ACORNSOFT LANGUAGES

The BASIC Editor

on the BBC Microcomputer Model B

The BASIC Editor

on the BBC Microcomputer Model B

ACORNSOFT

The BASIC Editor software was written by Pete Morris and Chris Gibson. This manual was written by Morrell Media Services, Reading, England.

ISBN 0 907876 18 8

First published in 1985 by Acornsoft Limited

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

FIRST EDITION
Printed in the United Kingdom
Acornsoft Limited, Betjeman House, 104 Hills Road. Cambridge CB2 lLQ, England. Telephone 1(223) 316039

Preface

This book tells you how to use The BASIC Editor. It is intended for users of BBC BASIC on the BBC Microcomputer.

The book is organised into seven chapters the first of which is an introduction to the editor. If you have never used an editor before then you should read this chapter carefully. If you have used an editor before you need only look through this chapter quickly.

Chapter 2 describes some of the more common editing functions provided by the editor. Chapters 3 to 7 cover more advanced editing functions.

Appendices A, Band C contain summaries of the information given in the main body of the book. They are intended for reference purposes. Appendix D gives a list of error messages.

Contents

1 Getting started 1

1.1 What is The BASIC Editor? 1
1.2 Switching on 1
1.3 Creating and editing a simple program 2
1.4 Saving a program 3
1.5 Running a program 3
1.6 Getting back into The BASIC Editor 3
1.7 Loading a program 4
1.8 The NEW and OLD commands 4
1.9 Using the function keys 4
1.10 Useful hints and help 4

2 Editing a program 6

2.1 Loading a program for editing 6
2.2 Moving the cursor 6
2.3 Inserting and overtyping text 7
2.4 Inserting extra statements 7
2.5 Joining and splitting statements 7
2.6 Swapping case 8
2.7 Creating extra lines 8
2.8 Marking a statement 8

3 Joining programs and renumbering 9

3.1 Joining programs 9
3.2 Renumbering a program 9

4 Copying, deleting and moving text 11

4.1 Line commands 11
4.2 Inserting line commands 11
4.3 Removing line commands 11
4.4 Deleting text 11
4.5 Moving text 12
4.6 Copying text 13

5 Finding and changing strings 14
5.1 The EDIT command 14
5.2 The FIND command 14
5.3 The CHANGE and QCHANGE commands 15
5.4 General notes on EDIT, FIND, CHANGE and QCHANGE 16

6 Using labels 17
6.1 What are labels? 17
6.2 Choosing label names 17
6.3 Inserting labels 17
6.4 Running a program with labels 18

7 Changing edit modes and defaults 19
7.1 The TAB key 19
7.2 The cursor 19
7.3 Programming your own function keys 20
7.4 Making your own editing profile 20
7.5 Scrolling 20
7.6 Changing mode and colour 21

Appendix A 22
Command reference table 22

Appendix B 29
Editing reference table 29

Appendix C 31
Function key reference table 31

Appendix D 34
Error messages 34

Index 39

1 Getting started

1.1 What is The BASIC Editor?

The BASIC Editor is a powerful and sophisticated tool to help you develop and maintain BASIC programs.

Like a word processor, it allows you to move around the program and change any part of it. Unlike a word processor, the end product is a program in a form which you can run directly. In addition, it provides you with many features specific to creating BASIC programs, such as the automatic generation of line numbers.

1.2 Switching on

If you have not yet installed the BASIC Editor ROM in your computer you should refer to the enclosed leaflet for fitting instructions or consult your dealer.

Place the function key card under the clear plastic strip above the red function keys at the top of your keyboard, and switch your machine on. If the words `The BASIC Editor` do not appear at the top of the screen, type:

```
*BE
```

and press RETURN.

You should now have a screen similar to this:

```
The BASIC Editor

Program size   : 2
Bytes free  : 25342
Screen mode    : 7
```

If you do not, check that you typed the command correctly and that the ROM is properly installed. The actual numbers on the screen may be different for your particular machine.

This is the 'command screen', it is from here that commands for such things as loading and saving programs are used. Programs are created and edited from the 'edit screen'.

1.3 Creating and editing a simple program

Type NEW, press RETURN and then press ESCAPE. You will see an empty screen with the number 10 at the top with a flashing cursor beside it. This is the 'edit screen', and it is here that you will create the text for your program. The number is the current BASIC line number and one of these is produced by The BASIC Editor at the start. of each new line. In the text below, line numbers are included, but there is no need for you to type them.

Now type this line:

```
10 FOR X=2 TO 30
```

and press RETURN. You should see the next line number, 20. appear 10 the left. Now complete the program by typing the following, pressing RETURN at the end of line 20.

```
10 FOR X=2 TO 30
20 PRINT X, X+X
30 NEXT X
```

Let us assume that the program needs to be changed to look like this:

```
10 FOR X=2 TO 20 STEP 2
11 PRINT X*X
20 PRINT X+X
30 NEXT X
```

To achieve this. we need to change line 10, add a new line 11 and delete part of line 20.

Changing a line

The four grey arrow keys and used move the cursor around the screen. To change line to. move the cursor so that it is over the `3` of `30` and type a `2`. The `2` will replace the `3`. this is called overtyping.

You can now add `STEP 2` by moving the cursor to the end of the line and typing the extra text.

Adding an extra statement

To add the new line 11 to the program. leave the cursor on line 10 and and press RETURN. Line number `11` is automatically created and you can complete the line by typing:

```
PRINT X*X
```

Deleting text

To delete text from line 20, move the cursor so that it is over the first `X`, hold the SHIFT key down and press and release the DELETE key; the `X` will be removed. You can delete the comma in the same way.

1.4 Saving a program

The ESCAPE key is used to switch between the command screen and the edit screen. Press it now to return to the command screen.

When you have created and edited your program, you can save it to the current filing system by going into the command screen and using the command `SAVE`. For example, to save a program called `MYPROG`, type:

```
SAVE MYPROG
```

and press RETURN.

1.5 Running a program

In order to run your program you will need to return to BASIC. To do this, go to the command screen, type:

```
EXIT
```

and press RETURN. This will take you back into BASIC where you can list or run programs in the usual way.

Alternatively, if you only want to run the program, go to the command screen, type:

```
RUN
```

and press RETURN. This will take you back into BASIC and run your program immediately.

1.6 Getting back into The BASIC Editor

To get back into the editor at any time, type:

```
*BE
```

and press RETURN. Once in the editor, you can edit a program already in memory or load a new program.

1.7 Loading a program

To load an existing program, return to the command screen and use the command `LOAD`. For example, to load a program called `MYPROG`, type:

```
LOAD MYPROG
```

and press RETURN.

1.8 The NEW and OLD commands

You have already seen the use of the command `NEW`. This is like the `NEW` in BASIC, it allows you to create a new program. The command `OLD` restores your original program if there is one in memory. If not, then you will see the error message:

```
Bad program
```

and the information about program size and number of free bytes will no longer be displayed.

1.9 Using the function keys

The red function keys at the top of the keyboard are very useful because they allow you direct access to editing functions from the edit screen. Many commands available from the command screen have function key equivalents. This means that you do not have to go to the command screen to use them.

For example, function key 9 is equivalent to `EXIT` and SHIFT plus function key 0 is equivalent to `NEW`.

