

BBC BASIC



REFERENCE
MANUAL



BBC BASIC



PART NO 0410, 006
ISSUE NO 1
JULY 1985

© Copyright Acorn Computers Limited 1985

Neither the whole or any part of the information contained in, or the product described in, this manual may be reproduced in any material form except with the prior written approval of Acorn Computers Limited (Acorn Computers).

The product described in this manual and products for use with it, are subject to continuous developments and improvement. All information of a technical nature and particulars of the product and its use (including the information in this manual) are given by Acorn Computers in good faith.

In case of difficulty please contact your supplier. Deficiencies in software and documentation should be notified in writing, using the Acorn Scientific Fault Report Form to the following address:

Sales Department
Scientific Division
Acorn Computers Ltd
Fulbourn Road
Cherry Hinton
Cambridge
CB1 4JN

All maintenance and service on the product must be carried out by Acorn Computers' authorised agents. Acorn Computers can accept no liability whatsoever for any loss or damage caused by service or maintenance by unauthorised personnel. This manual is intended only to assist the reader in the use of the product, and therefore Acorn Computers shall not be liable for any loss or damage whatsoever arising from the use of any information or particulars in, or any error or omission in, this manual, or any incorrect use of the product.

Published by Acorn Computers Limited,
Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN.

Within this publication the term BBC is used as an abbreviation for the British Broadcasting Corporation.

NOTE: A User Registration Card is supplied with the hardware. It is in your interest to complete and return the card. Please notify Acorn Scientific at the above address if this card is missing.

ISBN 0 907876 37 4 Acorn Scientific

Contents

1	Introduction	1
1.1	The BASIC language	1
1.2	32000 BBC BASIC	1
2	A description of 32000 BBC BASIC	3
2.1	Language reference	3
2.2	Restrictions and variations from BASIC I	3
2.2.1	The assembler	3
2.2.2	Calling MOS routines	3
2.2.3	LIST IF	5
2.2.4	LIST formatting	5
2.2.5	COLOR	6
2.2.6	Error handling	6
2.2.7	Array formal parameters	7
2.2.8	The value of PAGE	7
2.2.9	Precision of printing	7
2.3	Extensions to BASIC I	8
2.3.1	The EXT# = statement	8
2.3.2	The OPENUP function	8
2.3.3	The OSLI statement	8
2.3.4	The ‘ ’ delimiter in VDU	9
2.3.5	The ON ... PROC statement	9
2.3.6	‘Unlimited’ nesting of structures	9
2.3.7	The HELP command	10
2.3.8	The QUIT command	10
2.3.9	The EDIT command	10
3	Preparing Programs	11
3.1	The EDIT command	11
3.1.1	Getting started	11
3.1.2	The screen	12
3.1.3	Locating text	12
3.1.4	Manipulating text	12
4	Errors and debugging	15
4.1	Errors in alphabetical order	16
4.2	Errors in numerical order	24
4.2.1	Fatal errors and command errors	24
4.2.2	Non-fatal errors	25



1 Introduction

This document describes 32000 BBC BASIC in terms of its differences from previous versions of BBC BASIC. Unlike other *BBC Microcomputer System User Guides*, it does not form the basis of a tutorial, nor does it give detailed information about the use of all BASIC functions.

32000 BASIC refers to the implementation of BBC BASIC on Acorn Cambridge Series computers.

1.1 The BASIC language

BASIC is a general purpose programming language which has been implemented on most computers, and especially microcomputers. It is interactive, which means that programs may be typed into the computer, executed and corrected without the user having to use a text editor or a compiler. In addition, the values of a program's variables may be examined and altered to see the effect that this would have on the program.

BASIC is usually interpreted; instead of the BASIC program being converted into machine code instructions (as happens with the Acorn 32000 Pascal and C, for example), the BASIC interpreter examines each BASIC command as it is encountered and performs the required task.

BASIC programs are edited using numbered program lines; a simple line editor is incorporated into 32000 BASIC as well as a more sophisticated screen-based editor. In addition, a text editor may be used to create files which may be inserted using the *EXEC command.

