

---

# BBC BASIC

reference manual

---

## ARM Evaluation System

**Acorn OEM Products**

---



---

# BASIC

---

Part No 0448,010  
Issue No 1.0  
14 August 1986

© Copyright Acorn Computers Limited 1986

Neither the whole nor any part of the information contained in, or the product described in, this manual may be adapted or reproduced in any material form except with the prior written permission of the copyright holder. The only exceptions are as provided for by the Copyright (photocopying) Act, or for the purpose of review, or in order for the software herein to be entered into a computer for the sole use of the owner of this book.

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

- The manual is provided on an 'as is' basis except for warranties described in the software licence agreement if provided.
- The software and this manual are protected by Trade secret and Copyright laws.

The product described in this manual is subject to continuous developments and improvements. All particulars of the product and its use (including the information in this manual) are given by Acorn Computers in good faith.

There are no warranties implied or expressed including but not limited to implied warranties or merchantability or fitness for purpose and all such warranties are expressly and specifically disclaimed.

In case of difficulty please contact your supplier. Every step is taken to ensure that the quality of software and documentation is as high as possible. However, it should be noted that software cannot be written to be completely free of errors. To help Acorn rectify future versions, suspected deficiencies in software and documentation, unless notified otherwise, should be notified in writing to the following address:

Customer Services Department,  
Acorn Computers Limited,  
645 Newmarket Road,  
Cambridge  
CB5 8PD

All maintenance and service on the product must be carried out by Acorn Computers. Acorn Computers can accept no liability whatsoever for any loss, indirect or consequential damages, even if Acorn has been advised of the possibility of such damage or even if 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.

Econet® and The Tube® are registered trademarks of Acorn Computers Limited.

ISBN 1 85250 005

Published by:

Acorn Computers Limited, Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN, UK

Produced by Baddeley Associates Limited, Cambridge

# Contents

1. About this guide	1
1.1 Conventions used in this guide	1
2. General changes	2
2.1 Starting	2
2.2 Finishing	2
2.3 Line numbering	2
2.4 Token values	3
2.5 Space for strings	3
2.6 Trailing and leading spaces	3
3. New and amended keywords	4
3.1 ABS	4
3.2 AUTO	4
3.3 BPUT	4
3.4 CASE	5
3.5 CIRCLE	5
3.6 CIRCLE FILL	6
3.7 COLOUR	6
3.8 COUNT	7
3.9 DEF	7
3.10 DIM	7
3.11 EDIT	7
3.12 EDITO	8
3.13 ERROR	8
3.14 EVAL	8
3.15 EXT	8
3.16 FILL	9
3.17 FOR	9
3.18 GCOL	10
3.19 GET\$	10
3.20 HELP	10
3.21 IF	10
3.22 INPUT	11
3.23 INSTR	11
3.24 LINE	11
3.25 LIST	12
3.26 LISTO	12

3.27 LVAR	12
3.28 MID\$	13
3.29 OFF	13
3.30 ON	13
3.31 ON	13
3.32 OPENIN and OPENUP	14
3.33 ORIGIN	14
3.34 OSCLI	15
3.35 POINT	15
3.36 PRINT	15
3.37 RECTANGLE	16
3.38 RECTANGLE FILL	16
3.39 RENUMBER	17
3.40 REPORTS	17
3.41 RESTORE	17
3.42 RND	17
3.43 SAVE	18
3.44 SOUND	18
3.45 wline\$	18
3.46 SWAP	18
3.47 TIMES\$	19
3.48 TRACE	19
3.49 TWIN	20
3.50 VDU	20
3.51 WHILE	20
4. New operators	21
4.1 New binary operators	21
4.2 New unary operators	21
4.3 ? and !	21
5. New ways of handling errors	22
5.1 New errors	22
5.2 Fatal errors	22
5.3 Missing errors	22
6. ARM assembler	23



# 1. About this guide

This guide should be used in conjunction with *The BBC Microcomputer User Guide*, which deals with BBC BASIC version 1. We refer to that guide in this book as *User Guide*.

