful error detection and reporting routines which are a big help in debugging programs, especially as a program of any complexity is very unlikely to work quite as expected first time. This may create a whole series of error messages or strange results which can be quite difficult to sort out. If you have copied programs out of magazines or books you will probably have made some typing errors and found some difficulty in getting the program to run properly. Various different troubleshooting techniques may be necessary depending on the nature of the apparent errors, but before we look at them in detail it is well to understand the error handling facilities on the BBC micro, as these can affect debugging.

2. ERROR HANDLING ON THE BBC

In normal operation the BBC's Basic interpreter will report errors as a descriptive message, with an associated program line number if possible - perhaps the most familiar is the result of pressing the Escape key while a program is running, which usually results in a message of the form:

```
Escape at line 210
```

indicating that the program just happened to be executing line number 210 at the time the Escape key was pressed. Similarly, if line number 410, say, had been incorrectly entered as:

```
410 Y = SIN(X
```

then the error message should be:

```
Missing ) at line 410
```

which is a pretty clear statement of the problem. However, complications in debugging a program may arise if the error handling mechanism of the BBC computer has been modified with an 'ON ERROR GOTO...' or similar statement, usually near the start of a program, or in an initialisation procedure. In this case the reporting of the action following any error is the responsibility of the programmer, and in some cases this leaves much to be desired [Hopefully not with the programs we publish in BEEBUG - Ed.]. Using the ON ERROR type of statement is one way of disabling the Escape key, since depression of the key creates, as far as the computer is concerned, error number 17 ('Escape'). The programmer might wish to cause the program to start again when the Escape key is pressed, so a statement of the form:

```
10 ON ERROR RUN
```

is likely to be included near the start of the program. This is all very well, but if you have copied a magazine program and unfortunately introduced some errors, then when the program comes across the first error it will restart, find the error again, restart, and so on for ever. The simplest thing to do, having pressed Break and typed OLD (carefully! Oh L D, not zero L D), is to disable the ON ERROR statement temporarily by putting a REM in front of it thus:

```
10 REM ON ERROR RUN
```

which will enable the normal error reporting system to operate - at least you now have a chance of finding and correcting any errors. When you think

the program is operating correctly you can always restore the ON ERROR statement to its original form if necessary. Always look for this type of problem if you have copied or modified someone else's program, and it either appears not to work at all, or seems to run for a time and then unexpectedly restarts.

The BBC computer keeps an internal record of the last error encountered, and the line number associated with it, if any. This information can be recalled at any time by examining the values of two pseudo-variables called ERR (the last-error number) and ERL (the line number). This facility is sometimes used in programs to give a 'tidy' error detection and reporting routine, especially where a program has put the computer into an unusual state (e.g. small text window, different colours, a large character mode, etc., etc.) The additional command 'REPORT' will cause the error message associated with a given ERR number to be printed, so a nice way to handle errors, including terminating a program with the Escape key (error number 17) is to include the statement:

```
9000 IF ERR<>17 THEN ON ERROR OFF:
MODE 7: REPORT: PRINT "at line"ERL: END
9010 REM Tidy up following Escape
9020 VDU 15: VDU 20: VDU 26: REM Res
tore defaults, etc.
9030 END
```

At or near the start of the program this error-handling routine would be switched on with a statement of the form:

```
10 ON ERROR GOTO 9000
```

Note that this method of error interception gives the programmer complete control over the system following any error and not just Escape, as in the example above. It is thus possible to trap all run time errors such as 'Division by Zero' or trying to extract 'Negative Square Roots', to do something about it, and then return to the main stream of the program at some suitable re-entry point - the availability of such restart points will markedly depend on how well the program is structured.

3. COMMON ERRORS

If the computer comes across an instruction that it cannot understand it will stop running the program and do its best to report the type of error encountered, and the line number currently being executed in the program. The commonest cause is a typing error, or your misunderstanding of the format required for a particular type of instruction. For example, a Basic line reading:

```
320 PRONT "Enter a new position"
```

will cause the error message:

```
Mistake at line 320
```

since PRONT is not a valid Basic word, and 'Mistake' is the computer's way of saying that it cannot make any sense of the given line.

Sometimes the error reporting system has a problem in identifying the exact cause of the error, as in the line:

```
400 newheight = time ** droprate
```

which will result in the message 'No such variable at line 400' because the second * is interpreted as a variable name rather than as an extra arithmetic symbol. The 'No such variable' error is probably the most common type of mistake, mainly due to typing errors in variable names, not taking care with upper and lower case letters, or missing out % signs at the end of (integer) variable names. A more subtle variation of this error may occur if early on in a program a variable name is mis-spelt, but is referred to correctly in other lines of the program. In this case when you list the line that has been reported as faulty you may find there is nothing obviously wrong with it. For example:

```
360...
370...
380 droprat = startrate + gravity *
time
390...
400 newheight = time * droprate
410...
```

will again result in 'No such variable at line 400' even though line 400

looks perfect. There is nothing for it but to look back through the program for previous occurances of the variables named in the line reported as faulty and so find the real problem, which in this case is in line 380 (too easy!).

Probably the worst possible instance of this type of error, or at least the one that can be hardest to find, can occur when using a procedure in a program, especially one that has many arguments. When the procedure is called the necessary parameters (usually variable names) are listed in brackets after the procedure name, and their values are substituted into the corresponding arguments in the definition of the procedure. The problem is, if one of these variable parameters is invalid then an error will indeed be reported, but at the line number of the procedure definition, not the line number that called the procedure (which has at least got the faulty variable name included in it). If the procedure reporting the error is called only once from elsewhere in the program then it it not too difficult to trace back to the problem area - but a commonly used procedure could be called quite often, and it may not be easy to identify which call is creating the fatal error.

Clearly, some detective work may be necessary to identify the exact cause of errors reported by the computer when you first run a new program (or modify an existing one), but at least the BBC computer has quite comprehensive error checking and reporting, even if it is sometimes misleading. The best approach is to take the initial debugging carefully, and be prepared to look not only at the line reported as faulty, but at possibly related lines. With the right attitude of mind, you may even enjoy the puzzle presented.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

ANOTHER USE FOR *BASIC - S.R. Linter

If you have both Disc Doctor and Watford DFS in your machine, then Disc Doctor traps any use of the *EDIT command before the DFS can react to it. By typing in *FX187,n where n is the number of the ROM socket holding the DFS (0-15), *BASIC will now perform the equivalent of *EDIT directed straight to the Watford DFS.

AMUSING *FX CALL - W.J. Hicks

Following on from the weird GOTO hint, try the following:

```
*FX243,129
*FX243,151
```

These will produce some rather strange effects on the screen. Make sure that you have saved whatever is in the machine before hand. Press the Break key or Return a few times to produce more effects, or experiment with other values! Control-Break will reset the machine afterwards.

POWER UP RESET

If you want to simulate a power up reset (which does not check the *FX247 flag), then keying *FX151,78,127 followed by pressing Break will suffice.

BASICALLY MISTAKEN - Roger Garratt

Basic gets confused in assembler with variable names that begin with a capital 'A'; it thinks you want Accumulator addressing, and assembles accordingly. There are two solutions: use lower case variable names, or enclose the variable in brackets.

LENGTH OF A FILE

If you want to find out quickly the length of a file, use the following code, where name$ holds the file name:

```
len%=EXT# OPENIN name$:CLOSE#0
```

LONG DISC FILES - Dudley Long

To create a large disc file, use a command like:

```
*SAVE FILE 0+31E00
```

This saves a considerable amount of time over other methods.

Tested on Basic I & II and O.S. 1·2

CROSS REFERENCE LISTER (16k)

by Ian Gooding

This month's utility is a very useful program that provides a cross reference listing of a Basic program. It builds up lists of all the variables, procedures, functions, etc used in the program, and the lines at which they are referenced. This program is a must for the serious developer of Basic software.

INTRODUCTION

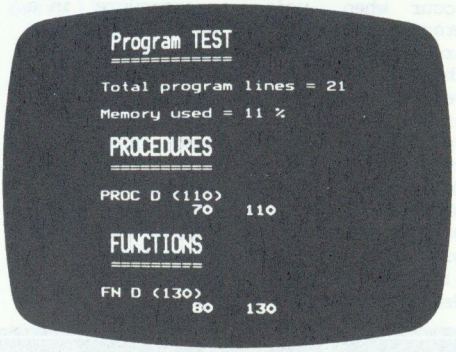
The purpose of this program is to provide the user with a list of names and lines numbers used within his Basic program to aid in debugging and documentation. The program searches for procedure and function names and the lines at which they are referenced; it deals similarly with variable names and lines which have GOTO or GOSUB statements referring to them. Support for printers is included within the program, and any or all of the search options mentioned above may be omitted. The total number of lines in the program is always displayed after the program has been run, as is an indication of the amount of remaining free memory.

USING THE CROSS REFERENCER

The cross referencer should be typed in and saved away for future use. The program to be analysed should be first saved onto disc or tape (provided that for tape you have motor control).

> The program as listed appears longer than it really is as many REM statements have been included to assist those who wish to examine the workings of the program. All REM statements may be omitted when you type the program in. Readers with Basic I should replace OPENUP at line 1200 with OPENIN.

Once run, the cross referencer prompts for your choice of the available options. These include selecting output to the printer, and the checking for procedures, functions, variables, and GOTO and GOSUB references. There is an option to reference all the options without having to type in 'yes' to each of them. Having answered the questions mentioned above, the program prompts for the file name of the program to be analysed, and then procedes to build up the cross reference list.



While the program is running, it displays on the screen the current line being scanned, the most recent procedure, function or variable name read, and the last line number referenced by a GOTO or GOSUB statement. No information is displayed for any option omitted (e.g. no reference to line numbers with GOTO or GOSUB) though the current line number is always displayed.

Once the analysis has been completed, the program lists out the information collected with coloured highlights, with the corresponding text sent to the printer if selected. For an example of the output produced, see the sample listing included with this article.

The program functions quite happily in a 6502 second processor (there is a significant speed improvement in fact). Once the run is finished, you may obtain the list of references again by typing PROCreport.

The necessary data is contained within the program in a fairly complex manner, and space does not permit here a detailed explanation of how this arranged. For the more adventurous

reader, an examination of the program should prove quite interesting as it is very well structured.

LIMITATIONS

A consequence of writing a utility like this in Basic, and keeping the program to a reasonable length is that some simplifications have to be made. Any computed GOTO or GOSUB line number references are ignored (these are impossible to determine in advance anyway). DATA, REM and assembler comment blocks are ignored, and any * calls are treated as variables; e.g. *FX243,129 is treated as a reference to the variable FX243. The program will mark arrays such as A(..) as A(, but for arrays such as A%(..) or A$(..), then only A% or A$ will be displayed.