1.2 32000 BBC BASIC

This version of BBC BASIC is an enhanced version of previous versions that operate with the BBC Microcomputer. The version described in the *BBC Microcomputer System User Guide*, BASIC I, will be treated as the standard in this document. Some readers may already be familiar with earlier versions of BBC BASIC, for example BASIC II, as described in the *Acorn Electron User Guide* and later versions of the *BBC Microcomputer System User Guide*. The BASIC II extensions to BASIC I have been incorporated into 32000 BASIC.

Extensions and improvements provided by 32000 BASIC include a built-in screen editor, faster execution (especially in real arithmetic), extensions to statements (e.g. the ON ... PROC statement), and better list formatting when using LISTO, and of course, considerably more memory.

The only feature which is present in the version of BBC BASIC provided for use with the 6502 second processor (6502 BBC BASIC) is the 6502 assembler. A 32000 assembler (Asm32) is provided with Acorn Cambridge Series computers. However, the BBC operating system routine addresses (e.g. OSWRCH at &FFEE) may still be used with the CALL statement to access the useful MOS routines. A 32000 assembler (Asm32) is provided with the 32000 system.

Unlike all other systems software provided with the equipment, 32000 BASIC does not rely on the Panos operating system. Instead it runs directly on the small firmware kernel, Pandora. This means that it operates in the traditional environment of * commands etc. familiar to BBC Microcomputer users. In this way, compatibility with existing BBC BASIC programs is facilitated. For details of how to install 32000 BASIC from the distribution disc, see the User Guide supplied with your system.

2 A description of 32000 BBC BASIC

2.1 Language reference

The language reference followed for 32000 BASIC is BASIC I as described in the *BBC Microcomputer System User Guide*, which is supplied with every BBC Microcomputer.

2.2 Restrictions and variations from BASIC I

This section describes features of BASIC I which are not present in 32000 BASIC, and the differences in the implementation of certain facilities.

2.2.1 The assembler

The 32000 system is supplied with the Asm32 assembler. This may be used to create object files which can be loaded and called by BBC BASIC programs. The 32000 Assembler is described in the *Acorn 32000 Assembler Reference Manual*.

2.2.2 Calling MOS routines

The CALL statement and USR functions may be used to call operating system routines. Where the address called is in the range &FFCE (OSFIND) to &FFF7 (OSCLI), BASIC emulates the appropriate function. As usual, the integer variables A%, X% and Y% are used to pass parameters to the calls. The difference occurs when X% and Y% contain the address of a parameter block, for example when calling OSWORD. The usual way of achieving this is:

```
1000 DIM parBlk 6
1010 X%=parBlk MOD &100
1020 Y%=parBlk DIV &100
1030 A%=1
1040 CALL &FFF1
```

This is incompatible with 32000 BASIC, which requires X% to hold the complete address of the parameter block, not just the lower eight bits of it. To write the above routine in a way which is compatible with all versions of BASIC use:

```

1000 DIM parBlk 6
1010 X%=parBlk
1020 Y%=parBlk DIV &100 : REM Not used by 32000 BASIC
1030 A%=1
1040 CALL &FFF1

```

As mentioned in the comment, line 1020 is not strictly necessary, but is included to maintain compatibility with 8-bit versions of BBC BASIC.

CALL and USR with added parameter list

An example is given here to illustrate how 32000 BASIC implements these functions with an added parameter list. The statement

```
CALL locn% A,B$,C%
```

calls a location held at locn%. On entering this address, R1 (CPU register 1) holds the address of the parameter block, and R2 holds the number of parameters (in this case 3).