In this book, function keys are denoted by 'f'. Function key 9 is, therefore, written f9. If you need to use SHIFT or CTRL with the function key, then this is written:

SHIFT f2, or:

CTRL f6

and so on. Remember to hold the SHIFT or CTRL key down while pressing the function key in such cases.

1.10 Useful hints and help

Getting help

The BASIC Editor has a help facility. From the command screen, type:

```
HELP
```

and press RETURN. This will give you a list of available commands and their syntax.

All commands can be abbreviated if required and the minimum abbreviations are given in Appendix A. Abbreviated commands do not need to be terminated with a full stop.

Note for BASIC I users

Users who have BASIC I in their machines will need to type `OPENUP` wherever they require `OPENIN` to appear in their program. Failure to do so will cause problems when BASIC I is used to interpret the program. Users with BASIC II should use `OPENIN` and `OPENUP` in the normal way.

Undoing changes made to a line

If, before you have left a statement you want to abandon any changes you have made to it, press SHIFT f2 (`UNDO`) which will restore the statement to the way it was before you made the changes.

Editing from a particular line

As well as the ESCAPE key, several other commands will take you to the edit screen. `TOP` will take you to the edit screen with the cursor on the first line of the program and `END` will take you to the edit screen with the cursor on the last line. Alternatively, typing a line number and pressing RETURN will take you to the edit screen with the cursor on that line or the next highest.

Information about programs

At any time, you can get information about your program and the current TAB, colour, scroll settings and so on. Type `INFO` from the command screen and you will see information like this:

```
ScroLL on
Overtype mode
White on bLack

Program name : MYPROG
Last search  : None

Tab vaLue    : 5        No. of Lines : 4
First Line   : 10       Last Line    : 40
Current Line : 10       Marked Line  : None

Pending commands: None
```

It is possible to change the default settings of TAB, scroll mode and colour and this is explained fully in chapter 7.

2 Editing a program

2.1 Loading a program for editing

Programs are loaded for editing from the command screen with the `LOAD` command. For example, to load a program called `MYPROG`, type:

```
LOAD MYPROG
```

and press RETURN. When the program is loaded, press ESCAPE to go to the edit screen. Your program will be listed with a flashing cursor to the left of the first line.

2.2 Moving the cursor

The simplest way to move the cursor around your program is to use the four arrow keys. These move the cursor left, right, up and down. However, you will often want to move to a particular place in your program such as the beginning, or a particular line.

The table below lists the various ways in which you can move the cursor.

Right, left, up, down	→←↑↓
Fast right	SHIFT →
Fast left	SHIFT ←
Top of previous screen	SHIFT ↑
Top of next screen	SHIFT ↓
Beginning of next statement	CTRL →
Beginning of previous statement	CTRL ←
First statement on screen	CTRL ↑
Last statement on screen	CTRL ↓
Beginning of next line	SHIFT TAB
Next tab position	TAB
Top of the program	f'2 (TOP)
Bottom of the program	f.3 (END)

To position a particular statement at the top of the screen you should move the cursor on to that statement and press f7 (TOP OF SCREEN).

A note on the use of TAB

The tab setting has been preset to every fifth character position. This means that when you press TAB the cursor moves to every fifth character position along a line. When it reaches the end of a line, it jumps to the beginning of the next.

It is possible to change the default setting of TAB and this is explained fully in chapter 7.

2.3 Inserting and overtyping text

If you move the cursor so that it is at the beginning of a line of text and type some characters, you will see that the existing text is overwritten. This is because you are in 'over type mode'. To insert text into a statement you will have to go into 'insert mode'. Do this by pressing f1 (INSERT/OVERTYPE). The cursor will change into a block and any text you type will be inserted into the statement moving any subsequent text to the right. To return to overtype mode, press f1 again.

2.4 Inserting extra statements

Extra statements can be inserted at any point in your program. Move the cursor to the line below which you want the extra statement to appear and press RETURN; a new line number and a blank line will be provided automatically. Subsequent uses of RETURN will add more lines.

Occasionally, inserting new statements between existing ones will mean that the program has to be renumbered. If this should happen the editor will do it for you automatically.

To insert a new statement at the top of your program, type the command IT from the command screen. This will take you to the edit screen with the cursor on the new line at the top of the screen. To insert a new statement at the end of your program, use the command IE.

You can also use the function keys to insert new lines, f8 (INSERT AT END) inserts a new line at the end of the program and SHIFT f8 (INSERT AT TOP) inserts a new line at the top of the program.

2.5 Joining and splitting statements

To join two statements together, move the cursor so that it is on the first of the two and press CTRL f3 (JOIN STATEMENTS). The second statement is added on to the end of the first and a colon is put between the two statements

automatically. If the combined length of the two statements exceeds the maximum allowed, then the join is not performed and an error message is displayed.

To split a statement, use CTRL f2 (SPLIT STATEMENT). Position the cursor on the character which will be the start of the new statement, press CTRL f2 and two separate statements will be formed. You can only split a statement somewhere in the middle and, as you are creating a new statement, this may cause a renumbering of your program.

2.6 Swapping case

To swap case, use SHIFT f3 (SWAP CASE). This converts one alphabetic character at a time from lower case to upper case, or vice versa.

2.7 Creating extra lines

When you need more lines for a statement, The BASIC Editor will automatically provide you with a fresh line and move the following statements down. Alternatively, you can create another blank line by using SHIFT f4 (EXTEND STATEMENT). Any blank lines that are put in but not used will be omitted from the final program.

Length of statements allowed

When you insert text, you should be aware of the maximum length allowed for a single statement. This is 251 characters which is about three and a half lines in mode 3 and seven and a half lines in mode 7.

2.8 Marking a statement

Sometimes you may want to mark a statement so that you can move back to it quickly later on.

To mark a statement, move the cursor on to the statement and press CTRL f0 (MARK). A full stop will appear on the screen between the line number and the start of the text. This is the 'marker'. Only one statement at a time can be marked; marking another line will cause the marker to be removed from the first.

To go to a marked statement, press CTRL f1 (GOTO). If no statement is marked GOTO will have no effect.

To remove a marker move the cursor on to the marked statement and press CTRL R.

3 Joining programs and renumbering

3.1 Joining programs

It is possible to join two programs together with The BASIC Editor, using the command `APPEND`. Load the first program in the usual way:

```
LOAD PROG1
```

and then, staying on the command screen, join another program on to the end of it:

```
APPEND PROG2
```

If the resulting program is not legal BASIC or if it is too large to fit into the available memory, an error message will be displayed and the original program will be restored.

If the new program is too big for the current editing mode, the mode will be reset to that which requires the least amount of memory tie mode 7).

If there is any overlap of line numbers, the new program will be renumbered.

3.2 Renumbering a program

You can renumber your program at any time from the edit screen or the command screen.

To renumber a program from the edit screen, use the function key f4 (RENUMBER).

To renumber a program from the command screen, use the command `RENUMBER`.

Renumbering will start at line 10 and go up in increments of 10. You can alter both the increment value, and the line at which renumbering starts, by using the `RENUMBER` command with this extra information included. For example:

```
RENUMBER 1000
```

will renumber from line 1000 in increments of 10 and

```
RENUMBER 1000,5
```

will renumber from line 1000 in increments of 5. The increment you specify

must be in the range 1-255. If the line number exceeds the maximum (32767), an error message will be displayed.