To provide even more help for programmers, we have included an additional programs on this month's magazine cassette/disc. This is the program PACK, first published in BEEBUG Vol.1 No.9, which allows you to remove spaces, REMs and comments from long programs to save space. The program is accompanied by sufficient instructions for its use if you don't have the original magazine handy. These programs together with the performance analyser published last month provide a most useful set of utilities for serious Basic programmers and we expect to add further such utilities in future issues.

```
   10 REM PROGRAM CROSS REFERENCER
   20 REM AUTHOR IAN GOODING
   30 REM VERSION B1.00
   40 REM BEEBUG NOVEMBER 1984
   50 REM PROGRAM SUBJECT TO COPYRIGHT
   60 :
  100 ON ERROR MODE7:PROCerror:END
  110 MODE 7:crlf$=CHR$13+CHR$10
  120 PROCassemble:PROCbanner:PROCoptio
ns:CLS:PROCbanner
  130 FOR I%=16 TO 23:PRINT TAB(0,I%);C
HR$(130);:NEXT
  140 REM If tape show tape movement, a
llow error retries
  150 A%=0:Y%=0:T%=(USR&FFDA) AND 15:IF
 T%=2 OR T%=3 THEN A%=139:X%=1:Y%=2:CAL
L&FFF4:X%=2:Y%=2:CALL&FFF4
  160 VDU15:REM Scroll screen
  170 mline%=260:REM Space for current
line
  180 DIM line% mline%
  190 REM Set up chain heads
  200 DIM proc% 3:!proc%=-1
  210 DIM fn% 3:!fn%=-1
  220 DIM var% 3:!var%=-1
  230 DIM goto% 3:!goto%=-1
  240 DIM gosub% 3:!gosub%=-1
  250 REM Set up flags
  260 eof%=FALSE:ass%=FALSE:gos%=FALSE
  270 count%=-1
  280 REM Open the program file
  290 f%=FNopenfile
  300 REM Loop reading each line
  310 REPEAT
  320 PROCreadline
  330 IF eof% THEN 360
  340 count%=count%+1
  350 PROCscanline:REM Store references
  360 UNTIL eof%
  370 MODE 7:PROCreport
  380 END
  390 :
 1000 DEF PROCreport
 1010 IF NOT print% THEN VDU 14:prlen%=
0 ELSE PROCsetprinter
 1020 PROCpdouble("Progra"+prog$,CHR$
(134))
 1030 PROCp("","")
 1040 PROCp("Total program lines = "+ST
R$(count%),CHR$130)
 1050 PROCp("","):REM blank line
 1060 DIM P% -1:PROCp("Memory used = "+
STR$(INT(100-(100*(HIMEM-P%))/(HIMEM-TO
P)))+" %",CHR$(130))
 1070 PROCp("","):REM blank line
 1080 IF lproc% THEN PROCpvar(!proc%,"P
ROCEDURES","PROC")
 1090 IF lfn% THEN PROCpvar(!fn%,"FUNCT
IONS","FN")
 1100 IF lvar% THEN PROCpvar(!var%,"VAR
IABLES","")
 1110 IF lgoto% THEN PROCpgo(!goto%,"GO
TO LINES","GOTO")
 1120 IF lgosub% THEN PROCpgo(!gosub%,"
GOSUB LINES","GOSUB")
 1130 PRINT:VDU 15
 1140 ENDPROC
 1150 :
 1160 DEF FNopenfile
 1170 LOCAL f%:VDU 28,6,23,35,16
 1180 REPEAT
 1190 INPUT "Program name :"prog$
 1200 f%=OPENUP(prog$)
 1210 UNTIL f%<>0
 1220 VDU 28,0,24,39,0:=f%
 1230 :
 1240 DEF PROCreadline
 1250 LOCAL i%
 1260 VDU 28,6,23,35,16:REM set screen
subset up
 1270 eline%=line%-1:i%=BGET#f%
```

```
 1280 IF i%<>&0D THEN PRINT "Bad file":
END
 1290 nline%=BGET#f%*256+BGET#f%:REM li
ne number in 2 bytes
 1300 IF (nline% AND &8000) <> 0 THEN C
LOSE#f%:eof%=TRUE:ENDPROC
 1310 i%=BGET#f%:REM character count
 1320 REPEAT
 1330 eline%=eline%+1
 1340 IF (eline%-line%)>mline% THEN PRI
NT "Line buffer overflow":STOP
 1350 ?eline%=BGET#f%
 1360 UNTIL (eline%-line%)=(i%-5)
 1370 ?(eline%+1)=&0D:REM terminator
 1380 VDU 28,0,24,39,0
 1390 ENDPROC
 1400 :
 1410 DEF PROCscanline
 1420 LOCAL i%,j%,k%,q%,quote%,proc$:pr
oc$=""
 1430 def%=0:fnc%=0:quote%=FALSE:gos%=F
ALSE
 1440 A%=POS:B%=VPOS:PRINTTAB(8,5);"Lin
e number : ";RIGHT$("    "+STR$(nline%)
,6);TAB(A%,B%);
 1450 FOR i%=line% TO eline%
 1460 REM &F4 => Keyword "REM", so skip
 out
 1470 REM &DC = Keyword "DATA", ignore
this too
 1480 IF ?i%=&F4 OR ?i%=&DC THEN i%=eli
ne%:GOTO 1890
 1490 REM look for start of assembler
 1500 IF ?i%=ASC("[") AND NOT quote% TH
EN ass%=TRUE:GOTO1890
 1510 IF ?i%=ASC("]") AND NOT quote% TH
EN ass%=FALSE:GOTO1890
 1520 IF ass% THEN GOTO 1890
 1530 REM &DD => Keyword "DEF"
 1540 IF ?i%=&DD THEN def%=2
 1550 REM ":" is statement separator
 1560 IF ?i%=ASC(""""") THEN quote%=NOT
quote%:GOTO1890
 1570 IF ?i%=ASC(":") OR i%=ASC("(") TH
EN def%=0:fnc%=0:proc$="":GOTO1890
 1580 REM Look for GOTO etc line numbers
 1590 REM &E4 = GOSUB    &E5 = GOTO
 1600 REM &F7 = RESTORE  &8C = THEN
 1610 IF ?i%=&E4 THEN gos%=TRUE ELSE IF
 ?i%=&F7 OR ?i%=&E5 OR ?i%=&8C THEN gos
%=FALSE
 1620 REM &8D = line number marker
 1630 IF ?i%=&8D THEN i%=FNscangoto(i%)
:GOTO 1890
 1640 IF ?i%=&B8 AND ?(i%+1)=&50 THEN i
%=i%+1:GOTO 1890:REM bodge for TOP = TO
+P
 1650 REM &F2 => Keyword "PROC"
 1660 REM &A4 => Keyword "FN"
 1670 IF (?i%<>&F2) AND (?i%<>&A4) THEN
 GOTO 1780
 1680 IF ?i%=&A4 THEN fnc%=4 ELSE fnc%=0
 1690 j%=i%+1
 1700 REPEAT
 1710 proc$=proc$+CHR$(?j%):j%=j%+1
 1720 UNTIL i%>eline% OR NOT (FNletter(
?j%) OR ?j%=ASC("#") OR ?j%=ASC("") OR
FNdigit(?j%))
 1730 i%=j%-1
 1740 IF lfn% AND fnc% THEN PROCline(FN
def(fn%,proc$)) ELSE IF (NOT fnc%) AND
lproc% THEN PROCline(FNdef(proc%,proc$))
 1750 proc$="":fnc%=FALSE:def%=FALSE
 1760 GOTO 1890
 1770 REM hexadecimal number ?
 1780 IF ?i%<>ASC("&") THEN GOTO 1810
 1790 REPEAT:i%=i%+1:UNTIL i%>eline% OR
 NOT FNhex(?i%)
 1800 i%=i%-1:GOTO1890
 1810 REM variable reference ?
 1820 IF quote% OR NOT FNletter(?i%) TH
EN GOTO 1890
 1830 var$=""
 1840 REPEAT
 1850 var$=var$+CHR$(?i%):i%=i%+1
 1860 UNTIL i%>eline% OR (NOT FNletter(
?i%)) AND (NOT FNdigit(?i%))
 1870 IF ?i%=ASC("(") OR ?i%=ASC("$") O
R ?i%=ASC("%") THEN var$=var$+CHR$(?i%)
 ELSE i%=i%-1
 1880 IF lvar% THEN PROCline(FNdef(var%
,var$))
 1890 NEXT i%
 1900 ENDPROC
 1910 :
 1920 DEF FNscangoto(i%)
 1930 LOCAL num%:num%=FNgnumber(i%):REM
 extract line number
 1940 i%=i%+3:REM skip over it
 1950 IF lgosub% AND gos% THEN PROCline
(FNgo(gosub%,num%)) ELSE IF (NOT gos%)
AND lgoto% THEN PROCline(FNgo(goto%,num
%))
 1960 =i%
 1970 :
 1980 DEF FNgnumber(x%)
 1990 REM Turn 3 bytes from x% into lin
e number
 2000 REM from internal GOTO format
 2010 ?&70=?(x%+1):?&71=?(x%+2):?&72=?(
x%+3)
 2020 CALL denumb:=256*?&73+?&74
 2030 :
 2040 DEF PROCassemble
 2050 DIM denumb 30
 2060 FOR I%=0 TO 2 STEP 2:P%=denumb:[O
PT I%
 2070 \ Decode GOTO line ref in &70,&71
,&72
 2080 \ to binary line number in &73,&74
 2090 \ Temporary storage in &75
 2100 LDA &70:ASL A:ASL A:STA &75
```

