Wordwise Plus

REFERENCE MANUAL

by JOHN COLL and ANDREW MYERS

Computer Concepts

Contents

Written by John A. Coll and Andrew W. Myers

Wordwise and Wordwise Plus are designed and distributed by Computer Concepts.

Both manuals were typeset direct from Wordwise files by Quorum Technical Services Ltd., Cheltenham.

The abbreviation BBC Micro for British Broadcasting Corporation Microcomputer has been used throughout this book.

1. Introduction

In the Wordwise-Plus box you should have one of each of the following

Wordwise-Plus Eprom chip
Wordwise-Plus Introductory Booklet
Wordwise-Plus Reference Manual
Wordwise-Plus program cassette
Wordwise-Plus registration card

The accompanying booklet "An introduction to Wordwise-Plus" explains how to get started with the word processor. This manual serves a different purpose – it is a reference manual for you to use as you become more experienced with Wordwise-Plus.

Since it is a reference manual the index is critical and we have tried to make it comprehensive. Wordwise-Plus is a flexible piece of software and if you are to become aware of all its many features then you will need to flick through the manual often. But don't think that you are meant to read it from cover to cover – there is too much information for that approach. Read up sections as the need arises.

As well as the Introductory booklet and this manual, a cassette is also supplied. This cassette contains a number of useful programs – some of which are quite complicated and designed for the experienced user. Don't lose the cassette even if you are a disc user – you will need it!

This version of Wordwise-Plus has been designed to load and save files to cassette, disc or network.

Versions of Wordwise-Plus contain serial numbers and action will be taken against any person or organisation found to be infringing the copyright of this product.

It is essential that you register your ownership of Wordwise-Plus by returning the enclosed registration card.

2. Menus

The Main Menu
When you first start up Wordwise-Plus you will be presented with a list of options. This is called the Main Menu. From the Main Menu you can load and save your text and can tell the computer to print it out.

The Text Area
If you press the key marked **ESCAPE** on the keyboard the menu will go away and you will see your text, between the words "Start" and "End". This is called the text area. Anything that you type in will become part of your document. To return to the Main Menu you just press **ESCAPE** key again.

The **ESCAPE** key moves you back and forth between the Main Menu and your Text Area.

Option 9 of the Main menu will move you to the Segment Menu.

The Segment Menu
From the Segment Menu you can load and save segments as well as printing them out. There are 10 segments and they are used to store programs or text. Menu option 5 is used to select another segment for editing.

The Segment Areas
If you press the key marked **ESCAPE** on the keyboard the menu will go away and you will see a segment, between the words "Seg 0" and "Endseg". This is called Segment 0. Anything that you type in will become part of that segment. To return to the Segment Menu you just press the **ESCAPE** key again. In this case the **ESCAPE** key moves you back and forth between the Segment Menu and a Segment Area.

Option 9 of the Segment menu will move you to the Main Menu.

As well as the options displayed on the Main 9 segment Menus there are two "invisible" options "*" and ":".

* will permit you to type in any operating system command – just as you can from Basic. This must be used with care. Some commands will destroy the text that you have in memory!

: will enable you to use all the Wordwise-Plus commands immediately. All these commands can also be used to write programs which are held in the Segments.

Whilst most of the Main Menu options are quite obvious there are a few points which it may be helpful to record for completeness.

1) Save entire text
If there is no text in memory, Wordwise-Plus will not attempt to save the file. This will prevent you accidently replacing a file on disc with an empty document.

When attempting to save text on disc, Wordwise-Plus first checks to see if a file of that name already exists on disc. If it does then you will have to confirm that you wish to save the file, since doing so will delete the file previously held on disc by that name.

When text is saved all markers are deleted before it is saved.

2) Load new text
If you attempt to load a text file into the computer while there is already one there you will delete the file in memory. To prevent this happening by accident you will have to confirm your command in this case.

If there are any characters in the text that you load in that Wordwise-Plus regards as "illegal" then these characters will be replaced with "‖". This character is chosen since it is the Pad Character and will be printed as a space. Legal characters are ASCII 32 to 126 and 221 which represents the **TAB** character.

3) Save marked text
If you have markers around a section of text you can choose to save the marked area only.

4) Load text to cursor
This will let you load any text without the computer first deleting the text in memory. It inserts the file being loaded into the text in memory at the cursor position.

5) Search and Replace
If you wish to search for special characters such as **RETURN** or **TAB** or the codes used in memory to represent *green* **f1** or *white* **f2** then you should use the following characters:

Search for:	use
TAB	‖T
RETURN	‖R
f1 *green*	‖G
f2 *white*	‖W

Note that these codes do not follow the usual BBC Microcomputer conventions. The letters have been chosen as mnemonics for the operations.

The 'hash' character (#) can be used as a *wildcard* symbol. When entered into the search string, a hash character will match with *any* character at all. Several hash characters may be used together or separately within a search string. By this means it is possible to find several different strings matching only a few known characters. For instance, the two words "sat" and "sit" could be represented generally by the string "s#t".

The above symbol conventions also apply to the SEARCH and REPLACE commands in the WORDWISE PLUS language.

6) Print text
This will print a formatted version of the text in memory. If some of the text is surrounded by markers then the computer will ask if you wish to print the marked section only.

7) Preview text
This will let you preview the formatted output without printing the text. The preview will occur in an 80-column screen mode, provided that there is sufficient memory left available. If less than about 15500 'Characters free' are left, then the preview will occur in 40-columns. This is due to the limitations of a standard BBC micro model-B, which requires a large amount of memory in order to achieve an 80-column display mode. There are some ways around the problem. Using a 6502 second processor will avoid the problem totally. There are also several 'screen RAM' add-on boards available which overcome this limitation on the standard machine.

8) Spool text
This will enable you to save a fully formatted version of the text on disc or cassette.

This formatted version of your document can then be *TYPEd or sent over electronic mail.

If you attempt to *TYPE or electronic mail the SAVEd form of a document then the embedded control codes and lack of RETURN characters may well cause problems.

9) Segment menu
This option will select the segment menu. Option 9 of the segment menu allows the user to return to the main menu.

3. Control Keys

Wordwise-Plus has two main modes of operation, known as Menu mode and Edit mode. When entering for the first time the main menu is displayed which shows a list of ten options that the user can choose. Nine of these have a number beside them and can be selected by simply pressing the number of the option required.

The last option on the list says:

```
ESC Edit Mode
```

If the **ESCAPE** key is pressed then you will enter the editing mode. Press the key again and you will be returned back to the main menu – Menu Mode. It is in Edit Mode that you enter your text into your main document or into any of the ten segments and then chop and change it.

The top line of the Edit Mode display, known as the status line, shows how many words have been typed into memory so far, and on the right, how many more characters can be typed in before the memory is full.

To enter text you can simply type at the keyboard as if it were a typewriter. There is however one exception in that, when the end of the line is reached, there is no need to press the **RETURN** key to start the new line. This is done automatically in Wordwise-Plus. The only time that the **RETURN** key should be pressed is if you want to end a paragraph or want to deliberately force the end of a line of text. Indeed it is important that the **RETURN** key is not pressed at the end of each line as it will affect the layout of the final document when printed.

The following pages contain a list of all the editing commands that are available in Edit Mode.

Using the cursor keys

→
Moves the cursor one position to the right, unless it is already at the end of the line, in which case it will move to the start of the next line down.

←
Moves the cursor left one character unless it is already at the start of the line, in which case it will move to the end of the line above.

↑
Moves the cursor up one line.

↓
Moves the cursor down one line.

The following cursor controls can be used by holding down the CTRL key and pressing the appropriate arrow key.

CTRL →
Moves the cursor right one word, leaving it under the first character of the next word in the text.

CTRL ←
Moves the cursor left one word, leaving it under the first character of the previous word in the text.

CTRL ↑
Moves the cursor up the text by 23 lines. This is one line less than a whole screenful and so allows rapid 'paging' through the text.

CTRL ↓
Moves the cursor down by 23 lines or one screenful.

The following cursor controls can be used by pressing the appropriate arrow key whilst also holding down the SHIFT key.

SHIFT →
Moves the cursor to the last character on the current line. Please note however that this does not always move the cursor to position 40 on the line, but to the last character. Because Wordwise-Plus normally displays lines on the screen without splitting words between the end of one line and the beginning of the next, the last character on the line is often not at the last possible position.

SHIFT ←
Moves the cursor to the first position on the line.

SHIFT ↑
Moves the cursor to the very start of any text stored in memory.

SHIFT ↓
Move the cursor to the last position in the text.

Other useful editing keys

CTRL A
Deletes the character at the cursor position. This is different from the delete key which always deletes the character immediately to the left of the cursor.

CTRL S
Swaps the case of the character at the cursor from upper case to lower case or vice versa. This will only affect the alphabetic characters A to Z.

CTRL D
Deletes the entire word at the cursor. Wordwise-Plus recognises a word as any collection of characters separated by spaces including punctuation and numbers.

CTRL F
Changes the on-screen formatting. Normally Wordwise-Plus does not split words between the end of one line and the start of the next, instead it transfers the whole word to the line below. There are occasions however when this can be a nuisance and so **CTRL F** can be used to prevent this on-screen formatting.

This actually re-displays the screen from the position of the cursor onwards. Pressing **CTRL F** again will switch the on-screen formatting back on.

CTRL R
Deletes all markers in your text. See Section 4 regarding use of the **f3** key for more information on markers.

CTRL W
Causes the computer to recount the total number of words in the text, setting W% and changing the word count in the corner of the screen. This performs exactly the same operation as the command `RECOUNT`.

DELETE
Deletes the character to the left of the cursor. If the cursor is at the start of the line then it will delete the last character of the previous line.

TAB
When pressed this enters a **TAB** code into the text which is shown as a small right arrow. This will then cause spaces to be output until the next defined tabs stop when printing or previewing. See the `DT` embedded command.

4. Function keys

The red function keys at the top of the keyboard can be used in three different ways in Wordwise-Plus.

A Pressing just the function key

B Pressing the function key while the Shift key is held down

C Pressing the function key while both the CTRL and Shift keys are held down

A – Pressing the function keys on their own.
If these keys are pressed on their own then this will activate one of the ten functions as listed on the key-guide supplied with the Wordwise-Plus package. These are explained below.

f0 Insert or Over
This changes between insert and overwrite modes. On the right of the status line, after the character count, there is displayed either a capital `I` or `O` telling you which mode you are in. Normally this would be in insert mode so that all text typed at the cursor. is inserted between the two characters on either side. If there is no text after the cursor then this can be thought of as inserting characters between the text and the red "`End`" marker.

f1 Green
This inserts a green code at the current cursor position and marks the start of an embedded command. All embedded commands are entered in green by first pressing this key then entering the command followed by **f2**.
See the chapter on embedded commands for more details.

f2 White
This is used to mark the end of embedded commands. See **f1** and the chapter on embedded commands.

f3 Markers
Pressing this will insert a marker into the text at the position of the cursor. They appear as white blocks in the text and are used for manipulating whole sections of text. Only a maximum of two are allowed at any one time, trying to insert more will cause the computer to beep and flash the message "MARKERS" on the top line. A flashing red square is displayed on the right hand end of the status line to indicate that a marker has been set. All text between the markers is referred to as the marked section.

f4 Move cursor to?
Once pressed this key will ask you to enter a character. The cursor will then move down the text until it finds the next occurrence of this character. As well as the ordinary textual characters, you can also move to a marker or even the green and white characters. This can be done by pressing f4 followed by either f3, f1 or f2.

f5 Word count to?
Like the previous key this will ask you to enter a single character. Once entered, Wordwise-Plus will then count the number of words between the current position and the first occurence of this character, showing the result on the left of the status line. The total count for the whole text will now be lost but can be recovered using the command RECOUNT.

f6 Delete to?
Like the previous two functions this waits until you type a character and then searches for its first occurrence. Now, however, all the text between the current cursor position and this point is deleted. If more than around 250 characters are to be deleted then it will ask if you are sure before carrying it out.

f7 Delete Marked Text
f7, f8 and f9 are all concerned with marked sections of text. This key will delete both of the markers and all of the text between them. See f3 for a description of a marked section.

f8 Move marked text
This will move your marked section of text to the position of the cursor. The cursor will be left at the start of the section that has just been moved and both of the markers will be deleted.

f9 Copy marked section
Like **f8** this will insert your marked section of text at the position of the cursor. But unlike **f8**, **f9** merely makes a copy of the text and does not remove it from its original position. The markers are not removed from the text so that you can make multiple copies of the same section.