Problems arising during renumbering

Renumbering is normally straightforward unless:

1. The editor is renumbering after joining two programs together, or
2. The BASIC Editor encounters a `GOTO` or a `GOSUB` statement which refers to a line number which does not yet exist

In the first case, confusion arises where a reference to a line number within a statement is ambiguous, for instance, `GOTO 10`. This happens when line 10 actually exists in both of the programs you have joined together. The BASIC Editor defaults into 'believing' that the line 10 referred to belongs to the first program. This may of course not always be the case. The problem can be avoided by temporarily using unique label references instead of line numbers and this is fully explained in chapter 6.

In the second case, The BASIC Editor will respond by replacing all non-existent line numbers referred to in `GOTO` and `GOSUB` statements by the characters `'@@@@'`. An error message will then be displayed. Again, the use of labels rather than line numbers will help you to avoid this. However, if you do get this problem you can correct all the `'@@@@'` strings using `FIND` and `EDIT` which are explained fully in chapter 5.

4 Copying, deleting and moving text

4.1 Line commands

The BASIC Editor provides facilities for deleting, moving and copying either single statements or blocks of text. Statements or blocks to be acted upon must be marked with 'line commands'.

4.2 Inserting line commands

When you wish to insert a line command, move the cursor to the appropriate line and hold down the CTRL key while typing the command which will consist of one or more letters. The line command will replace the line number of the line like this:

```
    10 X = A+B
CE     X = A+B
```

Apart from the destination for a move or copy command, which will be described later, only two lines at a time may contain line commands.

4.3 Removing line commands

To remove a line command from a line, move the cursor to that line and press CTRL R. If you move the cursor to a line without a command, pressing CTRL R will remove all line commands from the program.

Line commands are not stored as part of your program and so there is no need to remove them before saving or running it.

4.4 Deleting text

Deleting characters

To delete the character immediately to the left of the cursor, press DELETE.

To delete the character at the cursor position, press SHIFT DELETE.

To delete all the characters from the cursor to the end of the line, press f6 (DELETE TO END OF LINE).

Deleting a single statement

The line command is `D`. Move the cursor to the line and press CTRL D. Now press f0 (EXECUTE) to execute the deletion.

Deleting a block of text

The line command is `DD`. Move the cursor to the first line of the block and press CTRL D twice. Do the same with the last line of the block and press f0 to execute the deletion.

To delete from a line to the end of the program, the line command is `DDE`. Move the cursor to the line, insert the command and press f0.

To delete from a line to the top of the program, the line command is `DDT`. Move the cursor to the line, insert the command and press f0.

4.5 Moving text

To move text within a program you will have to mark the destination of the text to be moved as well as the text itself. To mark the destination move the cursor to the appropriate line and insert the line command `A` if you want the moved text to appear after the line or the command `B` if you want the text to appear before it.

If you are moving text to the top or to the end of the program, you do not need to mark a destination, just use the appropriate line commands with the text to be moved.

You should note that moving text within the program will result in automatic renumbering of the program.

Moving a single statement

The line command is `M`. After marking the destination, move the cursor to the line and press CTRL M. Press f0 (EXECUTE) to execute the move.

To move a statement to the end of the program the line command is `ME`. Move the cursor to the line, insert the command and press f0.

To move a statement to the top of the program the line command is `MT`. Insert the command and press f0.

Moving a block of text

The line command is `MM`. After marking the destination, move the cursor to the first line of the block and press CTRL M twice. Do the same with the last line of the block and press f0 to execute the move.

To move a block of text to the end of the program, insert the line command `MM` at the first line of the block and the command `MME` at the last line of the block. Press f0 to execute the move.

To move a block of text to the top of the program, insert the line command MM at the first line of the block and the command `MMT` at the last line of the block. Press f0 to execute the move.

4.6 Copying text

To copy text within a program, you will have to mark the destination of the duplicated text as described in section 4.5. If you are copying text to the top or to the end of the program you do not need to mark a destination, just use the appropriate line commands with the text to be copied.

You should note that copying text within the program will result in automatic renumbering of the program.

Copying a single statement

The line command is `C`. After marking the destination, move the cursor to the appropriate line and press CTRL C. Press f0 (EXECUTE) to execute the copy.

To copy a statement to the end of the program, the line command is `CE`. Move the cursor to the line, insert the command and press f0

To copy a statement to the top of the program the line command is `CT`. Insert the command and press f0

Copying a block of text

The line command is `CC`. After marking the destination, move the cursor to the first line of the block and press CTRL C twice. Do the same with the last line of the block and press f0 to execute the copy.

To copy a block of text to the end of the program, insert the line command `CC` at the first line of the block and the command `CCE` at the last line of the block. Press f0 to execute the copy.

To copy a block of text to the top of the program, insert the line command `CC` at the first line of the block and the command `CCT` at the last line of the block. Press f0 to execute the copy.

5 Finding and changing strings

The BASIC Editor provides special facilities for searching a program for one or all occurrences of a particular string and changing it if required. There are four commands available and these are `EDIT`, `FIND`, `CHANGE` and `QCHANGE`. All four commands are used from the command screen.

5.1 The EDIT command

The `EDIT` command is used to search a program for the first and subsequent occurrences of a string, allowing you to edit each statement as it is found.

For example, to change some `DATA` statements type:

```
EDIT DATA
```

and press RETURN. The command screen will be replaced by the edit screen with the first line containing the string `DATA` at the top. You can now edit the statement in the usual way.

Press f5 (CONTINUE) to move on to the next occurrence of the string.

Press ESCAPE to return to the command screen.

5.2 The FIND command

The `FIND` command searches through a program and displays all the statements containing the search string. For example, to look at all the `DATA` statements, type:

```
FIND DATA
```

and press RETURN. The command screen will be replaced by a screen with all the lines containing the string `DATA` displayed. If there are more occurrences of the string than can be displayed on a single screen, press f5 (CONTINUE) to see the next screenful.

Although the screen looks much like the normal edit screen, the cursor will have changed into a block and you will not actually be able to edit any text, although you will be able to move the cursor up and down with the arrow keys. Only a few keys are valid during a `FIND` command.

- Press ESCAPE to return to the command screen.

- Press f2 (TOP) to go to the top of the program. You will be taken to the normal edit screen where your program will be displayed with the cursor on the first line.

- Press f3 lEND) to go to the end of your program. Again, you will be taken to the normal edit screen with the cursor on the last line of your program.

- Press CTRL f0 (MARK) to mark the statement you are on.

- Press CTRL f1 (GOTO) to edit the marked statement. You will be taken to

the normal edit screen with the cursor on the marked statement at the top of the screen.

- Press f7 (TOP OF SCREEN) to position the current statement at the top of the screen and edit from that point.

5.3 The CHANGE and QCHANGE commands

There are two commands which enable you to replace one string with another. The first is the `CHANGE` command which will unconditionally replace all occurrences of one string by another. For example:

```
CHANGE water wine
```

will change the string `water` into `wine` wherever it is found in the program. The second command, `QCHANGE`, gives you the opportunity to decide whether or not each occurrence of the string should be changed. For example, to change some occurrences of `water` into `wine` type:

```
QCHANGE water wine
```

and press RETURN. You will be taken on to the edit screen with the first occurrence of the search string positioned at the top of the screen with the cursor on the 'first character of the string.

- Press Y to change the string and move on to the next occurrence.

- Press N to leave the string unchanged and move on to the next occurrence.