→

```
 2110 AND #&C0:EOR &71:STA &74
 2120 LDA &75:ASL A:ASL A:EOR &72:STA &
73:RTS
 2130 ]:NEXT
 2140 ENDPROC
 2150 :
 2160 DEF FNletter(x%)=((x%>=ASC("A") A
ND x%<=ASC("Z")) OR (x%>=ASC("a") AND x
%<=ASC("z")))
 2170 :
 2180 DEF FNdigit(x%)=(x%>=ASC("0") AND
 x%<=ASC("9"))
 2190 :
 2200 DEF FNhex(x%)=((x%>=ASC("0") AND
x%<=ASC("9")) OR (x%>=ASC("A") AND x%<=
ASC("F")))
 2210 :
 2220 DEF FNdef(chain%,name$)
 2230 A%=POS:B%=VPOS:PRINTTAB(8,7);"Sym
bol :";name$;SPC(40-POS);TAB(A%,B%);
 2240 LOCAL end%,found%,new%,last%
 2250 end%=FALSE:found%=FALSE:last%=cha
in%:chain%=!chain%
 2260 REPEAT
 2270 IF chain%=-1 THEN end%=TRUE:GOTO
2300
 2280 IF name$=$(chain%+11) THEN end%=T
RUE:found%=TRUE:GOTO 2300
 2290 IF name$>$(chain%+11) THEN last%=
chain%:chain%=!chain% ELSE end%=TRUE
 2300 UNTIL end%
 2310 IF found% THEN GOTO 2380
 2320 DIM new% LEN(name$)+11
 2330 !last%=new%:!new%=chain%:chain%=n
ew%
 2340 $(chain%+11)=name$
 2350 ?(chain%+9)=0:?(chain%+10)=0
 2360 DIM new% 3:!new%=-1:!(chain%+4)=n
ew%
 2370 ?(chain%+8)=0
 2380 IF def% THEN ?(chain%+9)=nline%/2
56
 2390 IF def% THEN ?(chain%+10)=nline%
 2400 =chain%
 2410 :
 2420 DEF FNgo(chain%,line%)
 2430 LOCAL end%,found%,new%,last%
 2440 end%=FALSE:found%=FALSE:last%=cha
in%:chain%=!chain%
 2450 A%=POS:B%=VPOS:PRINTTAB(8,9);"Ref
 Line    : ";RIGHT$("    "+STR$(line%),
6);TAB(A%,B%);
 2460 REPEAT
 2470 IF chain%=-1 THEN end%=TRUE:GOTO
2500
 2480 IF FNnumber(chain%+9)=line% THEN
end%=TRUE:found%=TRUE:GOTO 2500
 2490 IF FNnumber(chain%+9)<line% THEN
last%=chain%:chain%=!chain% ELSE end%=T
RUE
 2500 UNTIL end%
 2510 IF found% THEN =chain%
 2520 DIM new% 10:!last%=new%:!new%=cha
in%:chain%=new%
 2530 DIM new% 3:!new%=-1:!(chain%+4)=n
ew%
 2540 ?(chain%+8)=0:?(chain%+9)=line%/2
56
 2550 ?(chain%+10)=line%:=chain%
 2560 :
 2570 DEF PROCline(ref%)
 2580 LOCAL i%,p%,end%,new%,qu%,last%
 2590 end%=FALSE
 2600 last%=!(ref%+4):p%=!last%
 2610 REPEAT
 2620 IF p%=-1 THEN DIM new% 43:!last%=
new%:p%=new%:FOR i%=0 TO 40 STEP 4:!(i%
+new%)=-1:NEXT i%
 2630 i%=2
 2640 REPEAT
 2650 i%=i%+2:qu%=FALSE
 2660 IF i%=42 THEN qu%=TRUE ELSEIF FNn
umber(p%+i%)=&FFFF OR FNnumber(p%+i%)=n
line% THEN qu%=TRUE
 2670 UNTIL qu%
 2680 IF i%<42 THEN end%=TRUE ELSE last
%=p%:p%=!p%
 2690 UNTIL end%
 2700 ?(p%+i%)=nline%/256
 2710 ?(p%+i%+1)=nline%
 2720 ENDPROC
 2730 :
 2740 DEF FNnumber(x%)=(?x%)*256+?(x%+1)
 2750 :
 2760 DEF PROCpvar(chain%,title$,type$)
 2770 LOCAL a$
 2780 PROCpdouble(title$,CHR$(131))
 2790 PROCp("","")
 2800 IF chain%=-1 THEN PROCp("    (NON
E)","):PROCp("","):ENDPROC
 2810 REPEAT
 2820 a$=type$+" "+$(chain%+11)
 2830 IF FNnumber(chain%+9)<>0 THEN a$=
a$+" ("+STR$(FNnumber(chain%+9))+")"
 2840 PROCp(a$,CHR$(134))
 2850 PROCplines(chain%)
 2860 chain%=!chain%
 2870 UNTIL chain%=-1
 2880 ENDPROC
 2890 :
 2900 DEF PROCpgo(chain%,title$,ti$)
 2910 PROCpdouble(title$,CHR$(130)):PRO
Cp("","")
 2920 IF chain%=-1 THEN PROCp("    (NON
E)","):PROCp("","):ENDPROC
 2930 REPEAT
 2940 PROCp(" "+ti$+" "+STR$(FNnumber(c
hain%+9))+" From :",CHR$(134))
 2950 PROCplines(chain%)
 2960 chain%=!chain%
 2970 UNTIL chain%=-1
 2980 ENDPROC
```

→

```
 2990 :
 3000 DEF PROCplines(ref%)
 3010 LOCAL p%,i%,end%,a$:p%=!!(ref%+4)
 3020 REPEAT
 3030 IF p%=-1 THEN GOTO 3110
 3040 i%=2:end%=FALSE:a$=STRING$(6," ")
 3050 REPEAT
 3060 i%=i%+2
 3070 IF i%<42 AND FNnumber(i%+p%)<>&FF
FF THEN a$=a$+RIGHT$("      "+STR$(FNnum
ber(i%+p%)),6) ELSE end%=TRUE
 3080 IF (print% AND LEN(a$)>prlen%-7)
OR (NOT print% AND LEN(a$)>33) THEN PRO
Cp(a$,""):a$="      "
 3090 UNTIL end%
 3100 p%=!p%
 3110 UNTIL p%=-1
 3120 PROCp(a$,""):PROCp("","")
 3130 ENDPROC
 3140 :
 3150 DEF PROCp(text$,control$)
 3160 REM print control$+text$ on screen
 3170 REM print text$ only on printer
 3180 REM if enabled
 3190 PRINT control$;
 3200 IF print% THEN VDU 2
 3210 PRINT text$:VDU 3
 3220 ENDPROC
 3230 :
 3240 DEF PROCpdouble(text$,control$)
 3250 REM print control$ AND text$ in
 3260 REM double height on screen,
 3270 REM print text$ on printer (once)
 3280 REM if enabled
 3290 PRINT CHR$(141);control$;text$
 3300 PRINT CHR$(141);control$;
 3310 IF print% THEN VDU 2
 3320 PRINT text$:VDU 3
 3330 PRINT SPC1;control$;
 3340 IF print% VDU2
 3350 PRINT STRING$(LEN text$,"=")
 3360 VDU3
 3370 ENDPROC
 3380 :
 3390 DEF PROCsetprinter
 3400 REM user to do whatever needed
 3410 prlen%=80:REM line length to use
 3420 REM send line feeds
 3430 REM *FX6,0 if necessary
 3440 ENDPROC
 3450 :
 3460 DEF PROCoptions
 3470 REM User inputs options
 3480 REM Is print output required ?
 3490 print%=FNyesno("Do you want to pr
int the results"+crlf$+"as well as disp
lay them")
 3500 IF NOT FNyesno("Do you want to li
mit the types of detailwhich are record
ed") THEN lvar%=TRUE:lproc%=TRUE:lfn%=T
RUE:lgoto%=TRUE:lgosub%=TRUE:ENDPROC
 3510 lproc%=FNyesno("Do you want PROCs
")
 3520 lfn%=FNyesno("Do you want FNs")
 3530 lvar%=FNyesno("Do you want variab
les")
 3540 lgoto%=FNyesno("Do you want GOTOs
")
 3550 lgosub%=FNyesno("Do you want GOSU
Bs")
 3560 ENDPROC
 3570 :
 3580 DEF PROCbanner
 3590 FOR I%=1 TO 2
 3600 PRINT CHR$(141);CHR$(134);SPC(3);
CHR$(157);CHR$(132);CHR$(139);"Cross Re
ference Lister   ";CHR$(156)
 3610 NEXT:PRINT
 3620 ENDPROC
 3630 :
 3640 DEF FNyesno(prompt$)
 3650 LOCAL end%,ans$:end%=FALSE
 3660 PRINT prompt$;" (Y or N) ";
 3670 REPEAT
 3680 INPUT ans$
 3690 ans$=LEFT$(ans$,1)
 3700 IF ans$="Y" OR ans$="y" OR ans$="
N" OR ans$="n" THEN end%=TRUE ELSE VDU
7:PRINT "Answer Y or N please ";
 3710 UNTIL end%
 3720 =(ans$="Y" OR ans$="y")
 3730 :
 3740 DEF PROCerror
 3750 ON ERROR OFF
 3760 IF ERR<>17 REPORT:PRINT" at line
";ERL
 3770 END
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

DELAY LOOP

A REPEAT UNTIL TIME ... delay loop doesn't have to use positive values, as can be demonstrated by:

```
TIME=-500
REPEAT UNTIL TIME>0
```

This also means that different delay times (maybe set in another program) can use the same delay loop instruction.

THE LATEST ROM AND RAM EXPANSION BOARDS

Reviewed by Peter Rochford

Exprom ROM/RAM Expansion Board
Cost: £61.95 to £82.50 inc. VAT and p&p. depending on options fitted.
Supplier: Anderson Electronics,
2 Hollin Park Road, Calverley,
Pudsey, Leeds.

Aries-B12 ROM Expansion Board
Cost: £46.00 inc. VAT and p&p.
Supplier: Cambridge Computer Consultants,
Science Park, Milton Road, Cambridge.

Ramamp ROM Extension Board
Cost: £26.80 inc. VAT and p&p.
Ramamp ROM/RAM Extension Board.
Cost: £47.00 to £59.00 inc. VAT and p&p.
Supplier: Ramamp Computers,
25 Avon Drive, Whetstone, Leicester.

Computer Village CVX16 ROM/RAM Expansion Board
Cost: £48.88 inc. VAT and p&p.
Supplier: Computer Village Ltd.,
Hazeldine House, Central Square,
Telford, Shropshire.

Name	COB	CIT	2K	4K	8K	16K	RM	IM	EM	SN	BB	WP	Price
Exprom Rom/Ram	16	16	Y	Y	Y	Y	16	Y	Y	N	N	-	£61.95 basic
Ramamp ROM board	7	10	N	Y	Y	Y	0	Y	N	Y	N	N	£26.80
Ramamp ROM/RAM board	7	10	N	N	Y	Y	32	Y	N	Y	N	N	£47.00 basic
Aries B-12	12	16	N	N	Y	Y	16	Y	Y	N	N	-	£46.00
CVX16	16	16	N	N	Y	Y	16	Y	N	N	Y	Y	£48.88
Sir Computers	12	16	Y	Y	Y	Y	16	Y	N	N	N	-	£54.95
Watford MK2	13	16	Y	Y	Y	Y	16	Y	N	Y	Y	Y	£37.38
ATPL	12	16	Y	Y	Y	Y	16	Y	N	N	Y	Y	£39.00 *

Key:

COB	Maximum number of ROMs on the board	BB	Battery backup (if applicable)
CIT	Maximum number of ROMs in total	WP	Write protect (if applicable)
RM	RAM expansion/Maximum figure	£	Price, inclusive of VAT.
IM	Internal mounting	-	Not known
EM	External mounting	*	BEEBUG members price.
SN	Soldering necessary		

In a standard Beeb there are four paged ROM sockets for you to use. If you have Basic in one of them, as most people do, and then add a DFS chip, you are left with only two vacant sockets. With probably in excess of a 100 different ROMs now available for use with a Beeb, this is clearly not sufficient. Fortunately, the circuitry that handles paged ROMs can cope with up to 16 devices. In BEEBUG Vol.2 No.6 we reviewed ROM expansion boards from Watford Electronics, Sir Computers and ATPL. All the boards in that review are still available although the Watford board has undergone a significant improvement, and now includes full on-board buffering. In this review we will be looking at some of the more recent boards to see how they compare.

For reference, the basic information on each board, including those reviewed before, is shown in the table above.

EXPROM ROM EXPANSION BOARD

The overall construction of this board is to a high standard. All the ROM sockets are numbered and easy to identify, as are all the other components such as the links and switches. RAM may be used in 2K or 8K chips; with the 2K devices there can be a maximum of 8K, whilst the 8K devices provide a maximum of 16K. There is a plug-in link for each socket to enable a 'wait state' for EPROMs slower than the normal 250ns.

On the standard board two toggle switches are mounted along the rear edge of the board, and these allow

sockets 14 and 15 to be switched out, or disabled. Two further switches to disable sockets 12 and 13 can be fitted as an option at the time of purchase, bringing the price of the board up to £65.50. When the board is positioned inside the computer, access to the toggle switches is via the slot at the rear of the case. The facility to switch off a ROM can be used to overcome the problem of one ROM interfering with the operation of another. [See also the review of Watford's ROM Manager elsewhere in this issue - Ed.]

A 'zero insertion force' socket can be provided as an alternative in position 4 on the board. This allows ROMs to be inserted and removed easily without much risk of damage. The Exprom board may be installed inside or outside the computer and this position may be changed as and when required.

Connection to the computer requires removal of the 6502 CPU and all the paged ROMs in the BBC micro. These are then plugged into the appropriate sockets on the Exprom board. A 40 way ribbon cable from the Exprom board is then connected to the 6502 socket in the micro with a DIL header plug.

When finally clipped into place on its pillars, the board is very secure and stable. The position of various chips on the board makes the insertion and removal of ROMs more difficult than it might be. However, if you use a proper ROM removal tool (highly recommended and cheap) then you will have no problems, particularly as the sockets are of the older type and it requires only a small amount of force for insertion and removal.

The documentation supplied with the board was in the form of an excellent manual of some 20 pages. The section on the link options takes a fair amount of close reading to understand fully but should not cause any real problem.

RAMAMP ROM EXPANSION BOARD

This board supports seven paged ROMs along with three on the BBC board (the fourth is taken up by the new board). It will accommodate 8K or 16K EPROMs, and by altering some links three of the sockets can utilise 24 pin 4K devices.

Although the quality of construction is very good, the design is less impressive. A four-way ribbon cable must be soldered to four of the pins of IC76 on the main BBC computer board. On some versions of the BBC micro, this IC is soldered in place. Personally, I do not like the idea of soldering direct to the pins of an IC like this. It is difficult to attach four wires to the closely-spaced pins without shorting them out, and there is the possibility of damaging the IC if too much heat is applied trying to rectify mistakes.

The documentation supplied with this board consisted of two sheets of clear instructions and a single sheet of diagrams. If you live close enough to their workshops (see address at head of review), then Ramamp will fit the board for you at no extra cost.

RAMAMP ROM/RAM EXPANSION BOARD

This board is very similar to the Ramamp ROM board described above, except that it can accommodate paged RAM. It allows the use of 8k and 16K EPROMs but not the 4K type. There are two versions of the board available; one fitted with 16K of paged RAM (costing £47.00) and the other with two 16K pages of RAM (costing £59.00). The documentation consisted of just a sheet of text and a sheet of diagrams.

ARIES B-12 ROM/RAM EXPANSION BOARD

This system comprises three boards. The main board carries the ROMs and a second the buffering and decoding chips. The quality and standard of construction of these three boards is very impressive indeed.

If you have the Aries B-20 RAM expansion board fitted (reviewed in BEEBUG Vol.2 No.9), the 6502 CPU is not removed and a 5-way ribbon cable is plugged into the socket provided on the B-20. If not, the 6502 is plugged into the other small board provided with the B-12 ROM board. This small carrier board is then plugged into the 6502 socket on the BBC by means of pins on the underside and the 5-way ribbon cable plugged into the socket provided on this.

The main board supports up to twelve paged EPROMs and 16K of paged RAM may be added in two 8K devices. All the sockets are numbered very clearly for ease of identification and are arranged in a 2 x 7 layout. There is no battery backup facility provided for the paged RAM.

Documentation consisted of a very comprehensive sixteen page manual, with clear instructions and plenty of illustrations.

COMPUTER VILLAGE CVX16 ROM/RAM EXPANSION BOARD

Again the quality and construction of the double sided PCB is excellent and features some eighteen buffering and decoding chips.

The board will accept 8K and 16K EPROMs and has a paged RAM facility allowing the use of 2K CMOS or NMOS RAM devices up to a maximum of 16K. A battery backup facility is provided for the paged RAM as standard, requiring only the addition of the Ni-Cad battery. Paged RAM on this board can be write-protected under software control. Using the cheaper 2K RAM chips uses up 8 sockets allowing space for only 8 ROMs to be fitted at the same time.

CONCLUSIONS

I am pleased to say that all the boards worked without fault during testing. Every board was operated continuously inside the computer for a period of at least six hours and none suffered with overheating. Because the Computer Village board is installed so close to the computer's RAM chips, which produce a good deal of heat, it was soak tested for twelve hours but continued to work perfectly.

Which of these boards you choose will depend on the cost, the facilities you need and the ease of fitting. If you change ROMs regularly, the convenience of doing this will be important to you.

The Ramamp ROM board provides an inexpensive solution for those needing up to six extra ROM sockets. The ROM/RAM version is particularly good value when you consider the RAM comes with the board and enables you to store some of your ROMs on disc and load them in when needed. However, you must be prepared to undertake some soldering (as described above) when fitting either of these boards.

The Computer Village board needs modifications, I feel, to provide a more satisfactory way of attaching it to the computer, and to make it more secure. It is sad that little thought seems to have been given to this, yet the board itself is so well made and provides good facilities. This is the only ROM/RAM board in the review that can provide battery backup for the paged RAM. I hope Computer Village give some attention to the problems I have mentioned before they release any new version of this board.

The last two units, the Exprom and Aries boards, both provide the best way, in my opinion, of attaching a ROM board to your computer. That is externally in its own case. A great deal of thought went into the design of both of these boards and it shows.

If it's facilities you're after then the Exprom board may be considered as the unit for you. I have yet to come across another ROM/RAM board that has as many operating options as this one. My only dislikes about this board were the unco-operative ribbon cable and the slight difficulty in removing ROMs. Don't let this put you off too much though, this unit is well worth serious consideration even if you don't need all the extra facilities.

Finally, my own personal choice out of all these boards is the Aries B-12. It is beautifully made, easy to fit, provides all the facilities that I need and is realistically priced. It can be mounted internally and externally, removal of ROMs is straightforward and the unit is secure and stable. True it supports only twelve paged ROMs but the paged RAM can be used to load some of your ROMs from disc. I find it hard to fault this board in any way.

Of the boards reviewed previously that produced by ATPL still stands out as being excellent value, especially at the BEEBUG members price. Our previous comments on the other two boards by and large remain valid. Ed.

BEEBUG Workshop

FORMATTING TEXT

by Surac

This month we take a look at the problem of printing text on the screen without splitting words over the end of a line. A method to achieve this is explained in this month's workshop together with a description of a machine code routine to put this into effect.

Imagine, say, that you are writing an adventure game and that you want to print out the description for the current room. You are working in mode 7 (with a screen width of 40 characters), but the text you have is over 40 characters long, and you wish to avoid any breaks in the words. Obviously, you could type in the text with breaks at the appropriate points, but this would be tedious, and make changing a description inconvenient. What you need is a routine that takes each word, and looks to see if it will fit into the current line totally, or if you need to start on the next line before you can print it. What we're going to do is discuss exactly how we want our formatting routine to work, and then I'll provide an example of how to code it written in machine code for you to use.

Basically, the way to print the text in the required format is to take each word, add its length to the current position of the cursor on the screen, and compare this with the width of the screen that we are using. If the new position is before the end of the line, we print the next word and a space, and carry on. If the new word finishes exactly on the end of the line, we just print the word, but no space. If the word overlaps, we start on the next line down, and print the word and a space. This process will ensure that each line is filled with whole words (but not right justified).

The routine listed here includes a short program in Basic to demonstrate its use. Type it all in and save it to tape or disc first (particularly important with machine code which may completely corrupt the program when run if any typing errors have been introduced). If the program is then run, it will assemble the machine code, and then prompt for the mode you wish to test it in. You will also be asked for a string to format, and the screen width you wish to use. If you want to use the full screen width for the mode selected, just enter zero. The demonstration is repeated as often as you wish.

Using the routine in your own programs is quite simple. All you have to do is to add the machine code routine contained in the procedure PROCassemble, the two functions FNequb and FNequs (the use of these is equivalent to the functions EQUB and EQUS available in Basic II and Hi-Basic), and the procedure PROCformat. The procedure PROCassemble should be called to assemble the machine code before PROCformat is used in your program. The format procedure takes two parameters, the string to format, and the line width to be used. The width is supplied as a parameter in case you should want to use text with a width less than that of the full screen (perhaps with a text window for example). However, if the width is specified just as zero then the current mode is used to determine the screen width using a small look-up table (the data at line 2150). This is much easier in machine code than using an expression to calculate the result as in last month's Basic program.

The formatting routine could quite easily be used by itself in a machine

code program. In this case the X and Y registers on entry point to a block of text, with a zero byte marking the end. The block can be virtually any length, but an individual word should not be longer than 255 characters. Note that Return and other characters that don't occupy a single space on the screen will lead to errors in the accuracy of the formatting.

```
   10 REM PROGRAM PRETTY
   20 REM VERSION B1.00
   30 REM AUTHOUR SURAC
   40 REM BEEBUG  NOVEMBER 1984
   50 REM PROGRAM SUBJECT TO COPYRIGHT.
   60 :
  100 PROCassemble
  110 PRINT "MODE :"
  120 MODE GET-48
  130 REPEAT
  140 INPUT'"String to format:"string$
  150 INPUT"Width for format:"linelengt
h%
  160 PROCformat(string$,linelength%)
  170 UNTIL 0
  180 END
  190 :
 1000 DEF PROCformat($buffer,W%)
 1010 ?width=W%
 1020 buffer?LEN$buffer=0
 1030 X%=buffer:Y%=buffer DIV 256
 1040 CALL pretty
 1050 ENDPROC
 1060 :
 1070 DEF PROCassemble
 1080 DIM space 300,buffer 256
 1090 look=&70:width=&72
 1100 xpos=&73:byte=&74
 1110 FOR PASS%=0TO2 STEP 2
 1120 P%=space
 1130 [OPT PASS%
 1140 .pretty
 1150 STX look
 1160 STY look+1
 1170 LDA width
 1180 BNE pretty1
 1190 JSR findwidth
 1200 STA width
 1210 .pretty1
 1220 LDA #134
 1230 JSR &FFF4
 1240 STX xpos
 1250 LDY #0
 1260 .pretty2
 1270 LDA (look),Y
 1280 BEQ pretty3
 1290 CMP #32
 1300 BEQ pretty4
 1310 INY
 1320 BNE pretty2
 1330 BRK
 1340 OPT FNequb(19)
 1350 OPT FNequs
      ("String too long")
 1360 OPT FNequb(0)
 1370 .pretty3
 1380 JMP patch1
 1390 .pretty4
 1400 TYA
 1410 CLC
 1420 ADC xpos
 1430 CMP width
 1440 BEQ pretty5
 1450 BCC pretty6
 1460 ROR byte
 1470 JSR &FFE7
 1480 JMP pretty7
 1490 .pretty5
 1500 LDA #0
 1510 STA byte
 1520 BEQ pretty7
 1530 .pretty6
 1540 LDA #255
 1550 STA byte
 1560 .pretty7
 1570 LDY #0
 1580 .pretty8
 1590 LDA (look),Y
 1600 CMP #32
 1610 BEQ pretty9
 1620 JSR &FFEE
 1630 INY
 1640 BNE pretty8
 1650 .pretty9
 1660 BIT byte
 1670 BPL prettya
 1680 LDA #32
 1690 JSR &FFEE
 1700 .prettya
 1710 LDA (look),Y
 1720 CMP #32
 1730 BNE prettyb
 1740 INY
 1750 BNE prettya
 1760 .prettyb
 1770 TYA
 1780 CLC
 1790 ADC look
 1800 STA look
 1810 LDA look+1
 1820 ADC #0
 1830 STA look+1
 1840 LDY #0
 1850 JMP pretty1
 1860 .patch1
 1870 TYA
 1880 CLC
 1890 ADC xpos
 1900 CMP width
 1910 BCC patch2
 1920 JSR &FFE7
 1930 .patch2
 1940 LDY #0
 1950 .patch3
 1960 LDA (look),Y
 1970 BEQ patch4
 1980 JSR &FFEE
 1990 INY
 2000 BNE patch3
 2010 .patch4
 2020 RTS
 2030 .findwidth
 2040 LDA #135
 2050 JSR &FFF4
 2060 LDA data,Y
 2070 RTS
 2080 .data
 2090 ]
 2100 NEXT
 2110 RESTORE
 2120 FORI%=0TO7
 2130 READ data?I%
 2140 NEXT
 2150 DATA 80,40,
      20,80,40,20,
      40,40
 2160 ENDPROC
 2170 :
 2180 DEF FNequb(A%)
 2190 ?P%=A%
 2200 P%=P%+1
 2210 =PASS%
 2220 :
 2230 DEF FNequs(A$)
 2240 $P%=A$
 2250 P%=P%+LEN A$
 2260 =PASS%
```

WATFORD ELECTRONIC'S ROM MANAGER

Reviewed by David Fell

Product : ROM Manager
Supplier: Watford Electronics,
35 Cardiff Road
Watford
Price: £21.50 (special price to BEEBUG members)

One of the problems that may arise now that so much ROM based software is available for the Beeb is that, almost inevitably, a well populated machine will contain two or more ROMs which have been programmed to recognise the same command; perhaps the most common example being the use of 'EDIT'. BEEBUG has already proposed a 'ROM Rule' to alleviate this situation (see last month's editorial), and now Watford Electronics have come up with 'ROM Manager' which is intended to provide comprehensive management of all your installed ROMs, including conflicts with ROM commands.

Probably the most useful feature is the *DIRECT command. This allows any command to be passed directly to a specified ROM. Because this command is so valuable, says the manual, there is an alternative command name for performing the same action; viz the *VECTOR command. *STOP and *START are two commands that allow sideways ROMs to be turned on or off. All * commands are then ignored by any ROMs in an 'off' state.

There are also various other commands provided by ROM Manager. These will list out the ROMs present in the machine, display detailed information on a specified ROM, list the contents of a ROM, edit memory, and calculate the checksum of a given ROM. There is also a command to list the function key definitions (although this doesn't cope with a delete character in a definition), and a command to display help on the first 22 (0 to 21) FX calls. This seems to be rather a space filler as the Advanced User Guide contains most of the relevant facts.

The final selection of commands allows you to develop sideways ROMs in normal memory (not sideways RAM), and to have all service calls directed to your 'ROM'. This is a nice feature, even if of interest to just a few.

If your Beeb is filled with ROMs, as mine is, then Watford Electronics' ROM Manager will at last allow you to avoid many of the conflicts and other problems that have bedevilled the use of ROM-based software until now. It is worth it for this feature alone.

BEEBUG has negotiated a deal with Watford Electronics whereby members may purchase ROM Manager for just £21.50 (including VAT and postage). Send your order direct to Watford Electronics. It is ESSENTIAL to quote your BEEBUG membership number, and state that you require the BEEBUG discount.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

6502 TUBE AND THE GRAPHICS ROM - Roy A. Harcup

To use the facility of the Computer Concepts' Graphics ROM to pass integer parameters over the Tube, you need to form a string and place this in an OSCLI statement. For example,

