# 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 missed. 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 f5 (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 f5 (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 ' 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 (O)

```
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
```

0**VERTYPE** switches to overtype mode for editing.

## 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 test on a blue background (except in mode 7 where they are white test on a black background). Line numbers are displayed to the left of the test 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 ↑ | Moves 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 (INSERT/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) | Creates 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 **OLD** 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 (GOT0) 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.

**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.