If when trying either **f7**, **f8** or **f9** the computer beeps and the message 'Markers' flashes at the top of the screen, this indicates that there has been an error. This could be because there is not a marked section to work with or that the cursor is currently between the two markers. It is not possible to move a marked section into the middle of itself!

B – Using the function keys while holding down SHIFT

Holding down **SHIFT** and pressing a function key makes the computer try to run a program in the segment corresponding to the function key pressed. If there is nothing in the segment then nothing will happen. If there is text in it, rather than a program, then an error will occur.

Running programs in this way will also work when either of the menus is on the screen.

C – Using the function keys with both CTRL and SHIFT held down.

As in `BASIC`, you can program the function keys yourself using the `*` command, `*KEY`. The major difference is that to operate these in the same way you must hold both **CTRL** and **SHIFT** down at the same time as pressing the function key. For example, suppose we program function key 4 from a menu by typing

```
*KEY 4 Chapter
```

If we now go into Edit Mode by pressing the **ESCAPE** key and then press **CTRL**, **SHIFT** and **f4** together, the word "`Chapter`" will be inserted in the text.

Special codes representing *green* and *white*, etc. may be used within a function key definition. These codes are NOT the same as those used with **FIND** and **REPLACE** commands. The codes to be used are listed below:
||!! – represents **f1**
||!" – represents **f2**
||I – represents **TAB**
||M – represents **RETURN**

For instance, to define a key to produce **f1** TI5 **f2** the sequence would be

*KEY 1 ||!!TI5||!" **RETURN**

5. Embedded Commands

In the previous chapter the effect of pressing various keys has been described. This chapter concentrates on "embedded commands" – that is commands which you embed in the document being written. These embedded commands control the layout of the document when it is printed but have no real effect on the text as it is being typed.

There are many embedded commands but you will probably only use about 8 or 10 of them regularly. Each person uses a word processor in a different way and it is doubtful if anyone will learn to use all the commands on each document. So don't be put off by the apparent complexity of some of the descriptions – just ignore the problems!

These are the embedded commands that you will probably use most often:

`TI` – causes a temporary indent, for example the first line in a paragraph is usually indented about 8 spaces using the command **f1** `TI8` **f2**

`CE` – causes a line to be automatically centred on the page.

`EP` – tells the computer to break your document into pages.

In fact you can safely ignore the rest at first.

As mentioned above, these commands are embedded in your text and you need some way to tell the computer to treat these two letters as commands rather than to print them in your document. Function key **f1** is used to mark the start of an embedded command and the function key **f2** or the **RETURN** key is used to mark the end of the command.

The use of these commands is described briefly in the Introduction manual. However, there are a few additional rules to follow.

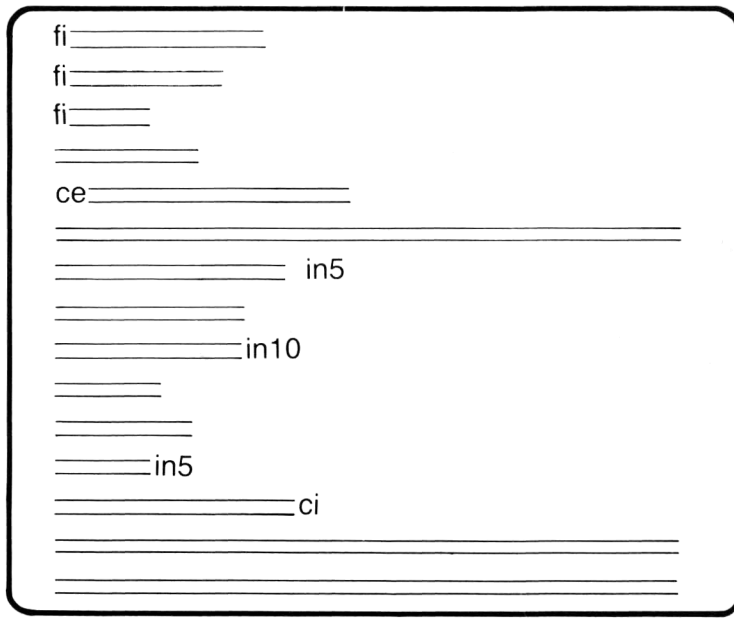
The start of an embedded command is marked with an **f1**. This is followed by the letters and numbers of the command itself. The command is ended with an **f2** or a **RETURN**. Remember this **RETURN** is ignored when printing the text. It is used here only as an 'end-of-command' code.

If an embedded command is to be directly followed by another embedded command then there should be no **f2** between the commands. The last command on the line however must end as normal with an **f2** or **RETURN**.

Embedded commands must never overlap the end of the screen line. If the embedded command is split so that part is at the end of the screen line, and part at the start of the next line, the command will not work. This situation should not arise very often but, if you are putting a string of commands together on one line, it makes sense to end some of them with **RETURN** instead of **f2**.

Other embedded commands control the size of the left margin (`LM`) – e.g. `LM10` gives a left margin of 10 character positions – about 25 mm. The diagram on the following page shows a selection of commands that control the text layout, and the result after formatting.

The pages which follow explain each of the embedded commands. They are listed in alphabetical order for quick reference.


fi
fi
fi
ce
in5
in10
in5
ci

'STAR' command

Examples

f1 *cat f2
f1 *TYPE fred f2

The * embedded command can be placed in text and the whole of the rest of the line, up to the next f2 or RETURN will call the operating system in the usual way. This allows commands to be passed to other ROMs in the machine at the correct place in the text, during formatting.

The resulting output will be sent to the printer or screen. Note that Wordwise-Plus is unable to keep track of the number of lines sent to the printer by the operating system or other ROM and, therefore, will be unaware of the position on the printed page. If paging has been enabled, with the EP command, the rest of the text from the document in memory may well be incorrectly placed on the printed output.

However previewing the output before printing will enable the user to make necessary adjustments.

When another ROM is called into action, Wordwise Plus no longer has control. If the ROM command sends information to the printer, then Wordwise Plus will not know that it has occurred and cannot prevent it. 'Star' commands are issued during preview and any direct printer control would still occur. This may make the machine 'hang' until ESCAPE is pressed if the printer is not switched on. DO NOT press the BREAK key in such an instance, or the text in memory may be lost !

BP Begin Page

Examples

f1 BP f2
f1 bp f2

This command unceremoniously forces the computer to print out the current line, so far assembled, and to start a new page together with footers and headers that have been defined with the DF and DH commands.

The BP command only works if paged output has previously been requested with the Enable Paging (EP) command.

A BP command placed as the very last entry in the text will ensure that the printed output finishes at the bottom of a page. If there is no BP command then the printer will be stopped in the middle of a page – when it runs out of text.

Note: when ending a document the f1 BP must not be followed by anything else – not even an f2 or a RETURN, otherwise the heading for the following page will be output.

BS Bottom Space

Examples

f1 BS 7 f2
f1 bs 0 f2
f1 bs E% - 1 f2

If you have told the computer to split your document into pages, by placing an EP command at the start of your text, then you may wish to alter the number of blank lines at the bottom of each page. This Bottom Space, as it is known, is set to 6 lines by default.

The total number of lines on a page is set by PL and the top and bottom spaces are controlled by TS and BS. The defaults are PL66, TS6 and BS6 which means that each page contains 66-12, that is 54 printed lines of text. Remember that these commands only work after an Ep command has been issued.

BS may have any value in the range zero to 50. Values outside this range are taken as BS6.

A page footer may be placed in the Bottom Space using the Define Footer (DF) command. There is now a default footer equivalent to

f1 DF f1 CE f2 Page f1 pp f2

which prints the word "Page" and the current page number, centred. If the default footer is not required, it may be removed by defining a blank footer

f1 DF f2 RETURN

Any other footing definition will simply replace the default version.

CE Centre

Examples

f1 CE f2 NOTICE
f1 ce2 f2 The annual General Meeting
of the Young Farmers Club

This command will centre one or more lines of text. The text to be centred should follow the f1 CE f2 command on one or more lines.

To centre more than one line you have to press the RETURN key at the end of each line – otherwise the computer will attempt to fill lines with as many words as possible before centring them. The number of lines to be centred must immediately follow the f1 CE f2 command, in *green*. Thus

f1 ce5 f2 Computer Concepts RETURN
Gaddesden Place RETURN
Hemel Hempstead RETURN
Herts RETURN
HP2 6EX RETURN

will print as

Computer Concepts
Gaddesden Place
Hemel Hempstead
Herts
HP2 6EX

Note: when centring underlined text, it is necessary to separate the underlining command (US) from the centre command with a space, in order to prevent underling all the way from the left margin. This makes the underling command part of the text to be centred. Use the commands thus

f1 CE f2 f1 US f2 centred and underlined f1 UE f2

CI Cancel Indents

Examples

f1 c i f2
f1 C I f2

This command cancels any indents that have been set up with the `IN` command. After the `CI` command all text will start at the left margin. `CI` has exactly the same effect as `IN0`. The CI command takes effect at the start of the next printed line. For example

f1 I N 5 f2 L i n e 1 RETURN
Line 2 RETURN
Line 3 RETURN
Line 4 f1 C I RETURN
Line 5 RETURN
Line 6 RETURN

produces

```
      Line 1
      Line 2
      Line 3
      Line 4
Line 5
Line 6
```

CO Continuous Output

Examples

f1 C O f2
f1 c o f2

The computer can print your text in two quite different ways:- either split into pages, or as one long continuous piece.

The **C O** command cancels paging and all its associated commands such as Define Header (**D H**), Define Footer (**D F**), Top Space (**T S**), Bottom Space (**B S**), Page Length (**P L**), Page Number (**P N**), Conditional Page (**C P**). To use any of these commands you must have the Enable Paging (**E P**) command in force and not the Continuous Output (**C O**) command.

By default the computer will assume that you want Continuous Output.

CP Conditional Page

Examples

f1 CP 10 f2
f1 cp S% f2
f1 cp15 f2

When you have enabled paging with the Enable Page (EP) command then the computer will automatically start new pages. If you wish to ensure that a table or paragraph is not split over a page boundary, then you can use the Conditional Page command to force a new page if there is insufficient space left on the present one.

For example, if you have a 10 line table of results you would be wise to precede the table with the command

f1 CP11 f2

which will force a new page if there are less than 11 lines left on the page. The CP command does take account of the Page Length (PL) and Bottom Space (BS) settings.

DE Double-strike End

Examples

f1 d e f2
f1 D E f2

If you have issued the Double-strike Start (**DS**) command to cause the printer to double strike all characters, then you should use the Double-strike End (**DE**) command to return the printer to its single-strike mode.

