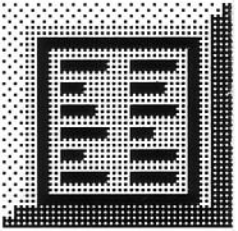
INDEX



INSTRUCTIONS

File the index page from every issue at the back of your Fast Access binder for a handy reference to your library of programs. The pages can be ordered by Icon or by issue, it's your choice.

80 TRACK DRIVES

Insert FAST ACCESS into your disk drive and type:
*CHANGE <RETURN>
The program will prompt you to insert a formatted 80 track disk when necessary. Single drive users will have to swap the disk several times.

TITLE	PROGRAM FILES	TEXT FILES
Astefix	ASTEFIX	T.ASTEFIX
Blast Out	EWE	T.BLAST
	HEAD	
Diary	D.DIARY	T.DIARY
	DINIT	
Disk Fixers	FIXERS	T.FIXERS
Filestore Assembler	ASM	T.ASM
French Tutor	FRENCH	T.FRENCH
Gallery	GALLERY	T.GALLERY
	GALSCR1	
	COMP	
Gameplan		T.GAMEPLN
KeyStrip Designer	FKEY	T.KEYSTRIP
	ELSTRIP	
	ACESTRP	
Laughter	LAUGH	
	ASM	
	VPlaugh	
Romlist	ROMLIST	T.ROMLIST
3D Rotation	ROTATE	T.ROTATE
	ROT2	
Sort Tutorial	TEACHIN	T.SORT
Trace	TRACE	
Variable Referencer	VREF	

ALL USERS

Make backup copies of both disks and keep the originals in a safe place with Write-Protect tabs on. Use only the copies, as many of the programs write to the disk, which will diminish the usefulness of the originals. For specific filing system information, please refer to the help file on disk.

NEW USERS

Don't Panic!. First find out whether you have 40 or 80 track drive(s) attached to your computer. Then go to your User guide or Welcome Manual and find out how to use the *COPY command. Next re-read the section above All Users, and then continue reading down from this point.

40 TRACK DRIVE SYSTEMS

FAST ACCESS is supplied on 40 track disks and will work on any 40 track BBC Micro system straight away. Remember to make a working copy before use.

80 TRACK DRIVES

If your filing system allows double-stepping, we recomend using the system's own command. As a general rule, built-in 40-to-80 track converters should be used where available; the documentation for your filing system or utility ROM will give full instructions.

ADRESSES

If for any reason your copy of Disk User will not work on your system then please carefully re-read the instructions given above. If you still experience problems then return it to: INFONET LTD, 5 River Park Estate, Billet Lane Berkhamsted, Herts HP4 1HL. TEL 0442 87661 to 4

Advertising enquiries to FAST ACCESS, Argus House, Boundary Way, Hemel Hempstead, HP2 7ST. TEL 0442 66551.

Editorial and technical enquiries to FAST ACCESS, 6C Belgic Square, Padholme Road, Peterborough PE1 IXF TEL 0733 53355.

Contributions should include full source code and instructions file on disk. Payments are extremely competitive.

Typesetting and artwork by Simon Fifield.

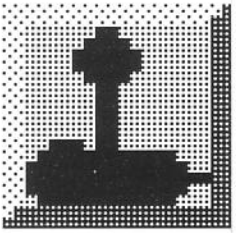
Printed by Loxleys, Sheffield.

Duplication by Direct Disk Supplies, Teddington, Middlesex.

RESOURCE FILES

TITLE	FILES	TYPE
Menu	FAMENU	BASIC
	GOSPR	Data file
	GOCODE	M/ code
Info Desk	IDESK	BASIC
Change	CHANGE	M/code
Help	T.HELP	Text file
Superior	FOOTY	Screen file
Soccer Ad	LOADIT	BASIC

BLAST OUT



GAME CONTROLS

Z.....MOVE BAT LEFT
?.....MOVE BAT RIGHT
SPACE.....FIRE LASER
Q.....QUIET
S.....SOUND

AUTHOR

EWEN SETTI

FILES

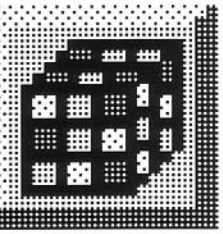
M.BLA.OUT'.....BASIC
'HEAD'.....BASIC
'EWE'.....BASIC
'T.BLAST'.....TEXT

This game is a modified version of the classic arcade game 'Breakout'. It has many options ranging from a small bat to screen wrap around. There are eight modes of play in all. The program provides full on screen instructions.