32000 BASIC automatically sets R1 and R2 before executing the code at locn%. Each entry in the parameter block consists of a doubleword specifying the address of the parameter followed by one byte specifying its type. The table below would be at the address held in R1.

doubleword	address of parameter 1
byte	type of parameter 1
doubleword	address of parameter 2
byte	type of parameter 2
:	:
:	:
doubleword	address of parameter n
byte	type of parameter n

Parameter types can be in the range 0 to 4. They have the following meanings:

type 0	parameter is a single byte, e.g. ?F%
type 1	parameter is an integer, e.g. C%
type 2	parameter is a real, e.g. A

type 3	parameter is a string, e.g. B\$
type 4	parameter is a byte string, e.g. \$A%

2.2.3 LIST IF

An extra part has been added to the syntax of the LIST command. It may be followed by the keyword IF, which is in turn followed by a string of characters. Only lines in the program which contain this string will be listed. For example

```
>LIST IF DEF      (list all lines with keyword DEF)
>LIST 100, IF =   (list all lines from 100 with the = sign)
```

When pseudo-variables such as LOMEM and TIME appear in the IF part, only lines using them as functions (as opposed to statements) will be found. For example:

```
>LIST IF TIME=100
```

will list lines such as

```
1230 UNTIL TIME=100
```

where TIME is a function, but not

```
3250 TIME=100
```

where TIME = is being used as a statement. This is due to pseudo-variables having two tokens each, only one of which may be found using LIST IF.

2.2.4 LIST formatting

The LISTO command now causes the FOR ... NEXT and REPEAT ... UNTIL structures to be listed in a more obvious manner than in previous versions of BBC BASIC. That is, UNTILs line up vertically with their corresponding REPEATs, and NEXTs are listed directly underneath FORs (assuming that only one loop is opened and closed on any one line). Compare the 'program' below listed firstly in BASIC 1 using LISTO 7:

```

1000 REPEAT
1010   FOR
1020     REM
1030     NEXT
1040   UNTIL
1050 REM

```

and then in 32000 BASIC:

```

1000 REPEAT
1010   FOR
1020     REM
1030     NEXT
1040   UNTIL
1050 REM

```

Trailing spaces in lines (i.e. spaces immediately before the end of the line) are always stripped, and leading spaces (i.e. spaces immediately after the line number) are stripped if a non-zero LISTO is set. The implication of this is that blank lines may not be entered if a non-zero LISTO is set.

2.2.5 COLOR

32000 BASIC accepts the alternative spelling of 'COLOR' for the 'COLOUR' statement. The word is always listed as 'COLOUR' by English versions of 32000 BASIC.

2.2.6 Error handling

When an error occurs while executing a program line, the line is listed in inverse video up until the point where the error was detected, then in normal video until the end of the line. This makes it easier to pin-point the source of the error in long program lines. The line may be edited by typing the statement REPORT, which prints it wholly in normal video.

The error messages printed by 32000 BASIC are different (longer and clearer) than the messages printed by earlier versions of BBC BASIC. The error numbers are the same for equivalent errors.

2.2.7 Array formal parameters

It is no longer possible to use array elements as formal parameters to procedures and functions. For example, the definition

```
1000 DEF PROCfred(n,a(n))
```

would be acceptable under previous incarnations of BBC BASIC. However, with 32000 BASIC, the address of formal parameter a(n) would only be evaluated on the first invocation of the procedure, rather than every time it is called.

2.2.8 The value of PAGE

The pseudo-variable PAGE is no longer confined to a 256 byte boundary. When 32000 BASIC is first called, PAGE is set to just above the interpreter itself (address 47BD approximately). PAGE should not be set lower than this, otherwise the BASIC program being entered will overwrite the interpreter.

LOMEM and TOP have the same meanings as under other versions of BASIC, i.e. they hold the address of the start of variable space and end of the program respectively. The HIMEM

pseudo-variable holds the address of the start of BASIC's stack which grows down in memory as FOR, REPEAT and GOSUB statements are executed, and shrinks again when NEXT, UNTIL and RETURN statements are executed. HIMEM should not be set to above its initial value which is the lowest memory location not used by Pandora.

2.2.9 Precision of printing

As in BASIC II, it is possible to print numbers to 10 digits of precision; the limit under BASIC I was nine digits. For compatibility (i.e. so that by default only nine significant digits are printed), the value of the @% variable is set to &90A when BASIC is first called. To gain the extra digit it should be set to &A0A. When this is done, integers of magnitude up to $2^{31}-1$ may be printed without 32000 BASIC resorting to 'E' notation.