The command sends a series of control codes to the printer. It is set up to operate with the Epson Rx-80 and related printers. The sequence that it generates by default is 27,72 or <**ESC**> H as it is sometimes shown in printer manuals.

If you wish to generate a different control sequence for another printer than you can use the "**RPS 3**" command to change the codes produced by the **DE** command.

By default, **DE** has exactly the same effect as
Print Sequence 3 (**OPS3**). or
f1 **OC27,72** f2. or
f1 **ES"H"** f2

When previewing in 80 column mode, text which the printer will double-strike is shown in inverted video (black lettering on white background).

DF Define Footing line

Examples

f1 D F f1 C E f2 C h a p t e r 1 RETURN
f1 d f f1 c e f2 C h a p t e r 1 . P a g e f1 p p f2 RETURN

You may define a line of text that will be placed at the foot of every page in the Bottom Space. By default the Bottom Space is set to 6 lines and the footer is placed 3 lines into the Bottom Space. The Bottom Space allocation can be changed with the BS command and the position of the footing within the bottom space may be set with the Footing Position (FP) command.

All the text and commands following DF, up to the next RETURN, will be treated as part of the footer line. Only one footing line may be printed, multiple line footers are not possible.

As you will see from the first example, the footer can be centred by including the CE command. The sequence

f1 D F f1 C E f2 C h a p t e r 1 RETURN

will place the text "Chapter 1" in the centre of line 4 (the default Footing Position) at the bottom of every page.

If you wish to place the page number in the footer then use the print Page number (PP) command. The sequence

f1 D F f1 C E f2 S e c t i o n 1 — p a g e f1 P P f2 RETURN

will print

Section 1 — page 1

as a footer to each page. The page number will be automatically incremented for each new page.

DH Define Heading line

Examples

f1 DH f1 CE f2 Chapter 1 RETURN
f1 dh f1 ce f2 Chapter 1. Page f1 pp f2 RETURN

You may define a line of text that will be placed at the head of every page in the Top Space. The default Top Space is 6 lines and the heading is placed on line 3 in the top space – though this may be changed with the Heading Position (HP) command. A heading will only be printed if paging has been enabled with the EP command.

All the text and commands following the DH command will be treated as part of the header.

As you will see from the first example the header can be centred by including the CE command. The sequence

f1 DH f1 CE f2 Chapter 1 RETURN

will place the text "Chapter 1" in the centre of line 3 (the dafault Heading Position) at the top of every page.

If you wish to place the page number in the header then use the print Page number (PP) command. The sequence

f1 DH f1 CE f2 Section 1 — page f1 PP f2 RETURN

will print

Section 1 — page 1

as a header to each page. The page number will automatically increment for each new page.

You may, or may not, wish to have the header printed on the very first page of your document. If you do want the header on page one then make the very first line of your text contain the

Enable Paging (**EP**) command and the Define Header (**DH**) command – thus

f1 EP **f1** DH **f1** CE **f2** Chapter 1 **RETURN**

On the other hand if you wish Wordwise-Plus to omit the header on the first page then just Define the Header when you have printed at least two lines. This way Wordwise-Plus will not know about the header until it is too late to print it on page 1!. e g.

f1 EP **RETURN**
RETURN
RETURN
f1 DH **f1** CE **f2** Chapter 1 **RETURN**

DM Disable Message "paper"

Examples

f1 D M f2
f1 d m f2

If the command Enable Message "**E M**" command has been issued together with the Enable Paging (**E P**) command then the printer will stop at the bottom of every page. This enables the user to insert a new sheet of paper, or simply check each one as it is printed. The computer "beeps" and prints a message on the screen. Simply press a key when you want it to continue.

The Disable Message (**D M**) command disables this mode of operation.

DP Define Pound sign

Examples

f1 DP35 f2
f1 dp96 f2

There is no international agreement on the code needed to make a printer print the "pound sterling" sign (£).

As a result you may wish to alter the code sent to the printer when the computer finds a £ sign in your text. By default Wordwise-Plus will send a code 96.

To produce £ signs on an Epson Rx-80 you will need to select the U K. character set on the printer. If your printer is not already set up for this character set then you should place

f1 ES "R",3 RETURN

at the start of your text to select the correct character set on the printer.

Then you should tell Wordwise-Plus to generate the code 35 whenever it finds a £ sign to be printed. On Epson printers the code for a pound sign is 35, so use the command

f1 DP 35 RETURN

to achieve this. Of course the two commands can be joined together (and need only occur once near the start of the text) as

f1 ES"R",3 f1 DP35 RETURN

Note that these codes apply only to Epson and 'Epson compatible' printers.

DS Double-strike Start

Example

f1 DS f2 Wordwise Plus f1 DE f2 RETURN

This command will cause certain printers to double-strike characters thus making them appear darker. It should be used in conjunction with the Double-strike End (DE) command, which turns off the effect. It provides a way of emphasising text.

The command sends a series of control codes to the printer. It is set up to operate with the Epson Rx-80 and related printers. The sequence that is generated by default is 27,71 or <ESC> G as it is sometimes shown in printer manuals.

If you wish to generate a different control sequence for another printer then you can use the RPS 2 command to change the codes produced by the DS command.

By default, DS has exactly the same effect as
Print Sequence 2 (OPS2). or
f2 OC27,71 f2 or
f1 ES"G" f2

When previewing in 80 column mode, text which the printer will double-strike is shown in inverted video (black lettering on a white background).

See also the DE command.

DT Define Tab-stops

Examples

f1 DT10,22,35,44,60 f2
f1 dt A%,B%,C%,D% f2

When you press the TAB key you will see an arrow appear on the screen, like this →. This arrow indicates that you have placed a Tabulation character in the text. When the text is printed out Wordwise-Plus will, when it meets a Tab character, print out spaces until it is on one of the pre-defined tab positions. By default these are column positions 10, 20, 30, 40 etc.

Thus the text

→abc→de→34
→q→2→456

will print out as

abc	de	34
q	2	456

Notice that the use of the TAB character has enabled columns to be easily produced.

If you wish to use different column TAB settings then they can be set with the DT command. The DT command is followed by a list of numbers which give the new tab-stop settings, as in the examples. The list of tab stops may not continue over one screen line. All previously set tab-stops are cancelled.

EM Enable Message

Examples

When you are printing you will probably normally use continuous paper – often called fan-fold paper. After printing you can tear the separate sheets apart along the perforations.

The alternative is to use separate sheets of paper. In this case it is essential that the computer pauses at the bottom of each page to give you time to insert the next sheet of paper. This command forces the computer to pause – and to help draw your attention, it makes a beep and prints a message on the screen. Simply press a key when you want it to continue with the next page.

To disable the pause and message use the Disable Message (**DM**) command. The default condition is that no messages are shown and fan-fold paper is assumed – the computer keeps printing one page after the other.

EP Enable Paging

Examples

f1 E P f2
f1 e p f2

This command tells the computer to split the printed output into pages, rather than producing one long, continuous, page.

When you first enter Wordwise-Plus the computer is set to produce a single document without page breaks. If you wish to

break the text up into pages or
use page numbers or
use page footers or
use page headers

then you must have the E P command at the top of your text – normally as the very first thing in the file and immediately followed by any Define Header command -e g.

f1 E P RETURN
f1 d h f2 C h a p t e r 1 RETURN

To resume Continous Output (no page breaks) you can use the C O command.

ES Escape Sequence

Examples

f1 ES"H"f2
f1 ES,A%,&55,"R",20f2

Many printers require a sequence of characters called an "Escape sequence" to select, for example, different character fonts. The ES command helps the user to send such sequences to the printer. It performs the same function as the OC command, except that it sends an Escape code automatically at the start of the sequence.

f1 ES 52f2 would send two codes to the printer: ASCII 27 followed by ASCII 52. On the Epson RX-80 this sequence would select italic printing. It is equivalent to

f1 OC 27,52f2 except that it is quicker to type.

f1 ES "R",8f2 would produce three characters 27,82,8. On the Epson Rx-80, this would select the USA character font.

Note that the codes can be given in various ways:

character – e g. "R"
decimal – e g. 10
hexadecimal – e g. &34
binary – e g. %10101110
octal – e g.@377

The first code can be placed next to the ES but subsequent codes should be separated with commas:

f1 ES"D","E","M","O",32,&31,@171f2

Note that, as usual, the command line must not extend beyond the end of the line.

FI Fully Indent

Example

f1 FI f2 address

Text normally clings to the left hand margin. There are occasions where it would be useful to make it cling to the right hand margin and the Fully Indent (FI) command does this for a single line of text. For example the text

f1 fi f2 Computer Concepts RETURN
f1 fi f2 Gaddesden Place RETURN
f1 fi f2 Hemel Hempstead RETURN
f1 fi f2 Herts. HP2 6EX RETURN

is printed as

```
Computer Concepts
  Gaddesden Place
  Hemel Hempstead
    Herts. HP2 6EX
```

Note that the FI command takes all text up to the next RETURN as the text to be acted upon.

Contrast the above to the layout produced by the IN5Ø command:

```
Computer Concepts
Gaddesden Place
Hemel Hempstead
Herts. HP2 6EX
```

FP Footing Position

Examples

f1 FP 3 f2
f1 fp5 f2
f1 FP A% f2

When a footing has been defined with the DF command you may wish to have the footing printed out at some position in the bottom space other than the default (4th line). You can alter the position of the footing within the Bottom Space by use of the FP command.

This command will only have an effect if:
firstly paging has been enabled with an EP command and secondly a footing has been defined with a DF command (a default footing will be used unless deliberately cancelled).

FP can be followed by a number between one and the number of lines defined as the Bottom space (BS). If the value is larger than BS the footer will not be printed.

GF Get File

Examples

```
f1 GF"tempfil" f2
f1 gf A$ f2
```

When a document is being printed it is sometimes convenient to be able to pull in some text from a disc or cassette file. The Get File command will accept text from a file and insert it into the printed output.

This command does not process or alter the text or data from file in any way before outputting it. It is therefore possible to print a file which contains complex printer commands etc.

The inevitable penalty that has to be paid for this flexibility in driving the printer is that Wordwise-Plus will be unaware of the position of the paper at the end of the file that has been read in.

If you wish to process the file, if it is a WORDWISE PLUS file containing embedded commands for instance, then you should instead use the Print File (**P F**) command, described later in the manual.

HP Heading Position

Examples

f1 HP 3 f2
f1 hp5 f2
f1 HP A% f2

When a heading has been defined with the DH command you may wish to have the heading printed out at some position in the top space other than the default. You can alter the position of the heading within the Top Space by use of the HP command.

This command will only have an effect if:
firstly paging has been enabled with an EP command and secondly a heading has been defined with a DH command.

HP can be followed by a number between one and the value of TS – the number of lines in the Top Space. If the value is larger than TS the header will not be printed.

IN Indent

Examples

f1 IN0 f2
f1 in 15 f2
f1 in A% f2

This command is used to set an indent so that all lines start indented from the left margin. LM10 and IN10 together have the same effect as setting LM20. In both cases the first printed character will be preceded by 20 spaces on the left. However there are differences and the LM command should be used to control overall page size whilst the indent commmands should be used to control individual lines or paragraphs within the total layout.

To cancel an indent you can either use f1 IN0 f2 – literally set an indent of zero – or the special command Cancel Indent (CI) which has exactly the same effect.

A frequent requirement is to cause the first line of a paragraph to start to the left of the rest of the paragraph. There are several ways of achieving this but the following are suggested.