MODES OF PLAY

1 - Normal game
2 - Ball Wraproud
3 - Random screens
4 - Bat wraproud
5 - Mode 2 & 4 combined
6 - Small bat and alien
7 - Invisible blocks
8 - Quick flashing blocks

Unless in mode 6 after 1000 points, the bat will shrink to half the normal size!



ASTE-FIX

AUTHOR

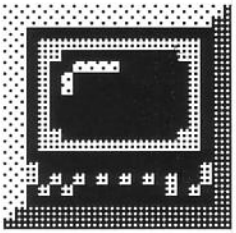
DES CATLIN

FILES

'ASTEFIX'.....BASIC
'T.ASTEFIX'.....TEXT

This brain teaser from the master of computer puzzles presents a real challenge. Lots of time and thought will be required to unscramble the colours in the asterix shape on the screen. Full instructions on solving the puzzle are presented on screen by the program.

DIARY



AUTHOR

JONATHON GIFFARD

FILES

'DINIT'.....BASIC
'D.DIARY.....BASIC
'T.DIARY'.....TEXT

To use this program you must prepare a blank disc. Onto this disc the files 'DINIT' and 'D.DIARY' must be copied. When this has been done the disc can be prepared to hold diary data. This is done by running the 'DINIT' program with the prepared disc in the drive. It sets up the blank data file on disc and allows 255 characters per day. Before the program starts creating the file, it will ask for which year the diary will be for. Enter the year and be prepared to wait as the data file takes up about 91K on a 40 track disc.

Now you can run the Diary program (D.DIARY). Type in theyear which you entered in the 'DINIT' program. You will then be presented with a menu.

```
1989 - This is the current
year's diary selected.

1.Edit diary..Current choice
2.January....Current month
3.End.....Returns to BASIC
```

By pressing <1>, the menu cycles through the different options available in the diary program. These are:

EDIT: Allows entries to be created and edited.

PRINT: Entries can be printed from this. A whole month can be printed or just one day. Printer output is identical to that on the screen.

DELETE: Entries are deleted byaltering a variable on the data file. The actual data isn't lost until it is over written.

Press <2> and you cycle through the months of the year. Stop at the one you want.

The year can be altered by pressing <Y>. Type in the year required. You can only alter the year to a corresponding data file eg you cannot enter 2020 if there is not a data file for it.

Once you have decided upon the options press <RETURN>. On the screen now will be the calender for the month you have picked from the menu. Enter the day on which you want to make an entry. This will happen whatever option you decide upon.

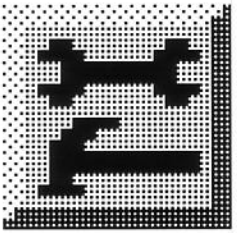
You will now be presented with a blank screen (unless you have made a previous entry, in which case the entry will be displayed) except for the day and space in the data entry.

Type in your entry for that day. The program automatically moves words onto the next line if they do not fit onto one line.

Once finished, pressing <CTRL>-<f0> will save the entry, <CTRL>-<f1> will print it but not save it and <CTRL>-<f2> will return to the menu without saving the entry. After each of the first two operations have been executed, the program returns to the menu.

The printing and deleting routines are easy to follow with prompts from the program. Once you have deleted the data it is gone and can't be recovered from the diary program.

DISK FIXERS



AUTHOR

DOV ROSNER

FILES

'FIXERS'.....BASIC
'T.FIXERS'.....TEXT

These programs are only compatible with single density, Acorn DFS compatible disks. Do not use them for double density disks or disks which have been upgraded to support more than 31 files on one disk surface. However, if you have a double density DFS, the programs are likely to work in single density mode.

The programs offered here are both useful in some circumstances. PARKFAR is designed to run from inside programs that you write or utility programs that you are able to modify. COPYCAT is for use with disks with a 'fixed' catalogue, or in other words, disks that you rarely save extra programs on.

ASSEMBLING

The programs mentioned above, ie PARKFAR, COPYCAT and CATBACK are written in assembly language. On the disk A you will find one BASIC program called 'FIXERS' which contains the source code for the programs mentioned in this article. Once chained directly or through the Fast Access menu, you will be asked to insert a disk with some free space on it to which the program will assemble and save the 3 object files. Please refer to the file 'FixerTXT' for more information on how to use the programs.