2.3 Extensions to BASIC I

2.3.1 The EXT#= statement

In BASIC I, EXT# is a function returning the length of an open file; in BASIC 32000, EXT# may also be used on the left of an assignment statement in order to set the length of the file. This requires the presence of a suitable filing system, (DFS, ADFS, or NFS). The main use of this statement is to reduce the length of a file, for example:

```
EXT#out=&1000
CLOSE#out
```

will set the length of the file whose channel number is out to &1000, regardless of its previous extent. PTR# may still be used to increase the extent of a file.

2.3.2 The OPENUP function

In BASIC II, and subsequent versions, a function was added for opening files. In BASIC I, OPENOUT opened a file for output only, and OPENIN opened a file for update (i.e. reading and writing). The new function OPENUP performs the same function as OPENIN did (and has the same internal token value), and OPENIN now opens an existing file for input only.

Programs written using BASIC 1 will still work under 32000 BASIC, but will have OPENIN listed as OPENUP. Programs written under 32000 BASIC which use OPENIN will not work under BASIC 1.

The functions OPENIN, OPENOUT and OPENUP correspond directly to calls to the host operating system routine OSFIND with A = &40, &80 and &C0 respectively on entry.

2.3.3 The OSCLI statement

A new statement, OSCLI, was added to BASIC II. It takes a single string parameter and passes it to the operating system command line interpreter. It therefore acts in a very similar way to preceding a line by *, the difference

being that * commands may not contain BASIC variables. For example, if `a$="FX"` and `b$="12,3"` then the command `*a$+b$` will pass the string `"a$+b$"` to the operating system, probably resulting in an error, whereas `OSCLI a$+b$` will pass the string `"FX12,3"` to the operating system, which will in turn set the auto-repeat speed.

2.3.4 The ' delimiter in VDU

If the character `'|'` appears in the place of a `';` or `'.'` in a VDU statement, the effect is the same as `,0,0,0,0,0,0,0,0`, i.e. nine zero bytes are sent to the VDU drivers. This enables long VDU statements to be made shorter, for example `VDU 19,1,0,0,0,0` can now be written as `VDU 19,1|`. The extra zero bytes are ignored by the VDU.

2.3.5 The ON ... PROC statement

An extension has been added to the `ON ... GOSUB` and `ON ... GOTO` statement. The new statement has the form:

```
ON choice% PROCleft(d%),PROCrigh(d%),PROCup(d%),PROCdown(d%)
```

As with `ON ... GOTO/GOSUB` the statement may be followed by an optional `ELSE`. The statement following the `ELSE` is executed if the `ON` expression is outside of the range 1 to n, where n is the number of `PROCs` listed.

2.3.6 'Unlimited' nesting of structures

The control structures `GOSUB ... RETURN`, `REPEAT ... UNTIL`, and `FOR ... NEXT` no longer have fixed limitations on the depth of nesting. The limit of nesting is imposed by the amount of memory available. Since this is usually at least 500 K bytes, there is no practical limit.

The lifting of the nesting limitation is due to the way in which 32000 BASIC organises its stack (the area of memory in which return addresses for loops are stored). 6502 BBC BASIC uses several stacks: one each for `GOSUBs`, `REPEATs`, `FORs`, and `FN/PROC`s. These are all combined into one structure in 32000 BASIC, which potentially can occupy all of the free memory. This means that the 'Too many' (`FORs`, `REPEATs`, `GOSUBs`) error messages all become 'out of memory' messages.

Because of this new structure, there are some limitations on the way in which the various constructs may be exited. This is reflected by the new error message 'Incorrect nesting of control structures'. An example of a technique which is viable on the version of BASIC which runs with the 6502 second processor, but not the 32000 version is:

```
FOR i=1 TO n:UNTIL TRUE:NEXT
```

which exits 'n' REPEAT loops. Such techniques are very bad programming practice and 32000 BASIC will not tolerate them.