To produce a 'left hanging indent':

set the main indent first then
set a Temporary Indent of zero for the first line. Thus

```
f1 IN5 f1 TI0 f2 A new poster had suddenly
appeared all over London. It had no caption,
and represented simply the monstrous figure
of a Eurasian soldier, three or four metres
high...
```

produces

```
A new poster had suddenly appeared all
      over London. It had no caption, and
      represented simply the monstrous
      figure of a Eurasian soldier, three
      or four metres high ...
```

An alternative technique is to place the `IN5` command After the first word in the line, thus

```
A f1in5f2new poster had suddenly appeared
all over London. It had no caption, and
represented simply the monstrous figure of a
Eurasian soldier, three or four metres high
...
```

this produces the same effect since the Indent command will have arrived too late to affect the current line:

```
A new poster had suddenly appeared all
      over London. It had no caption, and
      represented simply the monstrous
      figure of a Eurasian soldier, three
      or four metres high ...
```

JO Justify On

Examples

f1 j o f2
f1 J O RETURN

Justified text has a straight right margin as well as a straight left margin. Wordwise-Plus places as many words as possible on each line but does not spread the words out so that the right margin is straight. If you want a straight right margin then place a J O command at the start of your text.

The first paragraph is shown as Wordwise-Plus would normally produce it. The second paragraph is shown justified – as it would be produced with the J O command.

```
Although it is excruciatingly rich,
horrifyingly sunny and more full of
wonderfully exciting people than a
pomegranate is full of pips, it can hardly be
insignificant that when a recent edition of
"Playbeing" magazine headlined an article
with the words 'When you are tired of Ursa
Minor Beta you are tired of life', the
suicide rate there quadrupled overnight.
Not that there are any nights on Ursa Minor
Beta.

It is a West zone planet which by an
inexplicable and somewhat suspicious freak
of topography consists almost entirely of
sub-tropical coastline. By an equally
suspicious freak of temporal relastatics,
it is nearly always Saturday afternoon just
before the beach bars close.
```

From – The Restaurant at the End of the Universe by Douglas Adams.

LL Line Length

Examples

f1 LL 66 f2
f1 ll L% f2

The LL command is used to set the maximum number of characters that can be printed across the page. If there is no left margin (LM0) then setting LL50 would cause lines of 50 characters and spaces to be printed. On the other hand LL50 and lm15 together would make each line consist of 15 blank spaces followed by text in the next 50 character positions, a total of 65.

Settings of LL70 and LM0 are assumed, unless set otherwise, so that printed output fits onto normal printer paper which is 8.5 inches wide and onto A4 paper which is 8.25 inches wide. Printers normally print 10 characters per inch so there is room for an absolute maximum of 85 characters on normal printer paper. Setting LL70 gives a maximum of 7 inches of print with a half inch margin on the left. You might think that this will result in a 1 inch margin on the right – and so it should! however printer paper is invariably inserted so that the printer starts about a quarter of an inch from the edge of the paper and the text will, in fact, end up centred on the page. Why do people set the paper up like this? Well when printing Basic listings it is essential to have something of a margin so the printer paper is usually inserted to allow for this!

The computer attempts to put as many words as possible into each line without splitting any word in half. However, if the word is longer than the line, words may be split in 'half' – for example

```
f1 LL24 f2 The East Fichley
Antidisestablishmentarianist co-operative
meets on Fridays.

The East Fichley
Antidisestablishmentaria
nist co-perative meets
on Fridays.
```

LM Left Margin

Examples

f1 LM5 f2
f1 lm 15 f2
f1 LMV% f2

The LM command sets the number of blank spaces that are printed at the start of each line. It should be used (in conjunction with Line Length) to set the overall layout of the page. If a single line is to be printed with a different indent from the left hand side of the page then the Temporary Indent (TI) command should be used. The default value for LM is 0 which means that, unless you tell the computer otherwise, it will start every line at the first possible printing position

LM can be followed by a number between 0 and 150, values outside this range behave as if LMØ had been entered.

Note that setting the left margin does not change the number of text characters printed on a line so increasing the LM setting will shift both the left and right hand edges of the document to the right. The whole of the printed text will move across the page without changing the format in any other way.

LNE Line Number End

Example

f1 LNE f2
f1 lne f2

This turns OFF line numbering.

See the next entry, **LNS**, to find out how and why to turn it on in the first place.

LNS Line Number Start

Examples

f1 LNS f2
f1 lns f2

When you are trying to arrange text on a page it is sometimes very useful to be able to preview the text in its formatted form using menu option 7 – and to have line numbers shown as well This is particularly useful when trying to set the Top Space or Bottom Space, or trying to fit a Footer in the right position in the Bottom Space: using the Footer Position command.

The commands Line Number Start (**LNS**) and Line Number End (**LNE**) enable you to turn line numbering on and off as you move through the document.

You may also find it useful to place markers around the section of the document that you are interested in so that you don't have the rest of the document previewed as well.

If line numbering is on when previewing a marked section then the line numbers will show exactly where on the printed page the previewed section belongs. The command will take acount of all previous embedded commmands.

Line numbers are inserted in front of the first character of each line. It may cause long lines to spill-over onto the following screen line.

Note: line numbering is on-screen only, they never appear on the printed copy.

LS Line Spacing

Examples

f1 l s 2 f2
f1 L S D % f2

This command is used to set the number of blank lines printed between each line of text. Normally there are no blank lines between lines of text. This is called 'Single Spacing'. When writing articles that are to be sub-edited it is normal to leave a blank line between each printed line. This can be set by inserting an `LS 2` command at the start of your text, for instance:

f1 E P RETURN
f1 L S 2 RETURN
f1 L N S RETURN

together would enable paging, set double-spacing, and turn on line numbering.

To return to the normal 'print on every line' setting you can use the `LS 1` command. Alternatively there is a command to set Single Spacing specifically, the `S S` command.

f1 L S 1 f2 or f1 S S f2 will set single spacing – no blank lines between each line of text.

f1 L S 2 f2 will set double spacing – one blank line between each line of text.

f1 L S 3 f2 will set triple spacing – two blank lines between each line of text.

and so on...

NJ No Justification

Examples

f1 n j f2
f1 N J f2

Newspapers always 'justify' their lines, that is they spread the words out so that they always fill the full column width. Wordwise-Plus will produce justified text if the Justification ON (J 0) command is used. If, having used J 0, you wish to return to non-justified printout then you should insert a No Justification (N J) command.

See the command J 0 for examples of justified and non-justified print.

OC Output Control code

Examples

f1 OC 15 f2
f1 OC 18 f2
f1 OC 27,45,1 f2
f1 OC 27,45,0 f2

There are occasions when you will need to send characters to the printer only – for example to select the condensed print mode offered by some printers. The **OC** command will enable you to send any sequence of characters that you wish.

For example, on the Epson RX-80, Condensed mode is enabled by sending a CHR$(15) to the printer. To do this place the command

f1 OC 15 f2

in your text.

Since so many of the special printer commands start with an ESC code (27) a special command is provided in Wordwise-Plus to send that code followed by other codes. See the command **ES** for more details.

OPS Output Print Sequence

Examples

f1 OPS 0 f2
f1 ops9 f2
f1 OPS A% f2

There are 10 possible defined sequences that can be set up and sent to the printer on command.

Four of these sequences (numbers OPS0 to OPS3) are initially set to produce codes for Epson printers, but all may be re-defined using the Redefine Print Sequence (RPS) command.

Sequences 0 to 3 can also be sent to the printer with other specific commands:

Initially the sequences are set up thus

Sequence	Equivalent command	ASCII	Default codes	
OPS0	Underline Start	US	ESC "-",1	27,45,1
OPS1	Underline End	UE	ESC "-",0	27,45,0
OPS2	Double-strike Start	DS	ESC "G"	27,71
OPS3	Double-strike End	DE	ESC "H"	27,72

OS Operating System call

Examples

f1 os"cat" f2
f1 OS"HELP" f2
f1 OS A$ f2

This command will pass the string to the operating system and at the same time will copy all the resultant screen output to the printer if using option 6. Thus

```
And the following Help messages are
available in my computer f1 OS"Help" f2
```

will include the help messages in the printed text.

Note however that Wordwise-Plus will not keep track of the number of lines output and all paging will probably be ruined This is unavoidable since these commands are passed to the operating system from where anything could occur. It is also possible, if not likely, that some such commands will destroy some or all of your text in memory. Make sure that you know what commands do before calling them, and ensure that you have saved your text before experimenting.

PA Pause

Examples

f1 p a f2
f1 P A f2

If you wish to pause at some stage during the printing of a document then you can do so with the PA command.

This is useful, for example, if you wish to change the daisy wheel on a printer in the middle of a document, or if you need to stop at a specific place to check something.

When you wish to continue printing you just press the SPACEBAR to continue.

This command also operates during preview and spool operations.

PC Pad Character definition

Examples

f1 PC "%" f2
f1 PC A$ f2
f1 PC "#" f2

When text is being justified, that is after the J O command has been met, words will be spread out along the line so as to give a straight right margin. Justification works by increasing the spaces between words – it will never place spaces between letters in a word. If you wish to keep a certain fixed distance between words then instead of spaces between the words you can place "pad characters". The default pad character is "‖" so

This‖is‖one‖word but these are separate

will ensure that "This is one word" is treated as a single word. When the text is printed Wordwise-Plus will substitute a single space for each pad character AFTER the line has been justified.

The normal, default, pad character is "‖" but you may change it using the PC command. Thus

PC"%" f2
%%%%1%%%%2%%%%3%%%%4

will always be printed as

1 2 3 4

Note that the syntax of the PC command has changed from that used in previous versions of Wordwise where it was PC% The new syntax PC"%" has been introduced to permit strings (e g. D$ in which case it uses the first character of the string) to be used to define the Pad character. Just place the character in quotes when using Wordwise-Plus.

PF Print File

Examples

f1 pf A$ f2
f1 PF"Insert" f2

The Print File command enables another file to be included in the output produced when the text is printed, previewed or spooled.

All embedded commands in the included file will be obeyed except for a request for a further included file.

This command can be used to join together a number of standard paragraphs into a form letter. For example:

f1 PF"address" f2

```
Dear Mr. Jones,

We thank you for your order for a technical
manual. Please accept our apologies for the
delay in despatch.
```

f1 PF"terms" f2

```
It would however be most helpful if we could
have a complete product before we hand over
the manuscript.
```

f1 PF"sincere" f2

PL Page Length

Examples

f1 PL66 f2
f1 pl 30 f2
f1 pl R%+10 f2

Normal computer printer paper is 11 inches long and printers usually print 6 lines per inch – so there are 66 lines per page. The PL command is used to set the overall page length. The default value is 66. In the unusual event that you are using A4 paper in your printer then set PL72. Use PL78 for foolscap.

In practice it looks better to leave some space at the top and bottom of a page. These "top space" and "bottom space" areas are each set to 6 lines by default so when you create a document on Wordwise it will only have 54 printed lines on each page, unless you alter the default settings.

As indicated above you can tell the computer to assume a different Page Length if using A4 or foolscap paper and you can also alter the Top Space and Bottom Space with the TS and BS commands.

Note that the PL command only takes effect if the computer has been told to Enable Paging. Unless you place an EP command at the start of your text the computer will not separate your document into pages and thus the PL command, and other page related commands, will have no effect.

PN Page Number

Examples

f1 PN 1 1 f2
f1 pn 1 0 0 f2
f1 pn P% + 3 f2

When paged output has been selected, by the Enable Paging (EP) command, then it is possible to print out the page number at the bottom of every page. In fact, by default, Wordwise-Plus will print

PAGE 1

in the centre of Bottom Space of the first page when paging has been selected. This command sets the page number to the specified value. Normally Wordwsie Plus assumes the first page is number 1 but, by using this command at the start of the text, it is possible to start from any page number.