PARKFAR moves the head of the disk drive to the last track on the disk. It's a good idea to call PARKFAR after reading or writing to the disk drive, as well as at the beginning of the program. There are two ways to run the program, either *PARKFAR, which will load the program every time it is needed from disk, or *LOAD

PARKFAR to load it in memory, and thereafter with a CALL &A00 command.

The program always parks the disk drive head at the end of the currently selected drive number (by *DRIVE). If the format of the disk looks unknown to PARKFAR, it aborts and gives a Format! error.

COPYCAT accepts a drive number (between 0 and 3) as an optional parameter which must be separated from the command name by exactly one space. For example, *COPYCAT 2 will clone the catalogue on drive 2. Should this parameter be omitted, the current drive number is used.

Once loaded, COPYCAT waits for a key to be pressed. Hitting <ESCAPE> will abort it and any other key will continue. The pause after loading allows you to change disks in case COPYCAT is on a different disk.

CATBACK

This program makes a backup of the faulty disk to a newly formatted disk . The catalogue image is downloaded from the faulty disk to the destination disk to make it usable.

The program accepts two drive numbers and a number of tracks as parameters. This last parameter may be 40 or 80 but using 4 or 8 is equally acceptable. The parameters must come one after the other with no separating characters between them except for spaces. The first parameter is the drive number of the faulty disk, the second specifies the target disk and the third specifies the number of tracks that the faulty disk is supposed to have. For example,

*CATBACK 0 2 80 or

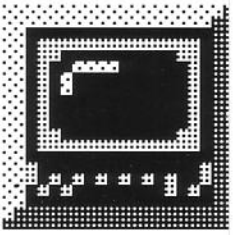
*CATBACK 028

will copy an 80 tracks disk from drive 0 to drive 2. Note that there MUST be a space after the command but not between the parameters.

CATBACK uses all available memory to copy the faulty disk. As a result any program or data in RAM is erased.

Note that you should copy only disks that have had their catalogue transferred to the end of the disk (with COPYCAT) before they developed the fault. command is displayed if you miss any parameters or the values are out of range. The program needs at least 5Kb of free RAM.

FILESTORE ASSEMBLER



AUTHOR

J. RAWLINSON

FILES

'ASM'.....BASIC
'T.ASM'.....TEXT

For most purposes, the assembler built into the BBC micro, is quite adequate. However, problems arise when it is necessary to assemble very big programs such as 16K ROM's. This is because the assembly language takes up so much memory that there is little left to put the machine code in. More extensive (and expensive) assemblers (eg, ADE+) overcome this problem by allowing assembly to and from disc filestore. The simple program described here will allow you to do this with the ordinary BBC assembler.

IN USE

To use the program, you should first list the program. In line 140 enter the value of the memory location at which the code is to be assembled as the value of 'BASE'. Lines 110 and 120 contain in 'PASS(1)' and 'PASS(2)' the value of the OPT parameter which is to be used for the two passes of the program. When the program is run, it will ask for the name of the file in which the assembly language is contained and then whether you want the machine code to be sent directly to a file. This latter option will only work if you have BASIC II and a disc drive. If you select assembly to disc, you will be asked for the name of the file in which the code is to be put. Assembly should then proceed in much the usual way.

LIMITATIONS

There are a few limitations on the use of the program. First, the file to be assembled must be pure assembly language. It must contain no BASIC or square brackets. If your program uses BASIC to set up variables or zero page locations, then the necessary procedure should be added onto the listing and called up before assembly begins. Similarly, the tricks which can be done by using a function as the parameter of the OPT directive should be possible but the necessary functions must be contained in the assembler program and not in the file which is assembled. Possibly, the only disadvantage of the technique described here, is that an OPT is done for every line in the file assembled. This gives a lot of unecessary screen output and slows assembly down.

The program works, by repeatedly assembling the contents of line 340. Initially, this is entered as a full line of spaces. However, just before the program reaches line 340, there is a call of the machine code routine FIX this gets a line from the assembly language file and copies it into line 340. By repeating this process, the complete file can be assembled. Dumping the machine code to disc, relies on the feature of Basic II that code can be generated at a different address to that at which it is to run. On the assemblers second pass, each time through the above loop, all the code that has been generated at the location O%, is dumped to file by PROCDUMP (line 1030) and then O% is reset. When this option is selected, the OPT values of the array PASS are automatically adjusted for offset assembly in lines 1150 and 1160.