- Press ESCAPE to return to the command screen.

Press f7 (TOP OF SCREEN) to allow normal editing facilities from the top of the screen.

In the examples given above, neither the search nor the replace strings contained any spaces. However, `CHANGE` and `QCHANGE` can both be used with strings containing spaces. Type `CHANGE` or `QCHANGE` alone and press RETURN. You will be prompted for the search string and then the replacement string.

5.4 General notes on EDIT, FIND, CHANGE and QCHANGE

l. When the last occurrence of the search string has been found, you will be returned to the command screen.

2. Always use upper and lower case in the search string exactly as they are used in the program.

3. If a search is unsuccessful, you will be returned to the command screen and the message `String not found` will be displayed

4. For `EDIT` and `FIND`, the search string may contain spaces within it but not at the beginning or the end. To take account of leading or trailing spaces, type `EDIT` or `FIND` alone and press RETURN. You will be prompted for the search string.

5. For `EDIT` and `FIND`, the search string is remembered so that subsequent presses of f5 (CONTINUE) from the edit screen will find the next occurrence of the string. The search string is also displayed on the `INFO` screen.

6 Using labels

6.1 What are labels?

Labels are a feature of The BASIC Editor which provide a useful way of referring to other statements when their line numbers are unknown. For example, when a program contains `GOTO` or `GOSUB` statements, it may be necessary to refer to a line number which does not yet exist. If the program is renumbered as a result of editing, any such references will be replaced with the string `'@@@@'` and you will be returned to the command screen with an error message.

A more serious problem may arise if renumbering occurs as a result of the `APPEND` command. In such a case, all references to line numbers that are duplicated in the two programs will be assumed to refer to the line in the first program and this will not always be the case. Using labels instead of line numbers will help you to avoid problems with statements whose line numbers are unknown or ambiguous.

6.2 Choosing label names

Label names must begin with `'@'` and then follow the rules for BASIC variables. It can be helpful to choose a name which hints at the function of the code that follows it.

Labels should not start with a BASIC keyword (`@PRINT`, `@ERROR` etc), but such words can be used in lower case.

Labels should be unique; if the same label is used more than once, only the first occurrence will be recognised.

6.3 Inserting la bels

The use of labels is best demonstrated with an example. It is common practice, when writing BASIC programs, to include a statement like:

```
10 ON ERROR GO TO 1000
```

somewhere near the beginning of the program to trap errors. Using a label, the statement would look something like this:

```
10 ON ERROR GO TO @TRAP
```

and this is known as a 'label reference'.

Later on in the program, when you want to code the error routine, you must insert a 'label statement' before the code. This takes the form of a REM statement, for example:

```
520 REM @TRAP
```

However the label is being used, all label references must have a corresponding label statement somewhere in the program.

6.4 Running a program with labels

Before you can run your program the labels must be converted back to line numbers and The BASIC Editor will do this for you automatically. Go to the command screen, type:

```
NUMBER
```

and press RETURN. The BASIC Editor will convert each label reference to the line number of its corresponding label statement. The program can then be run in the usual way. For example, a program that looked like this before the NUMBER command was used:

```
 40 GOTO @LABEL

110
120 REM @LABEL
130
```

would look like this when it is ready to be run:

```
 40 GOTO 120

110
120 REM @LABEL
130
```

If replacing a label with a line number would make the line too long, an error message will be displayed. An error will also be raised if there are any label references which are not matched by corresponding label statements.

If you wish to reinstate the label references, before using APPEND for example. type:

```
LABEL
```

and press RETURN. The label references will replace the appropriate line numbers.

The NUMBER and LABEL commands both have a function key equivalent:

SHIFT f7 and SHIFT f6, respectively.

7 Changing edit modes and defaults

7.1 The TAB key

The TAB key is a convenient way of moving quickly along a line and can be especially useful when you want to align code into columns to improve readability. Even BASIC programs can be easier to understand if they are well formatted. For example:

```
 10 FOR 1% = 1 TO 100
 20     READ name$
 30     IF LEN(name$) = 0 THEN PROC_finish
 40     J% = 1
 50     REPEAT
 60        IF name$ = surname$(J%) THEN
    PROC_print_letter
 70        J% = J% + 1
 80     UNTIL surname$(J%) = "*END*"
 90     surname$(I%) = name$
100 NEXT
```

The `TAB` command is used to change the distance between tab stops on the edit screen. The minimum and maximum values for the `TAB` command are 0 and 80. Setting `TAB 0` effectively disables the TAB key, whereas setting the `TAB` to a value greater than the edit screen width will make the TAB key always position the cursor at the beginning of the next screen line. For example, type:

```
TAB 10
```

and press RETURN to set the distance between tab stops to 10.

7.2 The cursor

The BASIC Editor already gives you a large variety of movement commands, but it is useful to be able to speed up normal cursor movements. The operating system command:

```
*FX 12,n
```

allows you to set up a wide range of cursor speeds and is issued from the command screen. The smaller the value of n, the faster the cursor. The default value is 8, and a lower value will give you a faster than normal cursor.

The COPY key can be used to give you a non-flashing cursor. If you press the COPY key from the edit screen, you will see that the cursor stops flashing. Pressing COPY again will make it flash more quickly and pressing it a third time will make it flash at its normal speed.

7.3 Programming your own function keys

The BASIC Editor makes extensive use of the normal function keys, but you can program your own in the usual way with the `*KEY` command. To access them press CTRL SHIFT and the function key. This applies to both the command and edit screens. You may use these keys for commonly used commands on the command screen or for long variable names, BASIC keywords, etc on the edit screen.

7.4 Making your own editing profile

Disc users can set up an editing profile. This is a set of commands which you can execute to change The BASIC Editor's defaults for such things as mode, colours, scrolling, etc. You can store a set of commands using `*BUILD` and then invoke it with the `*EXEC` command. You may choose to have this as your auto-boot file. For more information on `*BUILD` and `*EXEC`, refer to the *BBC Microcomputer System User Guide and the Disc Filing System User Guide*.

For example, the following sequence of commands will create a profile called `'START'`, which calls The BASIC Editor, sets the edit mode to 3, gives you green text on a black background and starts editing in insert mode:

```
*BUILD START    <RETURN>
*BE             <RETURN>
MODE 3          <RETURN>
FORE G          <RETURN>
BACK N          <RETURN>
INSERT          <RETURN>
                <ESCAPE>
```

You can then execute these commands by entering `*EXEC START`.

7.5 Scrolling

The BASIC Editor allows scrolling to be disabled so that moving down from the bottom of the screen brings the cursor on to the top line of the screen and moving up from the top of the screen brings the cursor on to the bottom line. To disable scrolling, type:

```
NOSCROLL
```

from the command screen or press CTRL f6 (SCROLL OFF). To enable scrolling, type:

```
SCROLL
```

from the command screen or press CTRL f5 (SCROLL ON).

7.6 Changing mode and colour

To change the mode in which you edit your program, type the command `MODE` followed by one of the numbers 0 to 7. Modes 2 and 5 are not allowed because they only show 20 characters on a line.

You can also change mode from the edit screen by pressing SHIFT 15 (MODE) which takes you through the allowable modes in sequence.

In mode 7, text is white on a black background, in the other modes, the text is white and the background blue. In any mode other than 7, you can change these colours to suit your display by pressing CTRL f7 (BACKGROUND) and CTRL f8 (FOREGROUND). These keys take you through each colour in turn.