If you have a very large document split into a number of files – a prudent way of working – then you may want to set the page number to be used for the first page of each file. This is done using the PN command.

During printing the page number is copied into the numeric variable P%.

PP Print Page number

Examples

f1 pp f2
f1 PP f2

The PP command can be used to print out the current page number at any point in a document. Thus

```
Here on page f1 pp f2 we are dealing with the
Print Page command
```

will produce

```
Here on page 56 we are dealing with the Print
Page command (obviously the page number will
be the current one).
```

If, when printing, you wish to split the document into pages then you must use the Enable Paging (EP) command at the start of your document. By default the computer will print the word "PAGE" and the current page number at the bottom of each page, centred. If you wish you may set up a different footer with the Define Footer (DF) command and this footer can include the page number at any position you wish. For example

f1 DF f1 CE f2 Section 1 — Page f1 pp f2 RETURN

PS Print String

Examples

f1 PS A$ f2
f1 ps STR$(A%+10) f2

This command will process a string and include it, as if it existed as part of the text, within the formatted output.

The PS command will evaluate a string expression, including string variables, literal strings, and string type functions. The evaluated string may contain multiple lines, separated with RETURN characters (CHR$13). The total string used may be up to 255 characters long.

No matter how many characters and lines there are in the included string, the formatting will be performed correctly. No characters will spill over the line length and the paging will be correct.

This command is ideally suited to mail merging applications where strings of varying length and complexity are to be included throughout the printed text. A program could for instance read values for A$, B$, and C$ from file, then issue the PRINT TEXT command. The text would be set up to contain the commands f1 PS A$ f2, f1 PS B$ f2 and f1 PS C$ f2 at the appropriate points. The result would be the standard text, including the contents of A$, B$ and C$ where required. Such a program would presumably go on to read the next set of information and print again, this time the new strings would be printed.

Since the current filename is held in F$ one can easily automatically include the working filename in headers or footers with the command shown below:

f1 DH f2 File f1 PS F$ f2 RETURN

RPS Redefine Print Sequence

Examples

```
f1 RPS 4,27,33,40 f2
f1 RPS9,54 f2
f1 rps A%,B%,C% f2
```

There are 10 possible defined sequences that can be set up and sent to the printer on command.

Four of these sequences (numbers `OPSØ` to `OPS3`) are initially set to produce codes for Epson printers, but all may be re-defined using the Redefine Print Sequence (`RPS`) command.

Initially the sequences are set up thus

Sequence	Equivalent command	ASCII	Default codes	
`OPSØ`	Underline Start	`US`	ESC "-",1	27,45,1
`OPS1`	Underline End	`UE`	ESC "-",0	27,45,0
`OPS2`	Double-strike Start	`DS`	ESC "G"	27,71
`OPS3`	Double-strike End	`DE`	ESC "H"	27,72

To redefine `OPS7` to produce an italic character set on the Epson RX-80 printer requires the following embedded commmand:

```
f1 RPS 7,27,52 f2
```

which is shown in the printer manual as ESC "4". The ASCII code for ESC is 27, and for "4" is 52.

All characters assigned to all of the print sequences are stored in a special area of memory. About 50 characters are reserved for this purpose. This means that no more than about 50 characters can be assigned to the printer sequences in total, which is unlikely to be a problem. If the limit is exceeded, for instance by

attempting to assign ten characters to each sequence, then formatting will halt with an error "No room!" at the sequence which would over-run the limit.

The area of memory containing the defined sequences is automatcally saved with the command

```
SAVE PARAMS "DAISY"RETURN
```

where 'daisy' is a filename of your choice. In this way you may set up all of the printer sequences for your printer in RPS definitions, preview the text to install them in memory, then save them. Next time you need them, simply use the command

```
:LOAD PARAMS "DAISY"RETURN
```

from the initial menu. They are then installed until WORDWISE PLUS is reset.

SEG Executes a program SEGment

Examples

f1 SEG 1 f2
f1 segA% f2

A program in a segment can be executed during formatting at the position in the text where the SEG command is placed. The program may print information of its own or change settings such as Line Length.

Any program output produced other than with the DOLINE command will not be taken into account by Wordwise Plus, upsetting page length etc. It is best always to use the DOLINE command in this situation.

SP SPace – insert blank lines

Examples

f1 SP 10 f2
f1 sp3 f2
f1 sp A% f2

This command inserts a number of blank lines in the text. For example inserting the code f1 SP2 f2

at this point will immediately cause two blank lines to be printed out followed by the remaining text.

SS Single Spacing

Examples

f1 S S f2
f1 s s f2

If you have used the command LS to increase the line spacing, then you can make the computer revert to single spacing by issuing the SS command. For example the text

```
f1 L S 2 RETURN
Line one
Line two
Line three f1 S S f2
Line four
Line five
Line six
```

will print as

```
Line one

Line two

Line three
Line four
Line five
Line six
```

For further information see the earlier description of the LS command.

TI Temporary Indent

Examples

f1 TI 0 f2
f1 ti 15 f2
f1 TI B% f2

The TI command has two effects: firstly it causes the text immediately following the command to be placed on the next line down: and secondly it indents that line, only. This is particularly useful for starting paragraphs, where the command TI 10 will produce all the normal formatting.

Thus the text:

```
Dear Jetta f1 TI 10 f2 Many thanks for the drafts
```

produces

```
Dear Jetta
          Many thanks for the drafts
```

TS Top Space

Examples

f1 TS 7 f2
f1 ts 0 f2
f1 ts e%-1 f2

If you have told the computer to split your document into pages – by placing an EP command at the start of your text, then you may wish to alter the number of blank lines at the top of each page. The Top Space, as it is known, is set to 6 lines by default.

The total number of lines on a page is set by PL and the top and bottom spaces are controlled by TS and BS. The defaults are PL66, TS6 and BS6 which means that each page contains 66-12, that is 54 printed lines of text. Remember that these commands only work after an EP command.

With the Top Space set to its default value of 6, there will be 6 blank lines automatically printed at the top of every page when paging is enabled. To remove these lines set the Top Space to zero line using the command

TS0 RETURN

TS may have any value in the range zero to 50. Values outside this range are taken as the default TS6.

UE Underline End

Examples

f1 UE f2
f1 ue f2

This terminates underlining that has been previously started with the Underline Start (**US**) command.

The codes generated are suitable for the Epson RX-80 and can be changed to suit other printers with the **RPS** command.

The codes generated are in fact

27,45,0

which is shown in the printer manual as

CHR$(27);"-";CHR$(0)

The Qume 9 and 11 printers require the sequence

27,74

to terminate underscore – though a new line automatically cancels underscore mode on the Qume.

US Underline Start

Examples

f1 US f2
f1 us f2

Certain printers can be told to underline text as they print it.

To start the printer underlining a sequence of characters has to be sent to the printer. The command US has been set up to send the correct sequence for Epson RX-80 and related printers. The sequence generated is

27,45,1

Sequences suitable for other printers can be set up with the Redefine Printer Sequence (RPS) command. For example the Qume Sprint 9 and 11 require the sequence

27,73

The sequence generated by the US command can be changed to suit the Qume printer by including the command

f1 RPS 0,27,73 RETURN

at the start of your text.

The Qume printer ends underscore at the end of each line and so it is prudent to sprinkle Underscore On commands heavily in text to be printed on the Qume in case a new line is started whilst underlining is in progress.

When previewing text, underlined characters are shown if the preview is in 80 column mode.

6. Commands

Introduction

The most important new feature in Wordwise-Plus is the ability to run programs which can manipulate your text. To allow you to do this a whole new language has been developed with over 70 different commands and functions. (a function is a statement that does something and then gives you a result back, whereas a command just does something.)

This chapter lists all the different commands in alphabetical order with a descripton of each.

Functions are described in chapter 7.
Most of the commands in Wordwise-Plus can be used in two ways, either as an instruction in a program or as an immediate command from a menu. To execute a command from either of the menus you must first type in a colon when the computer asks you to enter your choice, and then the rest of the command. For example, if you have the main menu in front of you and you type

```
:PRINT 3+2
```

the answer `5` will be printed on the screen. If you use this command in a program then you must not put the colon in front.

Note: like BBC BASIC many of the commands can be abbreviated. For instance `PRINT` can be shortened to `P` followed by a full stop.

*COMMANDS

Examples

```
*CAT
*basic
```

Introduction

In Wordwise-Plus *commands can be used either from a menu or from a program in a segment.

When you are in a menu and confronted by the message

```
Please enter choice
```

you can simply type a 'star' followed by the rest of your *command to execute it. *commands are the only commands that you can use from a menu without putting a colon in front.

As with Basic *commands can be freely entered in programs.

BPUT#

Examples

```
BPUT# N%,12

BPUT#C%,A%
```

Introduction

This command sends one byte to an open file on your disc or tape. A byte is a number which has a value between 0 and 255 inclusive. If you try to use a number greater than this then 256 will be repeatedly subtracted from it until the number is in the right range and this is then sent to your file. The file being written to must first have been opened using the OPENOUT command.

The BPUT# command word must be followed by two parameters. Firstly a variable which holds the channel number (as obtained by the OPENOUT function). This tells the computer where to send the information, and secondly, a number or variable which specifies the byte to send. For example, the sequence

```
X%=OPENOUT"Andy"
BPUT#X%,66
```

will send the byte 66 to the file called "Andy".

Definition

A statement which puts the least significant byte of its second argument to a file. The file is specified by a channel that must be given as the first argument.

CALL

Examples

```
CALL &FFEE

CALL G%
```

Introduction

This statement makes the computer execute a machine code routine. This could be one that the user has placed in memory or an operating system routine like Oswrch. Whichever is the case, **CALL** must be used with extreme care as calling an address that is illegal in Wordwise-Plus may destroy your text.

Please note that it is important that users shuld not use any zero page locations, i.e. from memory locations &00 to &FF, for their own routines as these are all used by Wordwise-Plus. It is suggested that users should use the define character buffer or the RS423 buffer for their own routines, but even these should be used with caution.

Definition

A statement that calls a piece of machine code. On entry parameters get passed to the A, X and Y registers from the least significant bytes of variables A%, X% and Y%. You can also set the carry flag on entry through setting the least significant bit of C%.

CLOSE#

Examples

```
CLOSE#X%
CLOSE#Ø
```

Introduction

This tells the computer that you have completely finished reading from or writing to a file. As with the other file handling commands, it must be followed by a number called a channel number which specifies which file you are dealing with.

Definition

A statement used to `CLOSE` a specific disc or cassette file `CLOSE#Ø` will close all open files. In some filing systems if a `Can't Extend` error occurs then a file may be left open. In this case you must close it using `CLOSE#Ø` from the menu before it can be deleted and re-saved.

CLS

Examples

```
CLS
IF X%=5 THEN CLS
```

Introduction

`CLS` can be used from a segment to clear the screen. When a program uses `CLS` and needs to restore the original text screen afterwards, the `DISPLAY` command should be used.

Definition

Clears the whole screen moving the cursor to its top left hand corner.

CURSOR

Examples

```
CURSOR RIGHT 3
CURSOR LEFT A%
cursor up
CURSOR DOWN 12
cursor top
cursor BOTTOM
CURSOR AT 39
```

Introduction

The command **CURSOR** is a way of moving the cursor around the text or segment being edited. It must always be followed by one of the following keywords:

RIGHT
LEFT
UP
DOWN
BOTTOM
TOP
or**AT**

CURSOR RIGHT and **CURSOR LEFT**
These commands move the cursor to the right or left on the screen as the right and left arrow keys would. If the cursor is at the start of a line and you execute the command **CURSOR LEFT** then, as with the arrow keys, the cursor will move to the end of the previous line. The opposite will occur with **CURSOR RIGHT**. Both of these commands can be followed by a number or expression which indicates how many spaces the cursor should move. For example,

```
CURSOR RIGHT 4
```

would move the cursor right 4 characters. If no number or expression is given then the cursor will only move one place.