2.3.7 The HELP command

If the user types HELP in immediate mode, 32000 BASIC will print the version number of the current interpreter and other useful information. Because HELP is a command and not a statement, it may not be used from within a program. HELP may not be abbreviated.

2.3.8 The QUIT command

Typing QUIT in immediate mode causes the BASIC interpreter to finish and the Pandora command line interpreter to be re-entered. Again, QUIT may not be used from within a BASIC program as it is a command, not a statement. There is no abbreviation of QUIT.

2.3.9 The EDIT command

A screen editor is provided with 32000 BASIC. This is described in the following chapter.

3 Preparing Programs

BBC BASIC contains a simple line editor, in common with most other dialects of the language. To enter a new line, the line is typed preceded by a line number. For example:

```
>1000 PRINT "Hello, world"
```

will enter a line 1000 with the statement PRINT "Hello, world" in it. Any other line 1000 which existed in the program will be overwritten. To delete a line, its line number is typed without any following text. To delete line 1000, for example, just type:

```
>1000
```

A block of lines may be deleted using the DELETE command:

```
>DELETE 1210,1560
```

will delete all lines between 1210 and 1560 inclusive.

When typing in lines, you may use the normal cursor editing facility, as described in the *BBC BASIC User Guide*. This enables whole or partial lines to be copied, edited, deleted etc.

3.1 The EDIT command

This section describes the facilities offered in the BASIC screen editor. This editor can only be used to edit BASIC programs; it does not cater for text files.

3.1.1 Getting started

If you wish to edit an existing BASIC file, type

```
>LOAD "filename"  
>EDIT
```

and the file will be loaded into the editor and displayed on the screen.

If you wish to create a new BASIC program type:

>EDIT

The editor can be entered at any time from BASIC to edit the currently stored program.

3.1.2 The screen

On entering the editor, you will find the cursor at the centre of the screen, and a status line at the bottom.

The cursor can be moved by using the four cursor keys. The status line at the bottom gives you information on current typing mode, markers and errors. For example:

```
#I#           insert mode active
#O#           overtyp mode active
#I# Marked   marked text involved
```

Insert mode means that characters will be inserted into the BASIC text. Characters to the right of the cursor are displaced to make way for the new characters. In contrast, when in overtyp mode, characters at the cursor position will be erased when new characters are typed.

3.1.3 Locating text

You can use the ordinary cursor keys in combination with either the **(SHIFT)** or **(CTRL)** keys to locate text efficiently.

Key	On its own	SHIFT	CTRL
↑	Up a line	Up a screen	Top of file
↓	Down a line	Down a screen	Bottom of file
←	left a character	Start of screen line	Start of BASIC line
→	Right a character	End of screen line	End of BASIC line

3.1.4 Manipulating text

Manipulation of text within the program is achieved by the use of function keys.

Key	On its own	SHIFT	SHIFT + CTRL
(f0)	Insert/Overtype	Jump to BASIC line	Delete BASIC line
(f1)	Set marker	Copy to buffer	
(f2)	Insert from buffer	Delete marked block	
(f3)	OS star commands	Return to BASIC	

f0 Insert/Overtype

Function key 0 switches between Insert and Overtype mode. The bottom status line of the screen shows the current typing mode. **#I#** means that the text typed is inserted into the document by shifting everything after the cursor to the right. **#O#** indicates that overtype mode is active and you can replace characters by overtyping them.

SHIFT + f0 Jump to line

Pressing **SHIFT + f0** and supplying a line number will cause a jump to that BASIC line. You will be prompted for the line number again if an illegal line number has been given.

SHIFT + CTRL + f0 Delete a BASIC line

Move the cursor to the beginning of the BASIC line (i.e. beginning of the line number) and press **SHIFT + CTRL + f0**. The entire BASIC line including any wrap-over lines will be deleted.

f1 Set marker

This is necessary for block-text operations. The marker is set at the current cursor position; subsequent block-text operations are performed on the block of text defined by the marker and the subsequent cursor position, including the character at the cursor. BASIC line numbers are not included in the block, since moving or deleting a block will cause line numbers to be altered.