You can also change the colour of the editing screen from the command screen. Use the `BACK` and `FORE` commands. These have the syntax `BACK <colour>` and `FORE <colour>` where the colours available are:

black	N
red	R
green	G
yellow	Y
blue	B
magenta	M
cyan	C
white	W

For example, to change the foreground to blue, type:

```
FORE B
```

and press RETURN.

Appendix A

Command reference table

This appendix outlines the syntax of each command, explains its function and details any restrictions. All commands can be abbreviated if required and the minimum abbreviation is given in brackets after each command. All commands can be entered in upper or lower case.

In addition to these commands, typing a line number will edit from that line. If the line does not exist, editing continues from the next line or from the end of the program.

APPEND (A)

```
APPEND <program>
```

`APPEND` loads a program from the current filing system on to the end of the program already in memory. If the resulting program is not legal BASIC or if it is too big to fit into the available memory, an error message is displayed and the original program is restored. If the program is too big for the current editing mode, the mode is reset to that which requires the least amount of memory. If the line numbers of the two programs overlap, the program is renumbered.

Example

```
APPEND MYPROG
```

BACK (B)

BACK <colour>

`BACK` changes the background colour of the edit screen. Available colours are black (N), red (R), green (G), yellow (Y), blue (B), magenta (M), cyan (C) and white (W).

Example

```
BACK B
```

CHANGE (C)

```
CHANGE <string1> <string2>
```

`CHANGE` searches through a program and replaces all occurrences of `<string1>` with `<string2>`. If the string is not found, or if the

replacement would make a line too long to be edited, an error is raised. If spaces are to be significant, the prompted version of the command should be used. In this case, the replace string may be null, effectively deleting the search string wherever it is found.

Example

```
CHANGE water wine
```

```
CHANGE
    Search string    : INPUT
    RepLace string   : INPUT LINE
```

EDIT (ED)

```
EDIT <string>
```

EDIT searches through a program for the given string. The search string may contain embedded spaces but if leading or trailing spaces are to be significant, the prompted version of the command should be used. If the string is not found an error is raised. If the string is found, the program is displayed on the edit screen with the first occurrence of the string at the top of the screen. Subsequent searches can be carried out using the to (CONTINUE) key.

Example

```
EDIT PRINT TAB
```

END (E)

```
END
```

END switches to the edit screen where the last page of the program is displayed with the cursor positioned on the last statement.

EXIT (EX)

```
EXIT
```

EXIT clears the screen, returns to BASIC and issues a BASIC OLD command if there is a program in memory.

FIND (F)

```
FIND <string>
```

FIND searches through a program and displays all the statements containing the search string. If there are more occurrences than can be displayed on one screen, the to (CONTINUE) key can be used to display the next screenful until

no more occurrences are found and the command screen is returned to. Other valid keys are;

ESCAPE	- return to command screen
f7 (TOP OF SCREEN)	- edit from selected statement
Cursor up/down	- select statement for f7 or CTRL f0
CTRL f0 (MARK)	- mark selected statement
CTRL f1 (GOTO)	- edit from marked statement
f2 (TOP)	- edit from top of program
f3 (END>)	- edit from end of program

The rules for entering the string are the same as for the `EDIT` command.

Example

```
FIND PRINT TAB
```

FORE (FO)

```
FORE <coLour>
```

`FORE` changes the foreground colour of the edit screen. Available colours are black (N), red (R), green (G), yellow (Y), blue (B), magenta (M), cyan (C) and white (W).

Example

```
FORE W
```

GOTO (G)

```
GOTO
```

`GOTO` changes to the edit screen with the cursor on the marked statement at the top of the screen. If there is no marked statement an error message is displayed.

HELP

```
HELP
```

`HELP` lists all available commands with their syntax.

IE (I)

```
IE
```

IE creates a new statement at the end of the program, switches to the edit screen and displays the last page of the program with the cursor positioned on the new statement.

INFO (IN)

```
INFO
```

`INFO` displays information about the program currently in memory.

INSERT (INS)

```
INSERT
```

`INSERT` switches to insert mode for editing.

IT

```
IT
```

`IT` creates a new statement at the top of the program and switches to the edit screen with the cursor on the new statement. If necessary, the program is renumbered.

LABEL (LA)

```
LABEL
```

`LABEL` converts line number references to label names as long as they point to valid label statements. A label statement consists of REM followed by a valid label name. A label name begins with @ followed by any combination of alphanumeric characters and the characters I and _. If replacing a line number with a label name would make a line too long, an error is raised.

LOAD (L)

```
LOAD <program>
```

`LOAD` loads a program from the current filing system into memory. If the program is not legal BASIC or if it is too big to fit into the available memory, an error is raised. If the program is too big for the current editing mode, the mode is reset to mode 7 (which requires the least screen memory) and the message `No room - Mode reset` is displayed.

Example

```
LOAD MYPROG
```

MODE (M)

```
MODE <n>
```

`MODE` changes the edit screen mode. Valid modes are 0,1,3,4,6 and 7. If there is not enough memory to edit the program in the requested mode, an error is raised.

Example

```
MODE 3
```

NEW (N)

```
NEW
```

`NEW` is equivalent to the BASIC NEW command.

NOSCROLL (NO)

```
NOSCROLL
```

`NOSCROLL` disables scrolling on the edit screen for cursor movement. Instead, attempting to move the cursor beyond the screen boundaries causes screen wraparound.

NUMBER (NU)

```
NUMBER
```

`NUMBER` converts label references to line numbers as long as they are matched by valid label statements somewhere in the program. An error is raised if no label references are found, if replacing a label reference with a line number would make a statement too long or if there are any label references which are not matched by label statements.

OLD (0)

```
OLD
```

`OLD` is equivalent to the BASIC `OLD` command. If there is no valid BASIC program in memory an error message is displayed. If the program is too big for the current editing mode, the mode is reset to mode 7 (which requires the least screen memory).

OVERTYPE (OV)

```
OVERTYPE
```

`OVERTYPE` switches to overtype mode for editing.

QCHANGE (Q)

```
QCHANGE <string1> <string2>
```

QCHANGE searches through a program for all occurrences of <string1> and gives you the option of replacing each occurrence with <string2>. When a match is found, the edit screen is displayed with the statement at the top of the screen and the cursor positioned at the start of the string. Valid keys are;

Y	- replace string and display next match
N	- leave string unchanged and display next match
ESCAPE	- return to command screen
f7 (TOP OF SCREEN)	- edit from current statement

Example

```
QCHANGE TOP top
```

RENUMBER (R)

```
RENUMBER <n1>,<n2>
```

RENUMBER renumbers the program from <n1> in increments of <n2>. If only one argument is given, line numbers will begin with that number and increment in steps of 10. If no arguments are given, the program is renumbered from line 10 in steps of 10. If present, the increment must be in the range 1-255. If renumbering would cause a line number to exceed the maximum (32767), an error is raised. Any references to lines which do not exist will be replaced with the string @@@@ and an error message will be displayed.

Example

```
RENUMBER 1000,5
```

RUN(RU)

```
RUN
```

RUN returns to BASIC and runs the program in memory immediately.

SAVE (S)

```
SAVE <program>
```

SAVE saves the program in memory to the current filing system.