The BASIC supplied with the ARM evaluation system is BASIC version 5. This guide covers all the improvements made since version 1.

The changes to BASIC are dealt with in five main sections:

- some important general changes
- new and amended keywords, commands and functions
- new operators
- new ways of handling errors
- information on the ARM assembler.

For each change, we indicate on which version of BASIC it was introduced.

## 1.1 Conventions used in this guide

Text printed like this is text that you type or see on the screen:

`colour`

Keys that you press are shown like this: `RETURN`

Text printed like this shows the sort of item to type. Here you would type a filename, not the word filename:

`filename`

This example shows that you may type a comma then a numeric. The curly brackets indicate optional items in a syntax line.

`{, numeric}`

## 2. General changes

### 2.1 Starting

#### *BASIC 5*

ARM BASIC is stored on disc, not in ROM. To go into the BASIC interpreter from the ARM prompt, type:

AB **RETURN**

or

AB *filename*[RETURN] to start up with text currently held in a file.

(AB stands for ARM BASIC.)

### 2.2 Finishing

#### *BASIC 5*

To leave the BASIC interpreter, type:

QUIT [RETURN]

### 2.3 Line numbering

#### *BASIC 5*

If you type AB *filename*, and the text file you specify has no line numbers, then ARM BASIC inserts line numbers automatically. This also happens if you edit a program in TWIN and choose BASIC as the exit language from TWIN.

## 2.4 Token values

### *BASIC 5*

The token values for all commands (see *User Guide* page 483) have been changed to provide all the new tokens needed by the extensions. You can run old programs: the old token numbers still refer to appropriate keywords.

## 2.5 Space for strings

### *BASIC 2*

The allocation of space for strings has been improved. The interpreter can now execute:

```
REPEAT a$ =a$ + "***": UNTIL LEN a$ = 255
```

and only allocate 255 bytes.

## 2.6 Trailing and leading spaces

### *BASIC 4*

Trailing spaces will always be stripped from lines entered into the interpreter.

Leading spaces will be stripped from lines entered into the interpreter when a non-zero LISTO is set. The assumption is that there will be a formatted listing on screen when cursor editing is used when LISTO is non-zero.

## 3. New and amended keywords

This section lists, in alphabetical order, new and amended keywords, functions and commands.

### 3.1 ABS

#### **BASIC 2**

*num-var-ABS (numeric)*

can take the absolute value of integers that don't have bit 31 set without returning a string (that is, PRINT -ABS1 works).

### 3.2 AUTO

#### **BASIC 4**

*AUTO { num-const{ , num-const} }*

no longer outputs a space after the line number.

### 3.3 BPUT

#### **BASIC 5**

*BPUT channel, string*

outputs the contents of the string followed by CHR\$ 10 (unless the line ends with ;); the string must be in double quotation marks.

## 3.4 CASE

### *BASIC 5*

The CASE keyword makes it possible to branch in more than one direction.

```
CASE expression
WHEN expression{ , expression}
    any number of statements and lines
OTHERWISE any number of statements and lines
ENDCASE
```

The following restrictions must be observed:

- WHEN, OTHERWISE and ENDCASE must be the first non-space object on a line
- the CASE statement must be the last statement on a line.

Although WHEN keywords can have multi-line blocks, these cannot include CASE keywords: that is, you cannot nest CASE keywords.

OTHERWISE is optional. If it is not present, then no fault will be generated.

For example:

```
CASE JIM
WHEN 0
    PRINT "Zero"
WHEN 1,2
    PRINT "One or Two"
OTHERWISE PRINT "None of the above"
ENDCASE
```

## 3.5 CIRCLE

### *BASIC 5*

```
CIRCLE x,y,r
```

outlines a circle, centre *x,y* radius *r*, and is equivalent to  
MOVE *x,y*:PLOT&95,*x+r,y*

This keyword works only if:

- you are using a BBC Master Series computer with your ARM evaluation system, or
- you are using an earlier BBC Microcomputer with an Acorn Graphics Extension ROM.