```
*IN 1 500 Y%
```

would be encoded as

```
A$="IN 1 500 "+STR$Y%
OSCLI A$
```

Do not, however, try to use the trig routines built in to the Graphics ROM in this way, as these will not work, and the integrity of the machine cannot be guaranteed.

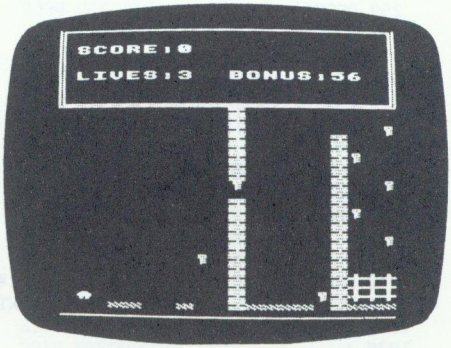
Tested on Basic I & II and O.S. 1.2

WEE SHUGGY (32k)

by Hugh Darby

If you enjoyed Block Blitz published in BEEBUG Vol.2 No.8 (acclaimed by many members as one of our best ever games) then you will certainly like Wee Shuggy, a game about a small guy trapped in a dungeon. He needs your help to find several keys and escape from the nightmare world in which he finds himself.

Wee Shuggy is a version of the popular computer game called 'Manic Miner' in which the player has to fetch all of the keys to open a grating so that he can escape from the dungeon. There are various obstacles on the way including thorn bushes, floors that collapse as you walk on them, stalactites and conveyor belts.



You lose a life if you jump into a stalactite or a thorn bush, and then have to start the current screen from the beginning again.

You have three lives in all, and there is a high score table and instructions in the program. The game uses just three keys to control the movements of the man, 'Z' and 'X' for left and right, and 'Return' for jump.

There are six different dungeons in this fast and colourful game, and each one provides a challenging, enjoyable and often frustrating experience.

We are sure that once you start playing Wee Shuggy you will agree that this is one of the most outstanding games that we have published in BEEBUG. The time and effort spent typing the program into your micro will be well rewarded.

PROGRAM NOTES

The program is well structured, but care is still required when typing it in. Extensive use is made of user-defined characters set up in the procedure PROCcharacter from line 3490 onwards. These characters are used extensively in displaying the different screens used in the game (procedures screen, screen2, screen3, screen4, screen5, and screen6) by putting the character codes in a series of VDU statements in each case. The VDU statement is more concise as only the character code is required (224, 225 etc) rather than the character itself (CHR$224, CHR$225 etc) as would be needed in a PRINT statement. It is also easy to include any of the other VDU codes (in the range 0 - 31) to change colour, move the cursor etc as required.

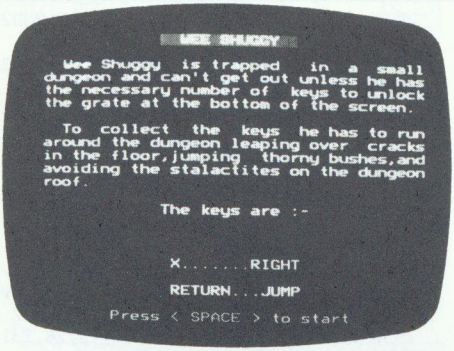
For speed, the program also incorporates a short machine code routine PROCassem (to check on the screen character at the current cursor position) and uses direct memory access (the procedure PROCobject for example).

All the procedures have reasonably meaningful names and you should be able to identify their functions without too much difficulty.

```
 10 REM PROGRAM WEE SHUGGY
 20 REM VERSION B0.2
 30 REM AUTHOR  HUGH DARBY
 40 REM BEEBUG  NOVEMBER 1984
 50 REM PROGRAM SUBJECT TO COPYRIGHT
 60 :
100 ON ERROR GOTO 3710
110 MODE7
120 PROCcharacter:PROCassem
130 PROCinstr:PROCarray
140 REPEAT
150 PROCvar:PROChi
```

→

```
  160 MODE5:VDU23,1,0;0;0;0;
  170 REPEAT:CLS
  180 ONSCR%GOTO190,200,210,220,230,240
,250
  190 PROCscreen:GOTO260
  200 PROCscreen2:GOTO260
  210 PROCscreen3:GOTO260
  220 PROCscreen4:GOTO260
  230 PROCscreen5:GOTO260
  240 PROCscreen6:GOTO260
  250 PROCfinale
  260 PROCheader
  270 REPEAT
  280 PROCplayer:PROCtest
  290 UNTILdeadORfinish
  300 IFdeadPROCfinishELSEPROCnewscr
  310 UNTILdead
  320 MODE7:PROCend
  330 UNTILFALSE
  340 END
  350 :
 1000 DEFPROCarray
 1010 DIMN$(5),HI%(5):FORI%=1TO5:N$(I%)
="BEEBUG MAG":HI%(I%)=3000-I%*500:NEXT
 1020 ENDPROC
 1030 :
 1040 DEFPROCassem
 1050 P%=&80
 1060 [OPT2:LDA#135:JSR&FFF4:TXA:CLC:AD
C#96:TAX:STX&70:RTS:]
 1070 ENDPROC
 1080 :
 1090 DEFPROChi
 1100 CLS:PRINT
 1110 HD$=CHR$129+CHR$157+CHR$131+"TODA
YS TOP SHUGGYS  "+CHR$156:PROCdouble(HD
$)
 1120 FORI%=1TO5
 1130 VDU31,0,I%*3+1
 1140 string$="   "+CHR$(129+I%)+STR$(I
%)+"..."+FNspc+N$(I%)+FNspc+"..."+STR$(
HI%(I%))
 1150 PRINTstring$
 1160 NEXT
 1170 PRINT'':HD$=CHR$134+"Press"+CHR$1
31+"<"+CHR$130+"SPACE"+CHR$131+">":PROC
cent(HD$)
 1180 REPEATUNTILGET=32
 1190 CLS
 1200 ENDPROC
 1210 :
 1220 DEFPROCdouble(A$)
 1230 A$=CHR$141+A$
 1240 PROCcent(A$):PROCcent(A$)
 1250 ENDPROC
 1260 :
 1270 DEFPROCcent(A$)
 1280 pos=20-LENA$/2
 1290 VDU31,pos,VPOS:PRINTA$
 1300 ENDPROC
 1310 :
 1320 DEFFNspc=STRING$((20-LENN$(I%))/2
,CHR$32)
 1330 :
 1340 DEFPROCscreen
 1350 CLS:COLOUR1:PRINTTAB(0,30);STRING
$(20,CHR$224);CHR$30;CHR$11
 1360 VDU31,4,27,224,224,224,224,224,22
4,17,2,232,232,232,31,10,26,232,232,232
,17,1,224,225,225,225,225,224,224
 1370 VDU31,0,22,224,224,224,32,32,32,2
24,224,224,224,224,17,2,232,232,232,17,
1,228,228,228,32,224,224,31,11,21,17,2,
232,232,232,17,3,229
 1380 VDU31,0,18,17,1,224,224,31,0,14,2
24,224,224,224,224,224,224,224,224,225,
225,224,225,225,225,224,224,224,224,224
 1390 VDU31,6,26,17,3,229,31,15,13,229,
17,2,231,31,19,11,231,31,4,10,231,32,17
,3,230,32,230,32,32,230,32,17,2,231
 1400 KEYS=4
 1410 ENDPROC
 1420 :
 1430 DEFPROCheader
 1440 VDU19,3,2;0;
 1450 IFG%<>0GOTO1470
 1460 COLOUR2:FORI%=28TO30:VDU31,17,I%,
233,233,233:NEXT
 1470 COLOUR132:FORI%=1TO9:PRINTTAB(0,I
%)SPC20:NEXT
 1480 PRINTTAB(1,3)"SCORE:";SC%;TAB(1,6
)"LIVES:";liv%;TAB(10,6)"BONUS:60"
 1490 MOVE0,704:DRAW0,992:DRAW1279,992:
DRAW1279,704:DRAW0,704
 1500 MOVE16,712:DRAW16,984:DRAW1262,98
4:DRAW1262,712:DRAW16,712
 1510 TIME=0
 1520 ENDPROC
 1530 :
 1540 DEFPROCinstr
 1550 PRINT:HD$=CHR$129+CHR$157+CHR$131
+"WEE SHUGGY  "+CHR$156:PROCcent(HD$)
 1560 PRINT'CHR$131" Wee Shuggy is tr
apped  in a small"CHR$131"dungeon an
d can't get out unless he has"CHR$131"t
he necessary number of keys to unlock"
CHR$131"the grate at the bottom of the
screen."
 1570 PRINT'CHR$131" To collect the
 keys he has to run"CHR$131"around the
 dungeon leaping over cracks"CHR$131"i
n the floor,jumping thorny bushes,and"
CHR$131"avoiding the stalactites on the
 dungeon"CHR$131"roof."
 1580 PRINT:PROCcent(CHR$131+" The keys
 are :-")
 1590 PRINT:PROCcent(CHR$129+"Z.......
LEFT")
 1600 PRINT:PROCcent(CHR$130+"X.......R
IGHT")
```



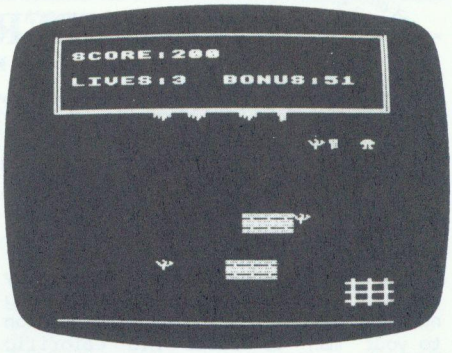