Example

```
SAVE MYPROG
```

SCROLL (SC)

```
SCROLL
```

`SCROLL` enables scrolling on the edit screen.

TAB (TA)

```
TAB <n>
```

`TAB` sets the distance between tab stops to a value between 0 and 80. The default value is 5.

Example

```
TAB 10
```

TOP (T)

```
TOP
```

`TOP` edits from the first line of the program.

Appendix B

Editing reference table

Editing takes place in the screen mode and colours which have been selected. The default colours are white text on a blue background (except in mode 7 where they are white text on a black background). Line numbers are displayed to the left of the text area and may not be edited. Only complete statements are displayed on the screen, blank lines are left at the bottom if necessary.

The screen scrolls as necessary to accommodate new or extended statements. An amended statement is only written to memory when you leave that statement. At this time, any unused screen lines are recovered and any BASIC keyword abbreviations are expanded. Before a statement is updated in memory, trailing blanks are removed.

The cursor keys

↑	Moves the cursor up one screen line. Scroll or wraparound occurs on the top screen line.
↓	Moves the cursor down one screen line. Scroll or wraparound occurs on the last statement line.
←	Moves the cursor left one character. Scroll or wraparound occurs at the beginning of the top screen line.
→	Moves the cursor right one character. Scroll or wraparound occurs at the end of the last statement line.
SHIFT ↑	Displays the previous page of the program with a single statement overlap.
SHIFT ↓	Displays the following page of the program with a single statement overlap.
SHIFT ←	Moves the cursor left two characters.
SHIFT →	Moves the cursor right two characters.
CTRL ↑	oves the cursor to the beginning of the first statement on the screen.

CTRL ↓	Moves the cursor to the beginning of the last statement on the screen,
CTRL ←	From the beginning of a statement, moves the cursor to the beginning of the previous statement. From the middle of a statement, moves the cursor to the beginning of that statement.
CTRL →	Moves the cursor to the beginning of the next statement,

Other keys

ESCAPE	Returns to the command screen.
RETURN	Creates a new statement immediately following the current one, Automatic renumbering may take place and if necessary, the screen will be scrolled. Editing continues from the new statement.
DELETE	Deletes the character to the left of the cursor. The cursor and all following text is moved left one position. If the cursor is already at the beginning of a statement, this key has no effect.
SHIFT DELETE	Deletes the character above the cursor and moves all the following text left one position. The cursor position remains unchanged.
TAB	Moves the cursor to the next TAB stop.
SHIFT TAB	Moves the cursor to the beginning of the next screen line.
COPY	Changes the cursor blink rate. The default cursor blinks at the normal rate. Pressing the COPY key once creates a non-blinking cursor. Pressing it a second time creates a fast-blinking cursor. Pressing it a third time returns to the default cursor.

Appendix C

Function key reference table

Key	Function
f0 (EXECUTE)	Executes line commands if any are present in the program.
f1 (lNSERT/OVERTYPE)	Switches between insert mode and overtype mode for editing. When in insert mode, a block cursor is displayed.
f2 (TOP)	Edits from the top of the program.
f3 (END)	Displays the last page of the program with the cursor positioned at the start of the last statement.
f4 (RENUMBER)	Renumbers the program starting at line 10 with an increment of 10,
f5 (CONTINUE)	Looks for the next occurrence of the search string in a `FIND` or `EDIT` command,
f6 (DELETE TO END OF LINE)	Deletes all the text from the cursor position to the end of the current statement.
f7 (TOP OF SCREEN)	Displays the edit screen with the current statement positioned at the top of the screen.
f8 (INSERT AT END)	reates a new statement at the end of the program and displays the last page of the program with the cursor on the new statement.
f9 (EXIT)	Clears the screen, returns to BASIC and issues a BASIC 0 LD command if there is a program in memory,
SHIFT f0 (NEW)	Equivalent to the BASIC `NEW` command,
SHIFT f1 (OLD)	Equivalent to the BASIC `OLD` command; editing continues from the top of the program.
SHIFT f2 (UNDO)	Undoes any changes made to the current statement and positions the cursor at the start of the line.

SHIFT f3 (SWAP CASE) Changes the character at the cursor position to upper case if it is lower case and vice-versa.

SHIFT f4 (EXTEND STATEMENT) Adds a blank screen line to the current statement if it is not already fully extended,

SHIFT f5 (MODE) Displays the edit screen in the next mode down. The order in which the modes are displayed is 7, 6, 4, 3, 1, 0, 7 ... If the program cannot be edited in the new screen mode, the mode is reset to mode 7 (which requires the least screen memory) and an error message is displayed.

SHIFT f6 (LABEL) Converts line number references to label names, as long as they point to valid label statements. If replacing a line number with a label name would make a line too long, an error is raised.

SHIFT f7 (NUMBER) Converts label references to line numbers, as long as they are matched by valid label statements somewhere in the program. If replacing a label reference with a line number would make a line too long, an error is raised.

SHIFT f8 (INSERT AT TOP)Inserts a new statement at the top of the program and continues editing from that statement.

CTRL f0 (MARK) Marks the current statement as the target for a subsequent GOTO. Any previously marked statement has its marker removed.

CTRL f1 (GOTO) Continues editing from the marked statement, If there is no marked statement the command has no effect,

CTRL f2 (SPLIT STATEMENT)Splits the current statement into two at the cursor position.

CTRL f3 (JOIN STATEMENTS)Joins the current statement and the following one to make one statement.

CTRL f4 (REPEAT) Creates a copy of the current statement immediately after it.

CTRL f5 (SCROLL ON) Enables scrolling for the edit screen.,

CTRL f6 (SCROLL OFF) Disables scrolling for the edit screen

CTRL f7 (BACKGROUND) For modes other than 7, this command sets the background colour for editing to the next one in sequence. The order of the colours is blue, magenta, cyan, white, black, red, green, yellow, blue ...

CTRL f8 (FOREGROUND) For modes other than 7, this command sets the foreground colour for editing to the next one in sequence. The order of the colours is white, black, red, green, yellow, blue, magenta, cyan, white ...

Appendix D

Error messages

When an error occurs, The BASIC Editor will return you to the command screen, if you are not already there, display an error message and bleep. All the error messages that can be raised by The BASIC Editor are listed below. Errors can also arise following a star command. For example, the command `*KEY ABC` will raise a `Bad key` error.

Argument missing

A command has been entered which needed additional data but this was not supplied. For example, entering `LOAD` without a program name would generate this error.

Bad number

A valid number was expected but the input was not numeric or was greater than 32767.

Bad program

The program in memory is not a valid BASIC program. The only commands valid are `NEW`, `LOAD`, `EXIT`, `INFO`, `HELP` and star commands. No editing facilities are available.

Destination missing

The destination has not been supplied for a move or copy line command.

Incompatible line commands

Two line commands have been entered but were found to be different. For example, inserting an `MM` command with a `CC` command would raise this error.

Incomplete Line command

The first line command of a block delete, move or copy is present, but the second is missing.

Increment must be in range 1-255

The increment for a `RENUMBER` command was outside the allowed range.

Invalid argument

A command has been entered with an argument where no arguments are allowed.

Invalid command

A command has been entered which is not recognised.

Invalid delete command

A destination has been supplied for a delete statement command or a destination of `A` or `B` was supplied for delete block command.

Invalid destination

The destination for a block move or copy lies within the block.

Invalid line number

A line number was entered from the command screen, but was followed by more text.