## 3.6 CIRCLE FILL

### **BASIC 5**

**CIRCLE FILL** *x,y,r*

fills a circle, centre *x,y* radius *r*, and is equivalent to  
**MOVE** *x,y*:**PLOT&9D**,*x+r,y*

This keyword works only if:

- you are using a BBC Master Series computer with your ARM evaluation system, or
- you are using an earlier BBC Microcomputer with an Acorn Graphics Extension ROM.

## 3.7 COLOUR

### **BASIC 5**

**COLOUR** *a,b*

defines logical colour *a* to be physical colour *b*, and is equivalent to  
**VDU19**,*a,b*;0;

You can type **COLOR** instead of **COLOUR**.

## **3.8 COUNT**

### ***BASIC 2***

A MODE change now resets COUNT.

## **3.9 DEF**

### ***BASIC 5***

You can now pass arrays as well as variables to a procedure or a function.

For example:

```
DEF PROC(A (), a% (), a$ ())  
PROC(FRED (), jim% (), harry$ ())
```

You must not insert the dimensions of arrays.

## **3.10 DIM**

### ***BASIC 2***

DIM is now a function if placed after a statement for example:

```
PRINT DIM(FRED ())
```

gives the number of dimensions

```
PRINT DIM(FRED (), 4)
```

gives the size of the fourth dimension.

Trying to declare an array with a negative dimension gives Bad DIM.

## **3.11 EDIT**

### ***BASIC 4***

EDIT

converts the program currently in memory into ASCII, and takes you into TWIN.

## 3.12 EDITO

### **BASIC 5**

EDITO *num-const*

has the effect of typing LISTO *num-const*, then EDIT.

## 3.13 ERROR

### **BASIC 5**

ERROR *number, string*

causes the error to be reported as *string*, followed by the error number you've specified: this enables you to report special errors.

## 3.14 EVAL

### **BASIC 2**

The lexical analyser is now called correctly from EVAL("TIME").

## 3.15 EXT

### **BASIC 4**

EXT# (*channel*) -*new length in bytes*

updates an open file's extent

This keyword uses OSARGS (see *User Guide* page 454) and works on suitable filing systems such as ADFS.

## **3.16 FILL**

### ***BASIC 5***

**FILL** *a, b*

flood-fills in foreground over background from *a, b* (see also **CIRCLE FILL** and **RECTANGLE FILL**)

This keyword works only if:

- you are using a BBC Master Series computer with your ARM evaluation system, or
- you are using an earlier BBC Microcomputer with an Acorn Graphics Extension ROM.

## **3.17 FOR**

### ***BASIC 5***

Integer FOR statements that try to overflow stop the loop immediately. For example:

```
FOR B%=&7FFFFFFF-10 TO &7FFFFFFF
```

stops immediately.

### ***BASIC 4***

General recursion is now allowed in the FOR loop set-up statement. For example:

```
DEF FNQ FOR J=1TO10  
PRINT J;  
NEXT  
=10  
FORI=FNQ-9 TO FNQ STEP FNQ/ 10
```

now works. In previous versions only the first FNQ, or FNQ keywords without the FOR loop, would work.

## 3.18 GCOL

### **BASIC 5**

GCOL *c*

sets the graphics colour and is equivalent to GCOL 0, *c*.

## 3.19 GET\$

### **BASIC 5**

GET\$#*channel*

gets a string to the next CHR\$ 10 or CHR\$ 13 (or EOF) or 0.

## 3.20 HELP

### **BASIC 5**

HELP

BASIC now has a help system; typing HELP displays information on the kinds of help available, and how to obtain them.

## 3.21 IF

### **BASIC 5**

You can now split an IF...THEN...ELSE statement over several lines. There must be an ENDIF to mark the end of the block.

```
IF expression THEN
  any number of statements or lines
ELSE
  any number of statements or lines
ENDIF
```

The following restrictions must be observed:

- ELSE and ENDIF must be the first non-space object on a line
- THEN must be the last thing on the line.

You cannot nest an IF keyword inside an IF block.

ELSE is optional. If it is not present, then no fault will be generated.