By taking advantage of all the features of this program, it should be possible to assemble a program and have around 24K of labels. This is plenty for a 16K ROM.

FRENCH TUTOR

AUTHOR

PETER J LUBANSKI

FILES

'FRENCH'.....BASIC
'T.FRENCH'.....TEXT

This program was written as an aid to learning and fun for anyone studying French. There are two slightly novel features incorporated which, it is hoped, will help make learning French a little more interesting:

1) You may add your own words in a fairly easy way - in fact you may even (with a few minor changes) make your own tutor for another language. The essential instructions to do this are displayed when function key 0 is pressed.

2) You can also add simple phrases to the appropriate section (these should be carefully chosen for little or no variations in translation).

The main part of the programme selects words/phrases from various DATA statements in a fixed sequence. The first word/phrase is always in English and the next word/phrase (after a comma) is always the French translation (they must all be in lower case).

DATA statements may be added to give quite a large vocabulary before your computer runs out of memory.

If you wish you may also include data as to whether the word is male or female. This is done by including an upper case M in front of the French word for male or F for a female word eg:

DATA girl,Ffille,boy,Mgarcon,

(the words you add must be in lower case)

Of course, to add words/phrases you need to leave the

program and list it. Then simply find the section under which you wish to make additions - these are all headed with REM statements - then add a few lines as described. Note that you should not add any lines after the DATAXXX,XXX line in each section (this line informs the program that it has reached the end of a section).

You will notice that the words displayed are selected on a random basis. Approximately one third of all the words are displayed each time you select an option. PROCdata selects one of three words, displays it and checks any answers - each time it is called.

Selecting the menu after starting is done by pressing <ESCAPE>. PROCerror deals with any system errors and uses those caused by ESCAPE to restart PROCmenu, all other erros should give their full message.

Use is made of program defined *KEY facilities to allow mode changes when requesting the "instructions for adding words" option at startup. This arrangement is used because mode changes within a procedure are not normally allowed.

A collection of words and phrases have been included to start with.

LET US KNOW !

We would like to know what you think of your new disk magazine, so let us know about it.

- Is there a program which you think is excellent.
- Do you think that there should be more of a particular type of program.
- Do you need more information.
- Are having compatibility problems with a program.
- Do you have something to say about Fast Access in general.
- Do you have any bug fixes or improvements to published programs.

Write To:

FAST ACCESS

6C Belgic Square, Padholme Road,

Peterborough.

PE1 1XF.

GAMEPLAN

AUTHOR

PETER SCOTT

FILES

'T.GAMEPLN'.....TEXT

On 'Thunderstruck II', I worked out that I had a maximum of 48 bytes to store each screen. That means that the data for any particular screen must be highly compacted if detail isn't to suffer.

I used graphics blocks of 12 by 16 pixels big, meaning that the screen was 13 by 8 of these blocks big (19.5 by 16 characters big). There were 128 of these blocks, and blocks 128-255 were taken as blocks 0-127 except they'd be reversed (using the technique I've explained before).

Each screen had also to store what colours were used, up to four sprites (their horizontal and vertical position, what sort they were and what direction they would move in) and any puzzles on it. Using 1 byte for each block, and each piece of this information would take at least 128 bytes per screen : far too much.

The answer was to store as much information in a byte as possible. There were only four different colour combinations, and I simply held a look-up table of these, and a number 0-3 in part of a byte. Each byte was split into sections, and as much information crammed into it as possible.

The actual screen itself was another matter. Instead of holding a simple table of blocks, I held a table of platforms. Each platform had a horizontal start position, vertical start position, length (in blocks) , direction, sort of block used and a special bit which, if set, would cause a line of as-

cending-numbered blocks to be printed (eg starting with block 4,then 5,6,7 etc.)

This sounds complex, and is rather awkward to program. If you've seen 'Thunderstruck II', you'll know the screens can be very detailed, and taking only 48 bytes per screen is a very high compaction rate!

Also, in 'Thunderstruck II' and 'Omega Orb', there were repeated screens : the same set of data used several times in different places. This expanded the map for the same number of screens. I had to be careful not to do this too frequently, as seeing the same screen over and over again is boring for the player.