Line has been truncated

Keyword abbreviations resulted in a line that would have been too long to be edited so the line was truncated.

Line too long

The program contains a line which is too long to be displayed by The BASIC Editor. The line is also too long to be edited using the normal BASIC line editor, but if you return to BASIC, you will be able to list it. To edit the program in The BASIC Editor, you will have to shorten the line to 251 characters. To find out which line is in error, use the `INFO` command and refer to the current line number.

Line too long to be numbered

While executing a `NUMBER` command, the editor found a label reference which, when replaced by a line number, would make a statement too long. The statement should be made shorter and the `NUMBER` command repeated to convert the remaining label references to line numbers.

Line too long to contain label

While executing a `LABEL` command, the editor found a line which would be too long if a line number were replaced with a label reference. The statement should be made shorter and the `LABEL` command repeated to convert the remaining line numbers to label references.

Lines too long to be joined

An attempt has been made to join two statements whose combined length exceeds the maximum allowed.

Missing Line command

A destination has been provided, but no line commands have been inserted.

No labels reinstated

The `LABEL` command found no line numbers which could be converted into label references.

No marked Line

The `GOTO` command has been entered, but there is no marked line.

No references to labels were found

The `NUMBER` command has been used, but the program contains no label references.

No room

The program has used up all the available memory. Changing the editing mode may give you some more room.

No room for this mode

The `MODE` command has been used, but cannot be executed as the program is too big for the mode specified.

No room - Mode reset

A `LOAD`, `APPEND` or `OLD` command has been used, but the program is too big to be edited in the current mode. This error could also occur when the mode is changed on the edit screen. The mode is reset to mode 7 (which requires the least screen memory).

No search string set up

The f5 (CONTINUE) key was pressed on the edit screen when no `EDIT` or `FIND` command had been entered,

Only colours N, R, G, Y, B, M, C, W are valid

A `FORE` or `BACK` command has been entered with an invalid colour.

Only modes 0,1,3,4,6,7 are valid

The `MODE` command has been entered with an invalid mode.

Only two strings allowed

More than two strings were entered for a `CHANGE` or `QCHANGE` command. If spaces are to be significant, the prompted version of the command should be used.

Replace string missing

No replace string has been entered for a `CHANGE` or `QCHANGE` command.

Start line/Increment too big

The arguments supplied for a `RENUMBER` command cause the BASIC line numbers to exceed the maximum (32767).

String not found

Either no occurrences or no more occurrences of a search string have been found.

Tab stop must be in range 0-80

The `TAB` command has been used with a value outside the allowed range.

Target not found

The `RENUMBER` command has been used and one or more statements have been found containing references to line numbers which do not exist. The program has been renumbered, but all such references have been replaced with the string `@@@@`.

Too many arguments

More arguments have been entered for a command than are allowed.

Too many line commands

A line command for a single statement has been inserted, but a second command is also present.

Unknown error message

An error has occurred while executing a star command which is not available for display. This error occurs when other programs do not follow the recommended approach to error-handling.

Unresolved labels

A NUMBER command has been executed and one or more references have been found to label statements which do not exist.

Updates would make line too long

Replacing a search string with a replace string would make a statement too long to be edited. The current command is abandoned at this point,

Index

*BE 1
*BUILD 20
*EXEC 20
*FX 12

@@@@ 17

Abbreviations 5,22-28
Adding statements 2
APPEND 9,22
Argument missing 34

BACK 21,22
BACKGROUND 21,33
Bad number 34
Bad program 4,34
BASIC I 5
Bleep 34

CHANGE 15,16,22
Changing
    a line 2
    colour 21
    mode 21
Colour 29
    default 29
Command screen 1
Commands 4
CONTINUE 14,23,31
COPY 20,30
Copying text 13
Creating a program 2
CTRL R 8,11
Cursor 2,6,7,24
    keys 29

DELETE 11,30
DELETE TO END OF LINE 11,31
Deleting text 3,11
Destination missing 34

EDIT 14,16,23,31
Edit screen 1
Editing a program 2
    from a line 5
END 15,24,31
Error messages 34
ESCAPE 14,24,27,30
EXECUTE 12,13,31
EXIT 23,31
EXTEND STATEMENT 8,32

FIND 14,16,23,31
FORE 21,24
FOREGROUND 21,33
Function key card 1
Function keys 4,20,31
    f0 12,13,31
    f1 7,31
    f2 15,24,31
    f3 15,24,31
    f4 9,31
    f5 14,23,31
    f6 11,31
    f7 7,15,24,27,31
    f8 7,31
    f9 31
    SHIFT f0 31
    SHIFT f1 31
    SHIFT f2 31
    SHIFT f3 8,32
    SHIFT f4 8,32
    SHIFT f5 32
    SHIFT f6 18,32
    SHIFT f7 18,32
    SHIFT f8 7,32
    CTRL f0 8,15,24,32
    CTRL f1 8,15,24,32
    CTRL f2 8,32

CTRL f3 7,32
CTRL f4 32
CTRL rs 21,32
CTRL f6 21,32
CTRL tt 21,33
CTRL f8 21,33
CTRL SHIFT fn 20
GOTO 8,15,24,32
HELP 4,24
IE 7,24
Incompatible line commands 34
Incomplete line command 34
Increment must be in range 1-255 34
INFO 5,16,25
INSERT 25
INSERT/OVERTYPE 7,31
INSERT AT END 7,31
INSERT AT TOP 7,32
Inserting text 7
statements 7
Invalid argument 35
Invalid command 35
Invalid delete command 35
Invalid destination 35
Invalid Line number 35
IT 7,25
JOIN STATEMENTS 7,32
Joining programs 9
LABEL 18,25,32
Label references 10,17
Labels 17
Line commands 11
Line has been truncated 35
Line too long 35
Linetoo Long to be renumbered 35
Line too Long to contain label 35
Lines too Long to be joined 36
LOAD 25
Loading a program 4,6
MARK 8,15,24,32
Marking a statement 8
Missing line command 36
Mode 29
MODE 26,32
Moving text 12
NEW 4,26,31
No labels reinstated 36
No marked line 36
No references to labels where found 36
No room 36
No room for this mode 36
No room - Mode reset 25,36
No search string set up 36
NOSCROLL 20,26
NUMBER 18,26,32
OLD 4,26,31
Only colours N,R,G,Y,B, M,C,W are valid 36
Only modes 0,1,3,4,6,7 are valid 37
Only two strings allowed 37
Operating system command 19
OVERTYPE 26
Overtyping text 7
QCHANGE 15,27
RENUMBER 9,27,31
Renumbering 9
problems arising 10
REPEAT 32
Replace string missing 37
RETURN 30
ROM fitting instructions 1
RUN 3,27
Running a program 3

SAVE 27
Saving a program 3
Screen colours 29
Screen mode 29
SCROLL 21,28
SCROLL OFF 21,32
SCROLL ON 21,32
Scrolling 20
SHIFT DELETE 11,30
SHIFT TAB 30
SPLIT STATEMENT 8,32
Start Line/Increment too big 37
String not found 16,37
SWAP CASE 8,32
Syntax 4
TAB 7,19,28,30
Tab stop must be in range 0-80 37
Target not found 37
Too many arguments 37
Too many line commands 37
TOP 15,24,28,31
TOP OF SCREEN 7,15,24,27,31
UNDO 31
Undoing changes 5
Unknown error message 37
Unresolved labels 38
Updates would make line too long 38
Word processor 1

The BASIC Editor

for the BBC Microcomputer Model B

Inserting The BASIC Editor ROM

Your BASIC Editor chip may be placed in any spare 'sideways ROM' (or 'paged ROM') socket. These are located on the front right-hand side of the circuit board inside the BBC Microcomputer casing.