```
 1610 PRINT:PROCcent(CHR$134+"RETURN...
JUMP")
 1620 PRINT:PROCcent(CHR$133+"Press"+CH
R$136+"< SPACE >"+CHR$137+"to start")
 1630 REPEATUNTILGET=32:CLS
 1640 ENDPROC
 1650 :
 1660 DEFPROCvar
 1670 X%=0:Y%=30:J%=0:DIR%=0:G%=0
 1680 OX%=X%:OY%=Y%:SC%=0:liv%=3
 1690 dead=0:finish=0:SCR%=1:keys=0
 1700 ENDPROC
 1710 :
 1720 DEFPROCprint(c)
 1730 IFc<>0GOTO1760
 1740 *FX19
 1750 VDU31,X%,Y%,32:ENDPROC
 1760 *FX19
 1770 VDU17,c,31,X%,Y%,253+X%MOD2
 1780 ENDPROC
 1790 :
 1800 DEFPROCplayer
 1810 PROCprint(0)
 1820 IFFNcheck(X%,Y%+1)=0ANDJ%=0:Y%=Y%+
1:IFADVAL-6=15SOUND1,2,30-Y%,1:GOTO1900
 1830 IFFNcheck(X%,Y%+1)<>0PROCobj2
 1840 OX%=X%:OY%=Y%
 1850 IFINKEY-74ANDJ%=0DIR%=INKEY-98-IN
KEY-67:J%=1:UNO%=-1:IFADVAL-6=15SOUND1,
3,1,2
 1860 IFJ%<>0PROCjump:GOTO1890
 1870 IFINKEY-67ANDX%<19X%=X%+1
 1880 IFINKEY-98ANDX%>0X%=X%-1
 1890 IFFNcheck(X%,Y%)<>0PROCobject
 1900 COLOUR2:PRINTTAB(16,6);60-TIMEDIV
100;CHR$32
 1910 PROCprint(2)
 1920 ENDPROC
 1930 :
 1940 DEFFNcheck(x%,y%)
 1950 VDU31,x%,y%:CALL&80
 1960 IF?&70=96OR?&70=128THEN=0ELSE=1
 1970 :
 1980 DEFPROCobject
 1990 IF?&70>=224AND?&70<229J%=0:UNO%=0
:Y%=Y%-1:ENDPROC
 2000 IF?&70=229OR?&70=230OR?&70=234dea
d=-1:ENDPROC
 2010 IF?&70=231PROCbonus:ENDPROC
 2020 IF?&70=232X%=OX%:Y%=OY%:ENDPROC
 2030 IF?&70=233ANDKEYS=keys finish=-1E
LSEIF?&70=233X%=16:Y%=30:ENDPROC
 2040 IF?&70=235PROCprint(0):X%=6:Y%=19
 2050 IF?&70=236PROCprint(0):X%=14:Y%=19
 2060 ENDPROC
 2070 :
 2080 DEFPROCjump
 2090 X%=X%+DIR%:J%=J%+1:Y%=Y%+UNO%
 2100 IFFNcheck(X%,Y%)<>0PROCobject
 2110 IFJ%>3UNO%=1
 2120 IFJ%=7UNO%=0:J%=0
 2130 ENDPROC
 2140 :
 2150 DEFPROCtest
 2160 IFY%>30dead=-1:Y%=30ELSEIFY%<11PR
OCprint(0):Y%=11
 2170 IFX%>18PROCprint(0):X%=18ELSEIFX%
<1PROCprint(0):X%=1
 2180 IFTIME>=6000 dead=-1
 2190 ENDPROC
 2200 :
 2210 DEFPROCcollapse
 2220 VDU17,1
 2230 IF?&70=227VDU31,X%,Y%+1,32:ENDPROC
 2240 VDU31,X%,Y%+1,?&70+1
 2250 ENDPROC
 2260 :
 2270 DEFPROCbonus
 2280 SC%=SC%+100:keys=keys+1:VDU17,0,3
1,X%,Y%,32
 2290 SOUND1,1,200,1
 2300 COLOUR2:PRINTTAB(7,3);SC%
 2310 ENDPROC
 2320 :
 2330 DEFPROCobj2
 2340 IF?&70>224AND?&70<228PROCcollapse
:ENDPROC
 2350 IF?&70=228X%=X%-1:ENDPROC
 2360 IF?&70=233ANDKEYS=keys finish=-1E
LSEIF?&70=233X%=16:Y%=30:ENDPROC
 2370 IF?&70=234dead=-1:ENDPROC
 2380 IF(?&70=229OR?&70=230)ANDJ%=0Y%=Y
%+1:ENDPROC
 2390 IF?&70=231ANDJ%=0Y%=Y%+1
 2400 ENDPROC
 2410 :
 2420 DEFPROCnewscr
 2430 T=TIME
 2440 CLS:SCR%=SCR%+1:SC%=SC%+60-TDIV100
 2450 X%=0:Y%=30:J%=0:DIR%=0:OX%=X%:OY%
=Y%:dead=0:finish=0:keys=0
 2460 IFSCR%<7 PRINTTAB(0,10)"GOING ONT
O SCREEN";SCR%ELSEENDPROC
 2470 T=TIME:REPEAT UNTIL TIME=T+300
 2480 ENDPROC
 2490 :
```

→

```
2500 DEFPROCfinish
2510 CLS:FORI%=100TO0STEP-4:SOUND1,3,I
%,2:NEXT
2520 liv%=liv%-1:IFliv%<=0dead=TRUE:EN
DPROC
2530 X%=0:Y%=30:J%=0:DIR%=0:OX%=X%:OY%
=Y%:dead=0:finish=0:keys=0
2540 IFTIME>=6000PRINTTAB(0,10)"YOU RA
N OUT OF TIME"
2550 ENDPROC
2560 :
2570 DEFPROCend
2580 *FX21,0
2590 FORT%=255TO0STEP-1:SOUND&11,4,T%,
1:SOUND&10,4,7,1:NEXT
2600 T=TIME:REPEAT UNTIL TIME=T+300
2610 IFSC%<=HI%(5)ENDPROC
2620 FORI%=5TO1STEP-1
2630 IFHI%(I%)<SC%C%=I%
2640 NEXT
2650 PRINT'''':PROCcent(CHR$129+CHR$136
+"WELL DONE!  ")
2660 PRINT':PROCcent(CHR$130+"YOUR SCO
RE OF "+STR$(SC%))
2670 PRINT':PROCcent(CHR$130+"IS ENOUG
H TO RANK YOU "+STR$(C%)+MID$("stndrdth
th",C%*2-1,2))
2680 PRINT':PROCcent(CHR$134+"PLEASE E
NTER YOUR NAME ")
2690 INPUTTAB(5,15)A$
2700 IFLENA$>20CLS:GOTO2650
2710 IFC%=5GOTO2740
2720 FORI%=4TOC%STEP-1
2730 HI%(I%+1)=HI%(I%):N$(I%+1)=N$(I%)
:NEXT
2740 HI%(C%)=SC%:N$(C%)=A$
2750 ENDPROC
2760 :
2770 DEFPROCscreen2
2780 *FX21
2790 VDU31,0,30,17,3:FORI%=0TO19:VDU23
4:NEXT:VDU30,11
2800 VDU31,0,10,17,3:FORI%=0TO19:VDU23
0:NEXT
2810 VDU31,0,15,17,1:FORI%=0TO19:VDU22
5:NEXT
2820 VDU31,0,31,224,224,224,11,11,32,3
2,224,224,11,11,32,32,224,224,11,11,32,
32,224,224,31,8,30,228,228,228,228,32,3
2,10,225,225,224
2830 VDU31,7,22,225,225,225,32,32,32,3
2,32,32,225,225,225,31,0,26,224,224,31,
3,22,224,224,31,5,18,224,224
2840 VDU31,8,29,17,2,231,31,8,21,231,3
1,17,21,231,31,15,14,231,231,231
2850 keys=0:KEYS=6
2860 ENDPROC
2870 :
2880 DEFPROCscreen3
2890 *FX21
2900 VDU31,0,30,17,3:FORI%=0TO19:VDU23
4:NEXT:VDU30,11
2910 VDU31,0,10,17,3:FORI%=0TO19:VDU23
0:NEXT
2920 VDU31,0,31,17,1,224,224,225,31,3,
28,228,228,11,11,11,228,228,11,11,11,22
8,228,11,11,11,228,228,225,225,32,32,22
4,224,225,225,225
2930 VDU31,0,27,224,224,31,0,23,224,22
4,31,0,19,224,224,31,17,27,224,224,31,6
,13,225,225,225,225,224,225
2940 VDU31,3,16,225,225,225,31,11,27,2
24,224,31,15,20,17,2,232,8,10,232,8,10,
232,8,10,232,31,13,31,232,232,232,232
2950 VDU17,2,31,0,26,231,31,18,26,231,
31,10,12,231,31,18,22,231,31,6,24,231
2960 keys=0:KEYS=5
2970 ENDPROC
2980 :
2990 DEFPROCscreen4
3000 *FX21
3010 VDU31,0,30,17,3:FORI%=0TO19:VDU23
4:NEXT:VDU30,11
3020 VDU31,10,12,17,2:FORI%=0TO18:VDU2
32,10,8:NEXT:VDU232
3030 VDU17,1,31,0,31,224,224,31,0,27,2
24,224,31,0,23,224,224,32,225,225,32,11
,11,224,11,11,224,11,11,224,31,0,14,228
,228,228,228,228,224,224,224
3040 VDU31,11,12,225,225,225,31,11,13,
17,2,230,230,230,17,1,31,14,15,228,228,
31,12,18,228,228,31,11,31,224,224
3050 VDU31,13,27,224,224,31,15,23,224,
224,31,17,19,224,224,17,2,31,16,27,232,
232,232,232,31,16,26,232,232,232,232,17
,1,31,16,31,224,31,18,15,224,224
3060 VDU17,2,31,0,26,231,31,11,30,231,
31,19,14,231,31,19,25,231,31,0,13,231
3070 keys=0:KEYS=5
3080 ENDPROC
3090 :
3100 DEFPROCscreen5
3110 *FX21
3120 VDU31,0,30,17,3:FORI%=0TO19:VDU23
4:NEXT:VDU30,11
3130 VDU31,10,10,17,2:FORI%=0TO20:VDU2
32,10,8:NEXT:VDU232
3140 VDU31,16,13:FORI%=0TO17:VDU232,10
,8:NEXT:VDU232
3150 VDU17,1:FORI%=13TO27STEP3:VDU31,1
7,I%,225,225,225:NEXT
3160 VDU17,1,31,0,31,224,224,224,31,5,
31,224,224,31,8,31,224,224,31,8,27,224,
224,31,5,24,224,224,31,0,21,224,224,224
,31,0,17,224,224,224
3170 VDU31,0,13,224,224,224,32,225,225
,225,225,225,31,9,20,224,11,32,8,11,32,
31,11,20,225,225,32,225,225,31,13,16,22
8,228,228,31,13,13,228,228,228,31,15,31
,225
```

→