CURSOR UP and **CURSOR DOWN**
These move the cursor up and down the text as the up and down arrow keys would. The commands can be followed by a number or expression which tells the cursor how many lines to move. If this number is not there then the cursor will be moved just one line.

CURSOR TOP and **CURSOR BOTTOM**
These commands move the cursor to the top or bottom of your text as pressing SHIFT ↑ or SHIFT ↓ would do.

CURSOR AT
This moves the cursor along its current line to a specified character position. For instance if you use the command

```
CURSOR AT 0
```

then the cursor will be moved to the start of the current line and

```
CURSOR AT 39
```

will move it to the end. When a character is read from the text or when the text is displayed, after using a **CURSOR AT** command, the 'actual' cursor position is adjusted. If the cursor had been moved beyond the last character on the line it is positioned immediately after the last character, otherwise it remains unchanged.

Definition

CURSOR RIGHT, LEFT, TOP and **BOTTOM** move the cursor in the appropriate direction. The number of spaces moved is given by an optional numeric argument whose default setting is 1. **CURSOR AT** moves the cursor to the character position on the current line specified by the numeric argument. This argument can only take values 0 to 39.

DEFAULTS

Example

```
DEFAULTS
```

Introduction

This resets the embedded commands, such as `LL70`, `LM0`, and tab stops back to their default values.

Definition

A command that resets the embedded command settings to their default values.

DELETE

Examples

```
delete left 4
DELETE AT 33
DELETE WORD A%
DELETE MARKED
DELETE MARKERS
DELETE text
```

Introduction

The command DELETE must be followed by one of 6 other keywords, LEFT, AT, WORD, MARKED, MARKERS and TEXT.

DELETE LEFT
This command deletes characters in your text or segment as though you were pressing the DELETE key. This means that, unlike DELETE AT, it is the characters to the left of the cursor that get rubbed out. DELETE LEFT can be followed by a number or, to be precise, a numeric expression which tells it how many characters to delete. If no number is given then only one character will be rubbed out as though the delete key were pressed just once. If it is followed by the number 3 for example, then the effect will be the same as pressing the DELETE key 3 times.

DELETE AT
This has the same effect as pressing CTRL A and deletes AT and to the right of the cursor. Like DELETE LEFT it can be followed by an optional number or variable which tells it how many characters to delete.

DELETE WORD
This has the same effect as pressing CTRL D and deletes the words at and to the right of the cursor. Like the two above it can be followed by an optional argument which specifies the number of words to be deleted.

DELETE MARKED
This deletes the text between the markers as f7 does.

DELETE MARKERS
This deletes the markers from the text as though CTRL R is pressed. If no markers are present then nothing will happen.

DELETE TEXT
This command should be used with care. It will delete ALL text from the currently selected segment or text area. If ALL segments, i.e. everything is to be cleared from memory for a total reset, then the NEW command should be used instead.

Definition

DELETE LEFT and AT delete characters to the left or at the cursor position. DELETE WORD deletes words at the cursor position. All three of these take an optional numeric expression as an argument which specifies the number of items to be deleted. The default setting is 1.

DISPLAY

Example

```
DISPLAY
```

Introduction

Whenever you change your text (or segment) using a program, it does so without immediately showing the changes on the screen The changes occur but it is not until you press a key that they are shown on the screen. For instance, if you have a simple program in a segment like

```
TYPE "hello"
```

and run it from Edit Mode by pressing SHIFT and the appropriate red function key, nothing will happen to the text on your screen. As soon as you press another key though, the inserted word "hello" will be shown.

You can get over this by using the **DISPLAY** command which shows the text on the screen in its updated condition. For instance if we change that program to

```
TYPE "hello"
DISPLAY
```

then the word "hello" will be automatically inserted and shown on the screen when the program is run.

Definition

A command that displays the currently selected piece of text in its current state.

DOLINE

Examples

```
DOLINE "||Gce||WThis is centred"
DOLINE "||Gfi||WThis is fully indented"
```

Introduction

This command takes a line of text and prints it on the screen. This line of text though can include embedded commands which will affect its format. To include an embedded command. in such a line you must precede it with ||G and follow it with ||W. These stand for the *green* and *white* characters that you get in Edit Mode from pressing function keys f1 and f2. This command is especially useful for sending just one line to the printer. The printer can be switched on and off using VDU2 and VDU3.

Definition

A command that prints on the screen the string expression that it takes as an argument, taking account of the current format settings and interpreting any embedded commands that this contains.

DOTHIS

Examples

```
DOTHIS
 PRINT "hello"
TIMES 12

DOTHIS
 TYPE "A"
TIMES 3*A%
```

Introduction

This construct makes the computer repeat a set of instructions a number of times specified in the expression following the keyword.

It is comparable with the FOR .. NEXT loop structure of the BASIC programming language.

Definition

A control structure which makes the computer execute the enclosed instructions a given number of times. This is given by the numeric expression that `TIMES` takes as an argument.

END

Example

```
END
```

Introduction

This tells the computer that it has reached the end of the program. **END** is optional and may be used as many times as you wish.

Definition

An optional end of program command which may be put anywhere and as often as is required.

ENDPROC

Example

```
ENDPROC
```

Introduction
This statement marks the end of a procedure. See the keyword `PROC` for details of procedures.

Definition

A program object that indicates the end of a procedure.

FIND

Examples

```
FIND A$

IF X%=4 THEN FIND "end"
FIND "ABC##FG"
FIND MARKERS
```

Introduction

FIND is a way of searching your text (or segment) for a word or character string. It searches down the text from the current position of the cursor until it finds the given word and then moves the cursor to the first letter of that word. If the word is not found then the cursor will be left at the end of the text. It is possible therefore to test for the presence of a word with the commands

```
FIND "hello"
IF EOT THEN PRINT"Not Found."
```

It is possible to search for special characters such as *green* and *white* using the codes listed below:
||G – *green*
||W – *white*
||R – RETURN
||T – TAB

There is also a *'wildcard'* character, the hash sign (#). This may be used to represent any single character. A search for a hash on its own would be pointless.. it would match with every character. But a search for "s#t" would match with "sat, "sit", "set", and so on.

The SEARCH command may use all the above conventions for its search string.

Because the hash sign is used as a wildcard, neither the FIND nor REPLACE commands may be used to find the actual hash sign itself. This can easily be achieved by using the FKEY4 command instead

```
FKEY4,"#"
```

which will move the cursor to the next hash sign.

FIND MARKERS
If FIND is followed by the keyword MARKERS then the cursor will be positioned at the next marker found in the text, or at the end if no marker is found.

Definition

A command that searches the currently selected text for the first occurrence of the specified string expression. The search starts at the current cursor position. If successful the cursor is moved to the first character of the string otherwise the cursor is moved to the end of the text.

FKEY

Examples

```
FKEY 1
FKEY Ø
FKEY 4,"||G"
FKEY 5,A$
fkey 6,"P"
```

Introduction

In Wordwise-Plus the ten red function keys **f0** to **f9** all have defined functions. **f3** for example inserts a marker in your text and **f1** will insert a *green* code. You can make a program act as though one of the keys has been pressed by using the FKEY command. For exmaple

```
FKEY 3
```

would insert a marker in your text at the current position of the cursor.

Normally if you press function keys 4, 5, or 6 you are asked to press a key which specifies which character the computer should search for. To simulate this with FKEY you have to type in the character after FKEY and the key number. For instance,

```
FKEY 4,"F"
```

would move the cursor to the first occurrence of F. You can also put in the codes for *green*, *white*, **RETURN** and **TAB**, represented by the codes ||G, ||W, ||R and ||T respectively. Please note that you cannot search for markers using this technique. This is best done by using the command FIND MARKERS.

When using function key 7 to delete the text between markers, if you are about to delete more than 250 characters, then the computer will ask you if you are sure. FKEY 7 does not do this, so make sure that your markers are in the right place first!

Definition

A command that replaces the function of the unshifted function keys. The number of the key is given by the expression which must follow FKEY. FKEY 4, FKEY 5 and FKEY6 also take a string expression as an extra argument, separated from the key number by a comma, the first character of which provides the search character.

GOTO

Examples

```
GOTO label
GOTO name
```

Introduction

This statement makes the computer jump to a specific part of your program and so changes the order in which the program instructions are executed. Programs in Wordwise-Plus however do not have line numbers as in Basic, which means that we must have some other way of telling where the `GOTO` will go to. Wordwise-Plus uses labels. For example

```
GOTO finish
PRINT"This is not executed"
.finish
```

A label is a series of characters which are preceded by a full stop and must have a line all to themselves.

A `GOTO` can only be executed in a segment and can only jump to a label within that segment.

Definition

A statement used to transfer control to a specified label unconditionally.

IF...THEN

Examples

```
IF A%>0 THEN TYPE " and "
IF D$=CHR$13 THEN GOTO finish
```

Introduction

This construct sets up a test condition which you can use to control what the computer does next. It is similar to the Basic command but with two important differences. The keyword **THEN** is not optional and must always be entered after the condition expression.

Definition

A command that evaluates the expression following the **IF** and then executes the following statement if the result is **TRUE**. Unlike Basic **THEN** is compulsory and there is no **ELSE** clause.

LET

Examples

```
LET A%=10
```

```
LET A$="string"
```

Introduction

This can be used for assigning the value of an expression to a variable. For instance

```
LET A%=6
```

will store the value 6 in the variable named `A%`. As in BASIC, its use is entirely optional and so the statement

```
A%=6
```

will have exactly the same effect.

Definition

An optional keyword that can be entered before assignments.

LOAD

Examples

```
LOAD TEXT "file1"
LOAD TTC "B."+D$
LOAD PARAMS "EPSON"
```

Introduction

Like BASIC, **LOAD** loads a file into the computer from disc or tape. In Wordwise-Plus however there are two things that you must also type in after it. The first is one of the following extra keywords

```
TEXT
TTC
PARAMS
```

The second is the file-name corresponding to the file you wish to load. For instance

```
LOAD TTC "INSERT"
```

LOAD TEXT
This is like option 2 on both of the menus and just loads a normal Wordwise file so that it can be edited as your main text or as a segment. Use the **SELECT** command to choose which. Unlike the menu option this does not give a warning if you are about to overwrite your current text.

LOAD TTC
This loads the Text To Cursor just like option 4 on the menus and can be used to load text into either your main text or one of the segments.

LOAD PARAMS
This loads a file that has been saved by **SAVE PARAMS**. This provides an easy way of setting your printer parameters. See **Save Params** for further details.

Definition
These commands load a file from your current filing system. The file is specified by the string expression which must be given.

NEW

Example

```
NEW
```

Introduction

This command clears ALL text and ALL segments at the same time. It can be used effectively to restart Wordwise Plus completely. Users of the old version of Wordwise will have been used to answering 'no' when asked if they want to keep the text after pressing **BREAK**. The `NEW` command is the Wordwise Plus equivalent.

If this command is used within a program it will clear the program itself from memory as well as everthing else. It is only really useful as a command issued from the menu.

Definition
A command to restart Wordwise Plus from scratch.

WARNING
This command clears EVERYTHING from memory. It should be used with *extreme* care!.