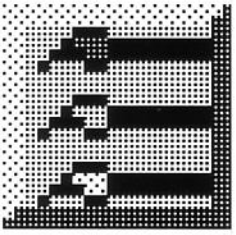
Below is a complete Breakdown of how a 'Thunderstruck II' screen was stored.

The Full GAME-PLAN Article is on Disk.

STRUCTURE BREAKDOWN

```
Byte :0 AND 3    (no. 0-3)    : colour combination used
        AND 12  DIV 4   (0-3) : number of sprites on screen
        AND 224 DIV 16 (0-15)        : number of platforms
        (IF NO SPRITES ON SCREEN, SKIP SPRITE DATA)
Byte :  AND 15   (0-15)        : sprite X position ( x3 for
actual posn.)
        AND 240 DIV 16 (0-15)        : sprite Y position
Byte :  AND 31   (0-31)        : sprite type (32 of them)
        AND 224  (0-7)         : sprite direction number
        (0=stationary, 1=up, 2=down, 3=left, 4=right)
Byte :  AND 15   (0-15)        : platform x position
        (0-12 used)
        (if =13, toggle printing on/off of changing platform
sort)
        (if =14, change main block type to the data held in the
next part of the byte multiplied by 8)
        AND112 DIV16 (0-7)   : platform y ( x2 in use)
        AND128 DIV128 (0/1) : part of type of block
Byte :  AND 31 (0-31)          : platform length & direction
        ( 0-12 = horizontal)
        (12-19 = vertical   )
        (20-25 = diag. down/right)
        (26-31 = diag. up/left)
        AND224 (0-7) + LAST BIT OF PREVIOUS BYTE (=0-15)
                                     : offset to main block type
        (added to main number to make actual block printed 0-
255)
```

3D ROTATION



AUTHOR

CARLOS RONDAO

FILES

'ROTATE'.....BASIC
'ROT2'.....BASIC
'T.ROTATE'.....TEXT

This program rotates smoothly and quickly, any prism or pyramid up to 30 sides in MODE 4. The technique used is a little unusual and takes up almost all the available memory but as you will see the effect of rotation is perfect. The program works by drawing the images in BASIC and storing them in memory successively using a small machine code routine. Then another routine ('RL' or 'RR') puts the images back on screen in rapid succession, creating the effect of rotation.

PROGRAM 1

This is the machine code program and consists of three important routines:

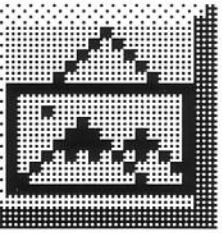
FOTO - Stores in memory the prism or pyramid.

RL - Rotates left

RR - Rotates right

CONTROLS

Q.....SPEED UP
W.....SLOW DOWN
C.....CHANGE COLOUR
SPACE.....BACK TO THE START
X.....CHANGE THE SENSE OF ROTATION



GALLERY

ARTIST

STEVEN NISBET.....UNICORN

FILES

'GALLERY'.....BASIC
'GALSCR1'.....COMPRESSED
'COMP'.....MACHINE CODE
'T.GALLERY'.....TEXT

In this issue's Gallery we have an excellent painting by Steven 'Hawk' Nisbet. He paints using the Quest Mouse and his own custom art program. The quality of his artwork is second to none as you will see if you select the Gallery icon from the menu screen on disk B.

The file 'GALSCR1' on disk is a compressed screen file. To display the screen a program called 'comp' must be *RUN. You can then use *PLD to load the compressed file. Also *PSV can be used to save a screen in compressed format. Examine the file GALLERY to see this in action.

If you have drawn a picture which you would like others to see then send it in to us on disk. If we think it is good enough we will include it in our Gallery and give a reward. So if you want to be famous (and a little richer) then start painting. Details of how the picture was put together and full name and address plus telephone number would be appreciated. Ensure that your disk is clearly marked as well. Send your disks to our editorial offices (the address is at the front of the magazine).

SORT TUTORIAL

AUTHOR

MOSTYN HELLARD

FILES

'TEACHIN'.....BASIC
'T.SORT'.....TEXT

The computer has become popular for many reasons, one of them must be it's ability to store information and then recall and present it to the user in the way that he or she desires. Very often this means completely re-ordering the information into a more useful format. This process is called 'sorting' and it is the subject of the Fast Access Interactive Tutorial number one.