1. To get to the board, undo the four screws marked 'FIX', Two of these are underneath the computer, and the other two can be found on the back,

2. Once the top is removed, release the bolts holding down the keyboard assembly, These are located on either side of the keyboard. Some machines have two bolts, others may have three.

3. There is no need to disconnect the keyboard completely, so the multi-wire connector to the main board can be left in place. Carefully displace the keyboard, rotating it clockwise through about 20 degrees so that the front right-hand side is accessible,

4. Locate the row of five large sockets, The four right-hand sockets (identified on the board as IC52, IC88, IClOO, ICIOI) are sideways ROM sockets, The fifth from the right is the operating system socket (IC51).

Read the section overleaf about the operating priority of the sideways ROM sockets, and then insert the ROM as described in the section entitled 'Inserting the chip'.

Sideways ROMs - operating priorities

The four sideways ROM sockets have an operating priority, decreasing from right to left: on a hard reset, or when the computer is switched on, the language chip in the rightmost ROM socket takes priority over the others, So the position of The BASIC Editor ROM in relation to the BASIC ROM, for example, will determine whether your machine starts up in BASIC or in The BASIC Editor,

If you want to start up in BASIC and get to The BASIC Editor from there (use the command `*BE` RETURN) then you must insert your BASIC Editor chip to the left of the BASIC language chip. If the BASIC chip is in IC52, ie fourth from the right, then you have to move it further to the right, to leave a lower priority socket for The BASIC Editor chip.

Those who use the computer primarily with The BASIC Editor can have it operating as soon as the machine is turned on by inserting the chip so that it is the furthest chip to the right of every other sideways ROM present.

If you are replacing the operating system ROM at the same time as fitting The BASIC Editor, then follow the same insertion procedure, and fit the chip in the socket marked IC51 (fifth from the right).

Inserting the chip

1. Before taking the chip out of its protective packaging, identify Pin 1 on the chip, It is either marked with a dot on the top, in the corner of Pin 1, or the half-moon notch at one end of the chip identifies the end of the chip nearest Pin 1. Pin 1 should be on the left if the notch is held up.

2. Hold the ends of the chip between finger and thumb, and line up all the pins over the destination socket. Pin 1 and the half-moon notch should point towards the back of the computer casing,

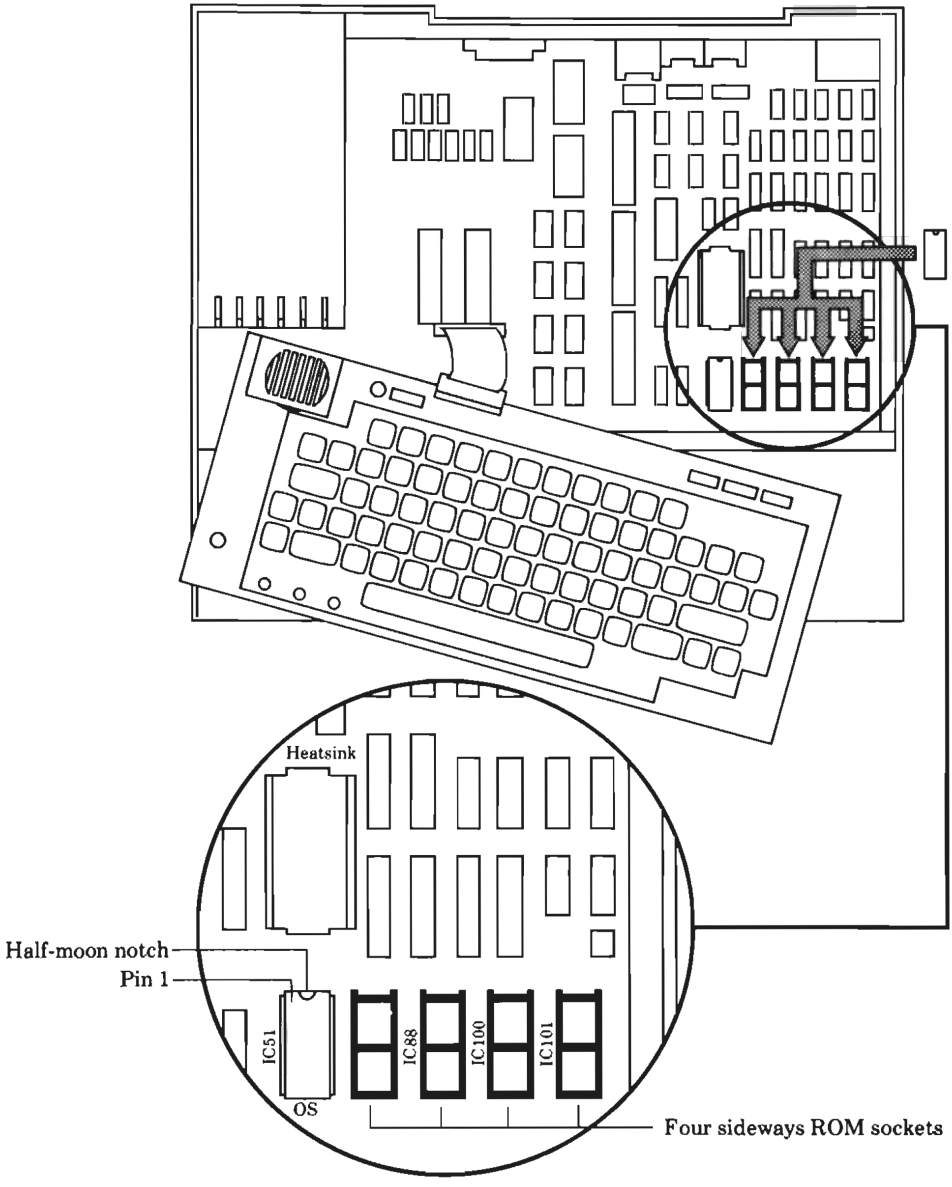
3. Now apply firm pressure to the chip, but try not to force it! When the chip is in, it appears to be slightly raised. Check that all the pins do enter the socket, and that none are bent out, or underneath.

Removing chips

To avoid bending any pins a chip must be removed very carefully. Take a screwdriver or similar tool and gently prise up each end, a bit at a time.

Inserting The BASIC Editor ROM

This diagram shows a plan view of the BBC Microcomputer. The top of the computer casing has been removed to reveal the four sideways ROM sockets. The BASIC Editor ROM can be inserted in anyone of these sockets.



ACORNSOFT
Acornsoft Limited, Betjeman House, 104 Hills Road,
Cambridge CB2 1LQ, England. Telephone (0223) 316039

The BASIC Editor

on the BBC Microcomputer Model B

Acornsoft Limited, Betjeman House, 104 Hills Road,
Cambridge CB2 1LQ, England. Telephone (0223) 316039

Note: British Broadcasting Corporation has been abbreviated to BBC in this publication.

This manual was written by Morrell Media Services, Reading, England.

ISBN 0 907876 18 8

SBD29