For example:

```
IF Jim THEN
PRINT "Jim was True"
ELSE
PRINT "Jim was False"
ENDIF
```

## 3.22 INPUT

### *BASIC 2*

*INPUT string ; variable*

is equivalent to *INPUT string , variable* – in other words, you can use a semi-colon instead of a comma.

## 3.23 INSTR

### *BASIC 2*

*num-var = INSTR (string, string { , numeric} )*

no longer corrupts the stack (see *User Guide* page 281).

## 3.24 LINE

### *BASIC 5*

*LINE x1,y1,x2,y2*

draws a line from *x1,y1* to *x2,y2*, and is equivalent to *MOVE x1,y1:DRAW x2,y2*

## 3.25 LIST

### **BASIC 4**

Cross-reference and search output is available from LIST. Lines are listed if the specified string is present.

```
LIST IF DEF
LIST 10,1000 IF PRINT
LIST ,2000 IF A*
```

Because of the lexical analysis, it is not possible to search for TIME=90 as a statement. It will only be checked for as a boolean expression. PTR#, HIMEM, PAGE, LOMEM are similarly affected.

## 3.26 LISTO

### **BASIC 5**

LISTO *numeric*

- 0 gives no enhancements, as before.
- 1 space after the line number.
- 2 indent structures.
- 4 split at the : statement delimiter.
- 8 don't list the line number; error at line number references.
- 16 list tokens in lower case.

## 3.27 LVAR

### **BASIC 5**

LVAR

displays all the variables, procedures and functions defined by the current program, together with their current values.

## **3.28 MID\$**

### **BASIC 5**

**MID\$ (a\$ , n{ , m} ) = b\$**

assigns characters from *b\$* into *a\$*, starting at position *n* up to the smallest of *m*, *LENb\$* or *LENa\$ - n*.

## **3.29 OFF**

### **BASIC 5**

**OFF** turns the cursor off.

## **3.30 ON**

### **BASIC 5**

**ON** turns the cursor on.

## **3.31 ON**

### **BASIC 5**

**ON ERROR { LOCAL} PRINT**

defines the restore state when an error occurs to be the state when the **ON ERROR LOCAL** statement was executed.

For example:

```
FOR Z=-10 TO 10
ON ERROR LOCAL PRINT "Can't"
NEXT
PRINT 1/ Z
NEXT
```

## Chapter 3

In addition:

ON ERROR TO

puts error status to stack

ON ERROR RESTORE

restores error status from stack.

### **BASIC 4**

ON *expression* PROCA,PROCB(1,2),PROCC("fred") ELSE PROCD("error")

is now legal syntax.

There is a bug in:

ON .ELSE *statement a*:*statement b*

in that, with GOSUB (and now PROC), *statement b* is executed whether or not the condition is true.

## **3.32 OPENIN and OPENUP**

### **BASIC 2**

OPENIN now opens files for input only. A new keyword OPENUP (open for update) opens files for update. The tokens have been adjusted so that old programs automatically change OPENIN to OPENUP when they are loaded into the new interpreter.

## **3.33 ORIGIN**

### **BASIC 5**

ORIGIN *x,y* sets the graphics origin to *x,y*, and is equivalent to VDU29,*x*;*y*;

## 3.34 OSCLI

### **BASIC 2**

A new statement OSCLI has been introduced. It takes a string expression and gives it to the operating system. For example:

```
OSCLI "KEY " + STR$ (2) + " LIST"
```

programs the function key **F2**

to print LIST, leaving the user to press **RETURN**

It has no unique errors of its own, just the normal Type mismatch error.

## 3.35 POINT

### **BASIC 5**

POINT *x, y*

plots a single point at *x,y*, and is equivalent to PLOT69, *x, y*.

## 3.36 PRINT

### **BASIC 5**

The accuracy has been improved. In general format in PRINT or STR\$ (see *User Guide* page 326), .05 prints as .05 not 5E-2.

### **BASIC 5**

Tabulation is better because COUNT is now based on a 32-bit value.