```
3180 VDU17,2,31,19,12,231,31,17,15,231
,31,19,18,231,31,17,21,231,31,19,24,231
,31,8,26,231,31,10,18,231,31,15,30,231
3190 keys=0:KEYS=8
3200 ENDPROC
3210 :
3220 DEFPROCscreen6
3230 *FX21
3240 VDU17,1,31,0,31,224,224,224,32,22
7,227,32,17,3,234,17,1,227,32,227,227,2
24,225,17,3,234,234,17,1,225,224,225,22
4,30,11,31,2,27,224,31,1,23,224,224,31,
1,19,235,31,5,20,228,228
3250 VDU17,2:FORI%=31TO19STEP-1:VDU31,
3,I%,232,31,9,I%,232:NEXT
3260 FORI%=27TO19STEP-1:VDU31,12,I%,23
2,232,31,16,I%,232:NEXT:VDU31,7,19,232,
8,10,232,8,10,232,31,6,29,232,8,10,232,
8,10,232
3270 FORI%=0TO19:VDU31,I%,18,232,31,I%
,17,232:NEXT:VDU31,14,17,32,32,31,14,18
,32,32
3280 VDU17,1,31,8,27,224,31,8,23,224,3
1,8,20,224,11,32,31,10,27,227,31,10,23,
227,17,3,31,10,24,230,31,11,21,230,17,2
,31,11,26,232,17,1,31,18,27,224,224,31,
18,23,224,224,31,18,19,236
3290 VDU31,14,17,226,226,31,14,20,226,
226
3300 VDU17,2,31,2,19,231,31,6,28,231,3
1,9,19,231,31,10,22,231,31,13,28,231,31
,16,28,231,31,17,16,231
3310 FORI%=14TO16:VDU31,0,I%,233,233,2
33,231,231,231:NEXT
3320 keys=0:KEYS=16:G%=1
3330 ENDPROC
3340 :
3350 DEFPROCfinale
3360 FORT%=0TO255:SOUND&11,4,T%,1:SOUN
D&10,4,7,1:NEXT:SOUND0,1,7,1
3370 ENVELOPE1,1,0,0,0,0,0,0,0,0,-1,-1
,126,126
3380 SOUND17,1,101,2:SOUND1,0,0,1:SOUN
D1,1,101,2:SOUND1,0,0,1:SOUND1,1,101,2:
SOUND1,0,0,1:SOUND1,1,125,5:SOUND1,0,0,
2:SOUND1,1,101,2:SOUND1,0,0,2:SOUND1,1,
125,5
3390 COLOUR1:PRINTTAB(5,4)"WELL DONE!"
3400 PRINTTAB(3,7)"WEE SHUGGY HAS";TAB
(6,9)"ESCAPED"
3410 VDU31,0,21,17,2:FORI%=0TO19:VDU23
2,8,10,232,11:NEXT
3420 Y%=20:X%=10:U=-1:REPEAT
3430 IFY%>16ANDU Y%=Y%-1ELSEU=0:Y%=Y%+1
3440 PROCprint(2):IFY%=20U=-1
3450 T=TIME:REPEATUNTILTIME=T+3:PROCpr
int(0)
3460 UNTIL0
3470 END
```




```
3480 :
3490 DEFPROCcharacter
3500 VDU23,224,255,170,255,170,255,170
,0,0
3510 VDU23,225,255,129,255,129,255,129
,0,0
3520 VDU23,226,0,0,255,129,255,129,0,0
3530 VDU23,227,0,0,0,0,255,129,0,0
3540 VDU23,228,146,146,255,129,255,255
,0,0
3550 VDU23,229,16,138,73,82,60,16,16,16
3560 VDU23,230,255,255,191,173,44,8,8,0
3570 VDU23,231,60,36,60,16,28,16,28,16
3580 VDU23,232,255,66,255,8,8,255,66,2
55
3590 VDU23,233,24,24,24,255,255,24,24,
24
3600 VDU23,234,0,0,0,146,73,36,146,73
3610 VDU23,253,28,62,107,127,20,20,20,
54
3620 VDU23,254,0,0,28,62,107,127,20,54
3630 VDU23,235,255,254,252,248,240,224
,192,128,0
3640 VDU23,236,255,127,63,31,15,7,3,1
3650 ENVELOPE1,1,0,0,0,0,0,0,0,0,-2,-2
,126,0
3660 ENVELOPE2,1,0,0,0,0,0,0,0,0,-5,-5
,95,0
3670 ENVELOPE3,1,5,5,5,-5,-5,-5,15,15,
-9,-9,126,126
3680 ENVELOPE4,0,0,0,-1,1,1,1,0,1,0,25
4,120,128
3690 ENDPROC
3700 :
3710 ON ERROR OFF:MODE 7
3720 IF ERR=17 END
3730 REPORT:PRINT" at line ";ERL
3740 END
```

Tested on Basic I & II and O.S. 1.2

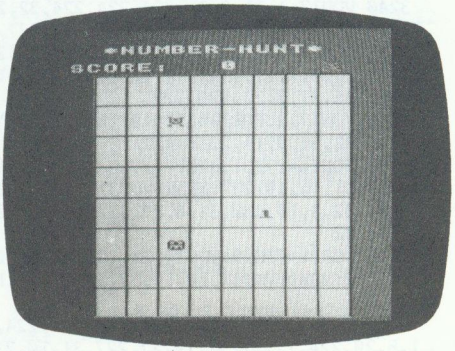
NUMBER HUNT (16k)

by Kevin Allen

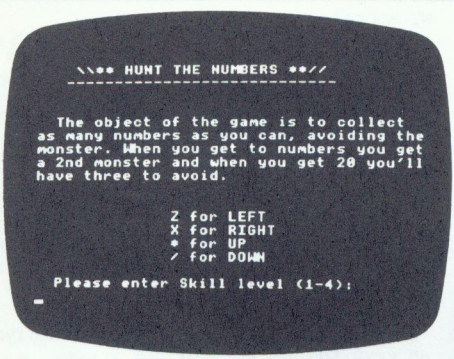
Hunt the Numbers is a simple but fast game written in Basic. It incorporates coloured graphics in mode 5 and at the fastest level is very quick and extremely challenging.

Your man is a number-eating beasty, or a kind of number-cruncher, I suppose, whose world consists of a nice bright yellow square. Within this, by the magic of computers, ever increasing numbers keep appearing as a temptation to your man to raise his calorific intake for the day. But as you move him to digest these numeric morsels, other nasty monsters will give chase all round the board with only the demise and consumption of your innocent glutton as their objective. These chaps are number-cruncher gulpers and they increase in quantity with your man's success. With higher levels of skill, selected at the beginning of the game, these aggressors become faster and more direct in their descent upon your ever-hungry number-cruncher which can result in a peculiarly loopy dance around the board's surface.

Controls to change the horizontal and vertical directions of movement are the usual 'Z','X', and '*' and '/' keys but it's important to note also that simultaneous use of a horizontal and a vertical control key will produce a diagonal motion, which is just as well because the 'nasty monsters' have no qualms about cutting corners when they're chasing you.




```
  10 REM PROGRAM NHUNT
  20 REM VERSION B0.1
  30 REM AUTHOR  K.ALLEN
  40 REM BEEBUG  NOV 1984
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 CLEAR:MODE7
 110 ON ERROR GOTO 330
 120 PROCinfo
 130 man%=3:score%=-50:Q%=0
 140 MODE 5
 150 PROCsetup
 160 PROCnumbers
 170 REPEAT
 180 PROCmonster1
 190 PROCman
 200 IF Q%>10:PROCmonster2:ELSE PROCde
lay(6)
 210 PROCman
 220 IF skill%>1:PROCmonster1:ELSE PRO
Cdelay(16)
 230 PROCman
 240 IF skill%>2 PROCmonster1:PROCman:
IF Q%>10:PROCmonster2:ELSE PROCdelay(8)
 250 PROCman
 260 IF skill%>3 PROCmonster1:IF Q%>10
:PROCmonster2:ELSE PROCdelay(8)
 270 PROCman
 280 IF skill%<4:PROCdelay(8)
 290 IF Q%>20:PROCmonster3
 300 UNTIL FALSE
 310 END
 320 :
 330 ON ERROR OFF:MODE 7:IF ERR<>17 TH
EN REPORT:PRINT "at line";ERL
 340 END
 350 :
1000 DEFPROCman
1010 IF Q%>20 AND (A=W AND B=Z) PROCde
ad:GOTO 100
1020 IF Q%>10 AND (A=F AND B=G) PROCde
ad:GOTO 100
1030 IF (A=M AND B=N) PROCdead:GOTO100
1040 IF NOT(A=K% AND B=L%) THEN GOTO 1
050 ELSE VDU 7:PROCnumbers
1050 IF INKEY-98 A=A-3:IF A<5 A=5
1060 IF INKEY-67 A=A+3:IF A>26:A=26
1070 IF INKEY-73 B=B+4:IF B>31:B=31
1080 IF INKEY-105 B=B-4:IF B<3:B=3
1090 IF a=A AND b=B:GOTO 1130
1100 *FX21,4
1110 SOUND 0,-15,4,3
```




```
 1120 GCOL 0,2:MOVE 40*a,26*b:VDU 241
 1130 GCOL 0,1:MOVE 40*A,26*B:VDU 241
 1140 a=A:b=B
 1150 ENDPROC
 1160 :
 1170 DEFPROCdelay(D%)
 1180 now=TIME:REPEAT UNTIL TIME>now+D%
 1190 ENDPROC
 1200 :
 1210 DEFPROCinfo
 1220 CLS
 1230 VDU 19,1,6;0;
 1240 FORA=1 TO2:PRINT TAB(5,A)CHR$141;
CHR$134"** HUNT THE NUMBERS **":NEXT:PR
INTTAB(3,3)CHR$129;STRING$(28,"-");TAB(
12,4)CHR$130"by K.Allen"
 1250 PRINT TAB(0,7)CHR$131"The object
of the game is to collect as"CHR$131"ma
ny numbers as you can, avoiding the"'CH
R$131"monster. When you get two digit n
umbers";
 1255 PRINTCHR$131"you get a 2nd monste
r and when you get"'CHR$131"up to 20 yo
u'll have three to avoid."
 1260 PRINT TAB(13,14)CHR$133"Z for LEF
T"
 1270 PRINT TAB(13,15)CHR$133"X for RIG
HT"
 1280 PRINT TAB(13,16)CHR$133"* for UP"