OSCLI

Examples

```
OSCLI"HELP"
OSCLI A$
```

Introduction

`OSCLI` is another way of executing *commands. For instance, the command

```
OSCLI "TAPE"
```

```
will do exactly the same as
```

```
*TAPE
```

The difference between the two is that `OSCLI` is followed by a normal string expression which means that it can include variables as well. For instance

```
A$=":1"
OSCLI "CAT "+A$
```

would catalogue drive 1 (Acorn DFS file conventions).

Definition

A command that sends the following string expression to the operating system command line interpreter.

PREVIEW

Examples

```
PREVIEW TEXT
PREVIEW PAGE 4
PREVIEW MARKED
PREVIEW FILE G$
```

Introduction

PREVIEW displays on the screen the document, or part of the document, as it will look when it is printed. It must be followed by any one of four further keywords as detailed below.

PREVIEW TEXT
This is just like the menu option 7 and shows the whole of your text or segment on the screen in its formatted state.

PREVIEW MARKED
This shows only the text between the markers. An error will occur if there are less thsn two markers set.

PREVIEW PAGE
This shows just the page whose number must be entered after the keyword PAGE. The number can be any numeric expression and so can be a simple number, a variable or a mixture of both.

PREVIEW FILE
This shows how a file on your tape or disc would look if it were printed. The file should be a normal Wordwise-Plus file including embedded commands. For example

```
PREVIEW FILE "part1"
```
The PREVIEW command does not select an 80-column screen mode. It is still adequate for just checking things like page breaks or counting pages in a multi-file document etc. Users of machines fitted with a 6502 2nd processor or screen RAM board may select an 80-column mode by pressing the key sequence CTRL V 0 and then entering the PREVIEW command. This may also be achieved in a program by using the commands

```
VDU22,0
PREVIEW FILE "PART1"
VDU22,7
DISPLAY
```

Remember the DEAULTS command if you need it.

WARNING: If the above is used on a standard BBC machine (one without 2nd Processor or screen RAM board) the text will be corrupted or destroyed completely.

Note that the PREVIEW command does not reset formatting parameters (i.e. embedded command settings) to their default values. If they need to be reset then simply issue the DEFAULTS command first. The non-resetting is deliberate, so that you may preview one file immediately after another as a continuous document.

Definition

A command that shows a piece of text on the screen as it would look when printed.

PRINT

Examples

```
PRINT A$
PRINT 3+4;
PRINT TEXT
PRINT MARKED
PRINT PAGE 3
PRINT FILE E$
```

Introduction

Just as in BASIC, the **PRINT** command can be used to put a number or a string of characters on the screen. You can also follow this command in its entirety by a semi-colon which tells the computer not to send a carriage return after printing. For instance

```
PRINT "Good ";
PRINT "Morning"
```

would print "**Good Morning**" all on one line.

You must be careful not to confuse this command with **TYPE** as **PRINT** does not insert text into your document or segment.

The **PRINT** can also be used with a further keyword following. These act in similar ways to menu option 6 for printing text. The secondary keywords are **TEXT**, **MARKED**, **PAGE** and **FILE**. Their use in conjunction with **PRINT** is the same as their use in conjunction with **PREVIEW**, except of course that output is to the printer as well as to the screen.

None of the following commands will reset the formatting parameters (i.e. embedded command settings) to their default values as occurs when using the equivalent menu option. Therefore, if you want the page number to be set at 1, and all the other default values reset, you must issue the **DEFAULTS** command before using any of the commands below.

PRINT TEXT
This sends the whole of your text or segment to the printer, like menu option 6, but without resetting formatting defaults.

PRINT MARKED
This sends the text between the two markers in your document or segment to the printer. Menu option 6 will ask if you want to print the marked section only, if markers are currently set. Formatting defaults are not reset.

PRINT PAGE
This just sends the page specified to the printer. This must be followed by a number or numeric expression which tells the computer which page to be printed. If the specified page does not exist in the current text then nothing is printed. Because formatting defaults are not reset by this command, the page number will not automatically be reset to 1, so the numbering will continue from whatever page was last printed. This may prevent the specified page from being found, since its number would have been increased. It is often wise to use the DEFAULTS command before the PRINT PAGE command.

PRINT FILE
This takes a normal Wordwise-Plus file from your disc or tape and sends it to the printer. Any embedded commands that are in the file will be interpreted and so the text will get formatted as it passes through the computer. This command must always be followed by a string to tell it which file should be printed. Formatting defaults are not reset.

Definition
This is a multi-purpose command. Used with an additional keyword it can print all of the current text, a marked section, a specific page, or all of the text in a specified file. Alternatively, it is used as the primary command in programs for displaying information on the screen.

PROC

Examples

```
PROC enter
PROCfred
```

Introduction

In Wordwise-Plus the command **PROC** makes the computer run a procedure. These are rather different from procedures in BBC BASIC and are actually more like subroutines.
In Wordwise-Plus a procedure looks like this:

```
.hello
PRINT "This is a procedure"
A%=5
A%=A%*A%
PRINT A%
ENDPROC
```

It starts with a label (**.hello**), which is a word or a series of characters with a full stop in front and ends just as in BBC BASIC, with **ENDPROC**.

The procedure can then be called by the command

```
PROC hello
```

and all the instructions between **.hello** and **ENDPROC** will be executed. Once the procedure has finished, the computer will then return to the statement immediately below the one that called it.
A program including a procedure would look like this

```
PRINT "The answer to the ";
PROC answer
PRINT 42
END
```

```
REM The procedure starts on


REM the next line
.answer
PRINT "universe is ";
ENDPROC
```

This will print

```
The answer to the universe is 42
```

A procedure in Wordwise-Plus can only be called from a program within the same segment. All variables are treated as global, no local variables can be defined, and no parameters may be passed other than globally.

Definition
A command that calls a procedure from a program present in the same segment.

PTR#

Examples

```
PTR#X%=PTR#X%+10
PTR#Y%=0
```

Introduction

This statement selects which character is to be read or written next in a file on your disc or network but unfortunately not your tape. The file is specified by following `PTR#` with a variable that contains the channel number. This is obtained from either the function `OPENIN` or `OPENOUT` depending on whether you are reading from or writing to the file. `PTR#` can appear on both sides of an equal sign and so is both a statement and a function.

Definition

A statement and function which allows the programmer to move a pointer in a serial file thus enabling random access.

RECOUNT

Example

```
RECOUNT
```

Introduction

When you change the value of the variable W% or alter the number of words in the text using a program, then the word count that is in the top left hand corner of the screen in Edit Mode may not be accurate. The **RECOUNT** command will re-count the words in the current text and store it in W%.

Definition

A command that re-counts the total number of words in your text or segment.

REM

Examples

```
REM written by Joe Bloggs November 84

REM this is ignored
```

Introduction

This is used to insert remarks or comments in your programs When the computer is running a program and comes across a **REM** statement, it completely ignores the rest of that line letting you put anything you like on it. **REM**s are extremely important when writing long programs as you can leave notes explaining how the program works for future reference.

Definition

This statement allows comments to be inserted in a program.

REPEAT

Example

```
A%=0
REPEAT
 A%=A%+1
UNTIL A%=5
```

Introduction

The REPEAT..UNTIL loop makes the computer repeat a set of instructions a number of times until some condition is met. They can be nested (one inside another) up to a depth of 7 and if you try more then you will get the error 'Too many Repeats'.

Definition

A statement which is the start of a REPEAT...UNTIL loop.

REPLACE

Examples

```
REPLACE "news" , "information"
REPLACE A$,B$
```

Introduction

REPLACE searches for and then exchanges one string for another in your text or segment. It starts searching from the position of the cursor until it finds the first occurrence of the 'search string' which is then exchanged for the 'replacement string'. For example,

```
REPLACE " as " , " because "
```

will look for the word "as" and if it is found, replace it with "because".

Several special characters may be used in the search and replace strings, these are listed with the details of the FIND command.

Definition

A command that searches in your text or segment for the first occurrence of its first argument and replaces it with the second Both arguments must be string expressions and separated by a comma.

SAVE

Examples

```
SAVE TEXT "index"
SAVE MARKED "A."+B$
SAVE PARAMS "EPSON"
```

Introduction

SAVE as you might expect saves whatever you specify onto your disc or cassette. It must always be followed by two things. Firstly, it must be followed by one of three keywords: TEXT, MARKED or PARAMS.

Secondly it must be followed by a string which tells the computer what the file-name will be.

SAVE TEXT
This saves your whole text or segment, whichever is selected, and so is just like option 1 on either of the menus. No warnings are issued if a file of the same name is about to be overwritten.

SAVE MARKED
This saves just the text between the markers like menu option 3. No overwrite warnings are issued.

SAVE PARAMS
This saves all current embedded command settings including line length, page length and all the special code sequences sent to the printer. Of course there is also a sister command called LOAD PARAMS that loads them back again, giving you a very useful pair of instructions. With these you need only set the codes for your printer once and then subsequently you can load them in from your tape or disc to save typing them in every time you use Wordwise-Plus.

DEFINITION

These commands save whatever is specified to the current filing system. The file-name is given by the string expression that must follow the command.

SELECT

Examples

```
SELECT TEXT
SELECT SEGMENT 4
```

Introduction

Most of the editing statements and functions in Wordwise-Plus can be used on either your main text or on any of the ten segments. This means that when you run a program from a segment you must be very careful that it is editing the piece of text you want it to. This is done using the SELECT command.

To select your main text for editing, use the command

```
SELECT TEXT
```

To select a segment, segment 0 for instance, use

```
SELECT SEGMENT Ø
```

Note that when you select a segment you must specify which one you want by entering its number. This can be done by typing a number, a variable or, to be more precise, a numeric expression. If you do not specify a number then the last segment used will be selected.

Take the following example

```
SELECT TEXT
A$=GCT$
SELECT SEGMENT 3
CURSOR TOP
TYPE A$
```

This will read a character from the text and will then enter it at the start of segment 3.

The commands SELECT TEXT and SELECT SEGMENT can both be used from either of the menus, though using the menu options provided will be easier. SELECT TEXT will transfer you to the main menu while SELECT SEGMENT will transfer you to the segment menu with the correct segment chosen.

Definition

A statement that selects the main text or one of the ten segments for subsequent editing.

SPOOL

Examples

```
SPOOL TEXT A$
SPOOL PAGE "fred",3
SPOOL MARKED "S."+"part"
SPOOL FILE "to","from"
```

Introduction

These commands correspond to option 8 on the menus and save your text on disc or tape in its formatted state just as it is previewed or sent to the printer. This is different from the normal **SAVE** commands which save the text in its unformatted state, i.e. with all the embedded commands still in. The original **SAVE**d version cannot be recreated from its **SPOOL**ed version. Note also that the file produced may be much longer than the **SAVE**d version, since all justification, margins, and centring cause insertion of spaces, which can add up to a considerable amount.

SPOOL must be followed by one of the four secondary keywords **TEXT**, **PAGE**, **MARKED** or **FILE**. Each of these is described below.

SPOOL TEXT
This saves all of your text or segment in its formatted state to the file specified on your disc or tape. It may be necessary, when pagination is involved, to issue the **DEFAULTS** command first.

SPOOL PAGE
This saves just a single page of your text or segment and must be followed by two things. Firstly a string containing the destination file-name and secondly a number or variable which tells it which page to spool. You may need to issue the **DEFAULTS** command before using this command, in order to get the page numbering correct.

SPOOL MARKED
This saves the marked text in its formatted state to the file specified. You may need to issue the DEFAULTS command in order to get the paging correct.