## **BASIC 2**

The binary-to-decimal string conversion routines, used for PRINT and str\$, have been changed to allow the use of 10 figures of precision on printing. The initial value of @% is now &0000090A to give the same results on startup.

Note that @%=10 now gives the internal default of 10 figures. The changes allow the maximum positive integer 2147483647 to be printed out (and get 2<sup>33</sup> right). Str\$ when not controlled by @% uses the new 10 figure default, so it will now give different answers. For example, 7.7 (a recurring binary fraction) will be converted to 7.699999999.

## **3.37 RECTANGLE**

### **BASIC 5**

RECTANGLE *x, y, a, b*

outlines a rectangle, position *x,y* width *a*, height *b*.

## **3.38 RECTANGLE FILL**

### **BASIC 5**

RECTANGLE FILL *x, y, a, b*

fills a rectangle.

This keyword works only if:

- you are using a BBC Master Series computer with your ARM evaluation system, or
- you are using an earlier BBC Microcomputer with an Acorn Graphics Extension ROM.

### **3.39 RENUMBER**

*BASIC 4*

RENUMBER is not confused by &8D in comments or strings.

### **3.40 REPORT\$**

*BASIC 5*

REPORT\$ returns the error REPORT as a string.

### **3.41 RESTORE**

*BASIC 4*

The bug of restoring to a line without a DATA token on it but with a , has been removed.

### **3.42 RND**

*BASIC 4*

The random number generator (rewritten in BASIC 3 to go faster) has now been changed. RND(1) and RND(*n*) give different values from earlier versions of RND. RND gives the same value. The changes are to avoid statistical errors in certain circumstances.

## 3.43 SAVE

### *BASIC 5*

SAVE

saves the program to the filename given after REM on the first line.

### *BASIC 3*

SAVE a\$ +b\$

works correctly.

## 3.44 SOUND

### *BASIC 5*

SOUND OFF

turns the sound off (it has the same effect as \*FX210).

SOUND ON

turns the sound on (it has the same effect as \*FX210,0,0).

## 3.45 STR\$

For a new effect of @% on str\$ , see PRINT.

## 3.46 SWAP

### *BASIC 5*

SWAP a,b

exchanges the values of the variables (which must exist)

## **3.47 TIME\$**

### ***BASIC 4***

If you are using a BBC Master Series computer as part of your ARM evaluation system, you can use the new psuedo-variable `TIME$` to find the time.

`TIME$` does not work with earlier BBC Microcomputers, which don't have a real-time clock.

The ARM evaluation system stores the time independently (you set it each session using the `DATE` command); `TIME$` overrides the ARM time and goes straight to the Master Series computer's real-time clock.

The format of the string held in `TIME$` is as follows:

```
Wed, 31 Dec 1900.23:59:59
```

Assigning `TIME$ = "fred"` passes the string directly to the operating system with the length in the first byte.

## **3.48 TRACE**

### ***BASIC 5***

ARM BASIC has the ability to trace procedures and functions and it can trace in single-step mode. Single-step mode gives the number or procedure name in curly brackets instead of square brackets and waits for a key to be pressed before continuing.

For example:

```
TRACE PROC  
TRACE STEP ON  
TRACE STEP 1000  
TRACE STEP PROC
```

## 3.49 TWIN

### *BASIC 5*

TWIN

converts the program currently in memory into ASCII, and takes you into TWIN.

## 3.50 VDU

### *BASIC 4*

A new terminator | is available in VDU. It can appear after expressions and sends nine zeros to the vdu drivers. For example, you can now say:

```
VDU19,1,2|
```

instead of:

```
VDU 19,1,2,0,0,0,0,0,0,0,0
```

## 3.51 WHILE

### *BASIC 5*

This new keyword is like REPEAT...UNTIL, except that the expression is evaluated at the beginning of the loop.

```
WHILE expression  
block  
ENDWHILE
```

If the condition is initially false, then the next ENDWHILE at the same nesting level of WHILE loops is searched for.

For example:

```
A=0  
WHILE A<0  
A+1  
ENDWHILE
```