 1290 PRINT TAB(13,17)CHR$133"/ for DOW
N"
 1300 PRINT TAB(2,19)CHR$134"Please ent
er Skill level (1-4):"
 1310 REPEAT:skill$=GET$:UNTIL INSTR("1
234",skill$)>0
 1320 skill%=VAL(skill$)
 1330 PRINT TAB(33,19);skill%
 1340 PRINT TAB(4,22)CHR$129 CHR$136"=>
PRESS SPACE BAR TO START<="TAB(6,23) ST
RING$(28,"-")
 1350 REPEAT UNTIL INKEY(-99)
 1360 ENDPROC
 1370 :
 1380 DEFPROCsetup
 1390 ENVELOPE 3,2,8,4,8,2,2,2,126,0,0,
-126,126,126
 1400 A=5:B=3:a=5:b=3:f=26:F=26:g=31:G=
31:N=7:n=7:M=26:m=26:W=5:w=5:Z=31:z=31:
K%=0:L%=0
 1410 VDU 24,160;16;1120;848;
 1420 VDU 23,240,102,153,153,255,129,16
5,165,126
 1430 VDU 23,241,195,126,90,90,102,60,6
6,195
 1440 VDU 5:VDU 18,0,130:VDU 12
 1450 VDU 26:VDU 19,0,4;0;:VDU 4
 1460 VDU 19,2,3;0;:VDU 19,1,2;0;
 1470 VDU 19,3,1;0;:VDU 18,0,2
 1480 start=TRUE:start2=TRUE:start3=TRUE
 1490 COLOUR2
 1500 PRINT TAB(1,2);"*HUNT THE NUMBERS
*";TAB(1,4);"SCORE:";TAB(15,4);
 1510 IF man%>1:COLOUR 1:FOR lives=2 TO
man%:VDU 9,241:NEXT
 1520 VDU 5
 1530 GCOL 0,3
 1540 FOR X=160 TO 1120 STEP 120:MOVE X
16:DRAW X,848:NEXT
 1550 FOR Y=16 TO 848 STEP 104:MOVE 160
Y:DRAW 1120,Y:NEXT
 1560 ENDPROC
 1570 :
 1580 DEFPROCmonster1
 1590 IF M=A:GOTO 1600 ELSE M=M+3*(-1-2
*(M<A)):IF M<5 M=5 ELSE IF M>26:M=26
 1600 IF N=B:GOTO 1610 ELSE N=N+4*(-1-2
*(N<B)):IF N<3 N=3 ELSE IF N>31:N=31
 1610 IF (M=F AND N=G) OR (M=W AND N=Z)
:N=n:M=m:ENDPROC
 1620 GCOL 3,1
 1630 MOVE 40*m,26*n:VDU 240:IF start=T
RUE:start=FALSE:GOTO 1630
 1640 MOVE 40*M,26*N:VDU 240
 1650 n=N:m=M
 1660 ENDPROC
 1670 :
 1680 DEFPROCmonster2
 1690 IF F=A:GOTO 1700 ELSE F=F+3*(-1-2
*(F<A)):IF F<5 F=5 ELSE IF F>26:F=26
 1700 IF G=B:GOTO 1710 ELSE G=G+4*(-1-2
*(G<B)):IF G<3 G=3 ELSE IF G>31:G=31
 1710 IF (M=F AND N=G) OR (W=F AND Z=G)
:G=g:F=f:ENDPROC
 1720 GCOL 3,1
 1730 MOVE 40*f,26*g:VDU 240:IF start2=
TRUE:start2=FALSE:GOTO 1730
 1740 MOVE 40*F,26*G:VDU 240
 1750 f=F:g=G
 1760 ENDPROC
 1770 :
 1780 DEFPROCmonster3
 1790 IF W=A GOTO 1800 ELSE W=W+3*(-1-2
*(W<A)):IF W<5 W=5 ELSE IF W>26:W=26
```

```
 1800 IF Z=B GOTO 1810 ELSE Z=Z+4*(-1-2
*(Z<B)):IF Z<3 Z=3 ELSE IF Z>31:Z=31
 1810 IF (W=F AND Z=G) OR (W=M AND Z=N)
:W=w:Z=z:ENDPROC
 1820 GCOL 3,1
 1830 MOVE 40*w,26*z:VDU 240:IF start3=
TRUE:start3=FALSE:GOTO 1830
 1840 MOVE 40*W,26*Z:VDU 240
 1850 z=Z:w=W
 1860 ENDPROC
 1870 :
 1880 DEFPROCnumbers
 1890 MOVE 40*(K%-14.7-(Q%>9)),26*L%:GC
OL 0,2:PRINT Q%
 1900 k%=3*RND(7)+2:l%=4*RND(7)-1:IF k%
=K% AND l%=L%:GOTO1900:ELSE K%=k%:L%=l%
 1910 GCOL 0,0:MOVE 40*(K%-14.7-(Q%>8))
,26*L%:Q%=Q%+1:PRINT Q%
 1920 S=score%:score%=score%+50
 1930 GCOL 0,0:score%=INT(score%):MOVE
64,896:PRINT S:MOVE 64,896:GCOL 0,2:PRI
NT score%
 1940 ENDPROC
 1950 :
 1960 DEFPROCdead
 1970 SOUND 1,3,80,24
 1980 VDU 19,0,0;0;19,1,7;0;
 1990 TIME=0:REPEAT UNTIL TIME=250
 2000 man%=man%-1:IF man%>0 GCOL 0,128:
CLS:PROCsetup:Q%=Q%-1:score%=score%-50:
GOTO 160
 2010 VDU 4:PRINT TAB(2,10)SPC(21)"WOUL
D YOU LIKE ";SPC(23);"ANOTHER GAME?(Y/N
)"SPC(20)
 2020 *FX15
 2030 REPEAT KEY$=GET$:UNTIL INSTR("YyN
n",KEY$)>0
 2040 IF KEY$="N" VDU22,7:END ELSE CLS:
AGAIN=TRUE:ENDPROC
```

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. Please note that the BEEBUG Reference Card and BEEBUG supplements are not supplied with back issues.

SUBSCRIPTIONS

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

MEMBERSHIP COSTS:

U.K.
£6.40 for 6 months (5 issues)
£11.90 for 1 year (10 issues)

Eire and Europe
Membership £18 for 1 year.

Middle East £21
Americas and Africa £23
Elsewhere £25
Payment in Sterling preferred.

Subscriptions & Software Address	Subscriptions and Software Help Line
BEEBUG PO BOX 109 High Wycombe Bucks	St.Albans (0727) 60263 Manned Mon-Fri 9am-5pm

PROGRAMS AND ARTICLES

All programs and articles used are paid for at around £25 per page, but please give us warning of anything substantial 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 machine readable form using "Wordwise", "Minitext Editor" or other means. If you use cassette, please include a backup copy at 300 baud.

HINTS

There are prizes of £5 and £10 for the best hints each month.

Please send all editorial material to the editorial address below. 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

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

BEEBUG Publications Ltd (c) April 1984.

BEEBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Editor: Mike Williams.
Assistant Editor: Geoff Bains, Production Editor: Phyllida Vanstone.
Technical Assistants: David Fell and Alan Webster.
Managing Editor: Lee Calcraft.
Thanks are due to Sheridan Williams, Adrian Calcraft, John Yale, and Tim Powys-Lybbe for assistance with this issue.

NEW BEEBUG BINDERS

We have produced a new hard-backed binder for the BEEBUG magazine. These binders are dark blue in colour with "BEEBUG" in gold lettering on the spine and allow for the whole of one volume of the magazine to be stored as a single reference book.



The new binders now have a slightly larger capacity to accommodate 10 thicker BEEBUG magazines, and are supplied with 12 wires. This enables the index and the latest copy of the supplement to be included within the binder if required. Individual issues may still be easily added and removed, allowing for the lastest volume to be filed as it arrives.

The price of the new BEEBUG binder is £3.90 including VAT. Please add 50p post and packing for delivery within the UK. Overseas members please send the same amount, this will cover the extra postage but not VAT.

BEEBUGSOFT, PO BOX 109, High Wycombe, Bucks, HP10 8HQ.

BEEBUG NEW ROM OFFER

1.2 OPERATING SYSTEM

A special arrangement has been agreed between Acorn and BEEBUG whereby BEEBUG members may obtain the 1.2 operating system in ROM at the price of £5.85 including VAT and post and packing. The ROM will be supplied with fitting instructions to enable members to install it in their machine. If the computer does not subsequently operate correctly, members may take their machine to an Acorn dealer for the upgrade to be tested, which will be done at a charge of £6.00 plus VAT. This charge will be waived if the ROM is found to have been defective. If the computer has been damaged during the installation processs, the dealer will make a repair charge.

ADDRESS FOR 1.2 OS:
ROM Offer, BEEBUG. PO Box 109, High Wycombe, Bucks, HP10 8HQ

THE BEEBUG MAGAZINE ON DISC AND CASSETTE

The programs featured each month in the BEEBUG magazine are now available to members on disc and cassette.

Each month we will produce a disc and cassette containing all of the programs included in that month's issue of BEEBUG. Both the disc and the cassette will display a full menu allowing the selection of individual programs and the disc will incorporate a special program allowing it to be read by both 40 and 80 track disc drives. Details of the programs included in this month's magazine cassette and disc are given below.

Magazine cassettes are priced at £3.00 and discs at £4.75.
SEE BELOW FOR FULL ORDERING INFORMATION.

This Month's Programs Include:

Superb action game Wee Shuggy, a useful and comprehensive Cross-Reference Lister for Basic programmers, BEEBUG Workshop Text formating routine, Auto-Keyword Generator for Basic, full extended version of the Domestic Accounts Home Budgeting program, improved utility for printing function key labels, another challenging game Number Hunt, and as an extra a copy of the BEEBUG Pack program for squeezing unnecessary spaces and comments out of Basic programs.

MAGAZINE DISC/CASSETTE SUBSCRIPTION

Subscription to the magazine cassette and disc is also available to members and offers the added advantage of regularly receiving the programs at the same time as the magazine, but under separate cover.

Subcription is offered either for a period of 6 months (5 issues) or 1 year (10 issues) and may be backdated if required. (The first magazine cassette available is Vol. 1 No. 10; the first disc available is Vol. 3 No. 1.)

MAGAZINE CASSETTE SUBSCRIPTION RATES

6 MONTHS	(5 issues)	UK £17.00 INC... Overseas £20.00 (No VAT payable)
1 YEAR	(10 issues)	UK £33.00 INC... Overseas £39.00 (No VAT payable)

MAGAZINE DISC SUBSCRIPTION RATES

6 MONTHS	(5 discs)	UK £25.50 INC... Overseas £30.00 (No VAT payable)
1 YEAR	(10 discs)	UK £50.00 INC... Overseas £56.00 (No VAT payable)

CASSETTE TO DISC SUBSCRIPTION TRANSFER

If you are currently subscribing to the BEEBUG magazine cassette and would prefer to receive the remainder of your subscription on disc, it is possible to transfer the subscription. Because of the difference between the cassette and disc prices, there will be an extra £1.70 to pay for each remaining issue of the subscription. Please calculate the amount due and enclose with your order.

ORDERING INFORMATION

Please send your order to the address below and include a sterling cheque. Postage is included in subscription rates but please add 50p for the first item and 30p for each subsequent item when ordering individual discs or cassettes in the UK. Overseas orders please send the same amount to include the extra post but not VAT.
SEND TO:

BEEBUGSOFT, PO BOX 109, HIGH WYCOMBE, BUCKS, HP10 8HQ

Printed in England by Staples Printers St Albans Limited at The Priory Press.

ISSN 0263-7561