In order to run the tutorial you will need to have a paper and pen handy because you will be playing an active part in the learning process. The tutorial is designed for those who are familiar with BASIC, you will need to understand simple data structures such as integer, real and string variables together with arrays. It does not however, assume any expertise with more advanced techniques such as data processing, indeed it sets out to teach it!

THE STRING SORT

The string sort is only slightly more complex than the integer. The only real difference involves the forcing of each data item into a state were all of it's characters are lower case (or upper case it's up to you). To do this we need another two temporary storage variables, in this case we will call them "temp1$" and "temp2$". The routine to process our strings looks at each character in the string in turn. If the ASCII code is between 65 (&41) and 90 (&5A) then it must be a capital. In which case it replaces that

character with the lower case brother. This is done by adding 32 to it's ASCII code.

The procedure would be called with a command such as;

60 PROClower(array$(5))

The result of this process is returned in the variable; "var$". So there you have it, a routine that transforms strings from mixed case to lower case only.

The full text for this Tutorial is on Disk.

EXAMPLE LISTING

```
5000 DEF PROClower(in$)
5010 var$=""
5020 L%=LEN(in$)
5030 FOR C%=1 TO L%
5040 C$=MID$(in$,C%,1)
5050 CH%=ASC(C$)
5060 IF CH%>64 AND CH%<91 THEN
var$=var$+CHR$((CH%+32)) ELSE var$=var$+C$
5070 NEXT
5080 ENDPROC
```

Free Disks !

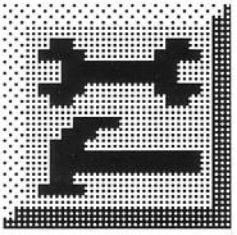
All you have to do is persuade a friend or colleague to subscribe and we'll reward you with £15 worth of ***SpriteSytem and Biochip*** the super all-in-one sprite creation package. *Biochip* is a great game created using *SpriteSystem*. This two disk package is fully compatible with all the BBC range, including Model B/ B+ Master and Electron.

Just get your friend to pop his or her application in the post with a cheque for £42.00 payable to A.S.P. Access/Mastercard & Barclaycard/ VISA orders also taken, so include all the necessary details. Make sure he or she includes your name, address, subscriber number and whether 40 or 80 Track disks are required.

To ensure you get your **Free Gift!** all applications should be sent to the following address:

Fast Access Free Disk Offer!
C/O Mark Webb, ASP ArgusHouse, Boundary Way, Hemel Hempstead, Herts.HP2 7BH

VARIABLE REFERENCER



AUTHOR

D.TAYLOR

FILES

'VREF'.....BASIC

To run the cross referencer either CHAIN "VREF" or select the Var-Ref icon from the menu screen. The program assembles machine code between &900 and &BFF in memory. When this is complete you can then load the program to be cross referenced. Simply type CALL &900 and the VREF command will be available. The command takes the form of:

```
•VREF A I R S F P
```

The parameters AIRSFP refer to Arrays, Integers, Real, Strings, Functions and Procedures respectively. You can cross reference just one, two or even all types at once. For example try:

LOAD "FAMenu"

VDU14

*VREF A I R S F P

The commands work very fast and produce clear lists of variable, procedure and function names with the lines that they occur on. Typing <CTRL>-<B> before issueing the *VREF command will send the output to a printer (<CTRL>-<C> switches back to screen only).

Note that it is not possible to cross reference upper case labels in assembler code as VREF can not distinguish these from the assembler mnemonics.

TRACE

AUTHOR

D. TAYLOR

FILES

'TRACE'.....BASIC

HOW TO USE TRACE

Running Trace is simple, just select the trace icon from the disc menu. When the program has finished assembling the machine code the program lies between &900 and &A00 in memory. You can now load any BASIC program and start the computer tracing the line numbers. For example, enter the following to trace the Fast Access Menu program.

LOAD "FAMenu" <RETURN>

*TRACE <RETURN>

RUN <RETURN>

Each program line is displayed in the top left hand corner of the screen as it is executed. It does not affect any VDU variables such as text and graphics windows.

The speed of the trace can be varied by pressing <CTRL> and a number (0-9) at the same time. The current speed is displayed to the right of the line numbers. Speed 9 is fastest (at 2MHz). The slowest is 0 which is one line per second.

WANTED

If you have written any useful, good quality programs for the BBC Micro why not send it to us for publication in Fast Access.

Send it to:

FAST ACCESS
6C Belgic Square,
Padhome Road,
Peterborough.
PE1 1XF.