SPOOL FILE
This command does not alter or use the text currently in memory. It works only in a small 'work space' area. It will read a normal Wordwise-Plus file, with embedded commands, from disc and saves it in its formatted state in a new filename. The process involves reading a small part of the file, formatting a line and saving it, and so on. SPOOL FILE must be followed by two filenames, the first is the name of the new file to be created. This will hold the formatted version of the file specified by the second filename. The two filenames are separated by a comma, as in the example. You may need to issue the DEFAULTS command in order to achieve the correct paging.

Definition

These commands all save text to a file in its formatted state.

SWAP

Examples

```
SWAP 3
SWAP A%+2
```

Introduction

When you are in Edit Mode you can use the key combination CTRL S to change the case of the character at the cursor position To do this from a program or from a menu you must use the function SWAP. If SWAP is typed with nothing following it then it will act as though CTRL S has been pressed just once, i.e. it will change the character at the cursor from upper to lower case or vice versa and move the cursor on one place. SWAP can be followed by a number or a variable, or an expression containing both, which makes the computer act as though CTRL S has been pressed a number of times. For example,

```
A%=4
SWAP A%-1
```

will swap the case of the character at the cursor and also the two to its right.

This command has no effect on non-alphabetic characters.

Definition

A command which takes an optional numeric expression as argument and swaps the case of alphabetic characters from the cursor onwards. If no expression is given then only one character is swapped.

TIMES

Examples

```
DOTHIS
 PRINT "hello"
TIMES 6

DOTHIS
 TYPE "||R"
TIMES I%+1
```

Introduction

This is part of the DOTHIS...TIMES construct. See the keyword DOTHIS for more details.

Definition

A program object that signifies the end of a DOTHIS...TIMES loop. It must be followed by a numeric expression which determines how many times the loop should be executed. If TIMES is used without a DOTHIS statement then the error 'No REPEAT' will be generated.

TYPE

Examples

```
TYPE "the rain clouds loomed overhead"

TYPE A$
TYPE SEGMENT 8
```

Introduction

TYPE is used to insert text into your document (or segment) at the position of the cursor, just as if it were typed at the keyboard. If you follow the command by a string or a string variable like A$ then that string will be put in the text. If you follow the command by the keyword SEGMENT and a valid segment number then the whole of that segment will be inserted at the cursor position. For example,

```
TYPE SEGMENT 5
```

will copy all of segment 5 into your main text or one of the other segments (except the same one, 5).

The words that you insert using TYPE will not affect the word count in the top left corner of the screen in Edit Mode. The RECOUNT command will update the count, should you find it necessary.

Definition
A command that inserts text into your main document or segment at the position of the cursor. If the command is followed by a string expression then that string is inserted. If it is followed by SEGMENT and a number then that segment is inserted.

UNTIL

Examples

```
A%=1
REPEAT
  PRINT "hello"
  A%=A%+1
UNTIL A%>5

REPEAT
  CURSOR DOWN
  DISPLAY
UNTIL EOT
```

Introduction

This is part of the **REPEAT...UNTIL** construct and is used to make the computer repeat a number of instructions until a logical condition is 'TRUE'. Each time round the loop, the condition following **UNTIL** is tested. The loop is repeated until the condition is found to be **TRUE**.

Definition

A program object signifying the end of a **REPEAT...UNTIL** loop. It must always be followed by a logical condition.

VDU

Examples

```
VDU 2
VDU31,12,Y%
```

Introduction

Like BASIC, the command **VDU** can be used to print characters on the screen by sending their ASCII numbers to the VDU drivers. In this respect it is the same as the sequence

```
PRINT CHR$(X%);
```

Like BASIC, the **VDU** command is most useful for sending control characters to the VDU drivers. Care must be taken in using some numbers, for example changing mode using **VDU22** on an ordinary BBC micro may lose ALL of your text.

Listed below are some useful examples:

VDU 7 – beep
VDU 31,X%,Y% – moves the cursor to X%,Y%
VDU 23,1,0;0;0;0; – turns the cursor OFF
VDU 23,1,1;0;0;0; – turns the cursor ON
VDU2 – switches printer ON
VDU3 – switches printer OFF

Definition

A statement that takes a list of numeric arguments and sends them to the operating system output character routine (OSWRCH) just as in BASIC. If a number is followed by a comma then only a single byte is sent but if it is followed by a semi-colon then a two byte pair is sent.

7. Functions

Introduction
Wordwise-Plus is not just an ordinary word-processor, it is also a programming language in its own right and like any other language it has functions. A function is a type of command which, when executed, returns a result that can then be put in a variable, printed on the screen, etc.

In Wordwise-Plus there are two ways of using functions. You can either use them in a program in one of the segments, or you can make them execute immediately from one of the menus by preceding the statement by a colon. For instance, if you have the main menu or indeed the segment menu on the screen then you could type

```
:S%=LEN"string"
```

which would set `S%` equal to 6. In this case `LEN` is the function returning the value.

This chapter lists all of the functions available in Wordwise-Plus with a description of each.

ASC

Examples

```
Z%=ASC("D")
PRINT ASC(z$)+13
```

Introduction

With every character on the computer there is an associated number called its ASCII value. (ASCII stands for American Standard Code for Information Interchange) Sometimes it easier or more convenient to use this number to describe a character and sometimes it is easier to use the character itself. The function `ASC` gives you a way of changing between these types as it generates the ASCII values from their characters. (The opposite function is performed by `CHR$`, which converts the numbers into their character form.) For example, if you use

```
PRINT ASC("B")
```

the number 66 will be printed which is the character `B`'s ASCII value. `ASC` can work on any string or string expression but will only give the ASCII value of the first character in that string. For instance

```
A$="GOOD"
PRINT ASC A$
```

will return only 71, the ASCII value of G.

Definition

`ASC` is a function which returns the ASCII value of the first character of its argument string and, like BASIC, returns the value `TRUE` if that string is empty (null).

BGET#

Examples

```
A%=BGET#F%

PRINT BGET#A%
```

Introduction

Like BASIC, **BGET#** reads the next character from a file and gives that character's ASCII value. The computer knows which file to read by the variable which must be typed in after the command itself. This contains the channel number for the file and is obtained from the function **OPENIN**, described elesewhere.

It is important though that **BGET#** is not confused with another Wordwise-Plus function, **GCF$#** which is short for Get Character from File. Although this also reads a character from a file, it gives the character itself rather than its ASCII value. For instance, if the next byte to be read in from the file is the character "A" then **GCF#** would return the character "A" which could then be stored in a string variable such as D$. **BGET#**, on the other hand, would give the character's ASCII value, 65, which could then be put in an ordinary numeric variable like D%.

Definition

BGET# takes a variable containing a channel number as argument and gets the next byte from the specified file returning its ASCII value.

CHR$

Examples

```
PRINT CHR$(66)
LET B$ = CHR$(100)
```

Introduction

The purpose of **CHR$** is to generate a character from its ASCII number, just as in BASIC. For example

```
PRINT CHR$(66)
```

will print the character **B**, which has an ASCII value of 66, and

```
A$ = CHR$(72)+"ELLO"
```

will store the string "HELLO" in the string variable A$.

The function **ASC** can be used to change the character back to its ASCII value.

Definition

A string function whose value is a single character string containing the ASCII character specified by the least significant byte of the numeric argument.

EOF#

Example

```
IF EOF#C% THEN PRINT "end of file"
```

Introduction

EOF# is used to find whether the end of a file has been reached. If the end of the file has not been reached then the value 0 (FALSE) will be returned otherwise it will return the number value 65535 (TRUE).

EOF# knows which file to look at from the variable that must be typed in after it. This contains the channel number and is obtained from the function OPENIN or OPENOUT, described elsewhere. For example

```
C% = OPENIN"file"
IF EOF#C% THEN PRINT"Empty"
CLOSE#C%
```

will open a file called "file" and then test to see if the end of it has been reached. If it has, it prints a message to say that it is empty, then closes it.

Definition

EOF# is a boolean function that takes a variable containing a channel number as argument and returns the value TRUE if the end of the specified file has been reached or FALSE if it has not.

EOT

Examples

```
IF EOT THEN A%=5
UNTIL EOT
```

Introduction

EOT is used to tell if the cursor has reached the last character of the document or a segment. If it is at the end of the text then EOT will produce the value 65535 (TRUE) and Ø (FALSE) if not. EOT can be used on both the main text and also on any of the ten segments and so you must be very careful to check which is selected. The SELECT command is useful here.

Note that the EOT will only return TRUE if it is on the last character of the document. When merely moving down lines, you may reach the last line without being on the very last character. It is often best to position the cursor deliberately at the end of the line before testing, using the command:

```
CURSOR AT 39
```

Definition

A boolean function that returns the value TRUE if the cursor is at the end of the main text or one of the segments. If the cursor is not at the end then it will return FALSE).

EXT#

Examples

```
PRINT EXT#R%

IF EXT#Z%>1000 THEN CLOSE#Z%
```

Introduction

As in BASIC this is a function that finds out the length of a file. It gives the length, in bytes, of the file by its channel number. For instance

```
D% = OPENIN "file1"
PRINT EXT#D%
```

will print the length of the file called "file1".

Definition

A function which takes a variable containing a channel number as argument and returns the length in bytes of the file specified. The file is opened in order to find its extent and is therefore of limited use on a cassette system.

FALSE

Examples

```
UNTIL FALSE
```

```
IF A%=FALSE THEN PRINT "Oh Dear!"
```

Introduction

In Wordwise-Plus, as in Basic, **FALSE** is represented by the value 0.

Definition

A function that returns the value 0.

FREE

Examples

```
PRINT FREE
X%=FREE-10
```

Introduction

When editing a piece of text in Wordwise-Plus, whether it is your main document or one of the ten segments, there is always a number in the top right hand corner of the screen. This tells you how many characters are still free in memory. You can find out what this number is immediately from one of the menus or when running a program by using the function `FREE`. For example

```
X% = FREE
```

will put the number of characters free into the variable X%.

Definition

A function returning the number of characters free in memory.

GCF$#

Examples

```
D$ = GCF$#Y%
IF GCF$#Z% = "Y" THEN PRINT "Yes"
```

Introduction

GCF$#, which is short for "Get Character from File", is a way of reading the next character from a file on your disc or tape. In many ways it is similar to another function **BGET#** which also reads the next character from a file but returns the ASCII code, rather than a character. For example, if the character "A" is next in the file then the command

```
PRINT BGET#C%
```

will print the number 65, the character **A**'s ASCII value, whereas the command

```
PRINT GCF$#C%
```

will print the character **A** on the screen. Notice that in both cases the commands are followed by a variable. This contains the channel number which must be obtained using the **OPENIN** function before trying to use either of these commands.

Definition

A function which returns the next byte in a file as a single character string. The file is specified by the channel number supplied.

GCK$

Examples

```
UNTIL GCK$ = "Y"
```

```
LET C$=GCK$
```

Introduction

`GCK$`, which is short for "Get Character from Keyboard", waits for a key to be pressed on the keyboard and then returns that character. This is similar to the function `GET` except that `GCK$` returns a character that can be put in a string variable like C$. As with `GET`, the character typed will not appear on the screen when the key is pressed.

Definition

A function that waits for a character from the keyboard which is then returned as a single character string.

GCT$

Examples

```
UNTIL GCT$=" "
D$ = GCT$ + A$
```

Introduction

Like GCK$ and GCF#, GCT$ is a way of reading a single character but from your text (or one of the segments) rather than the keyboard or a file. GCT$ reads from the position of the cursor which is then moved on one place.

Definition

A function that reads the character at the cursor position of the currently selected piece of text returning a single character string.

GET

Example

```
H% = GET
PRINT CHR$(GET)
```

Introduction

GET is a function that waits for a key to be pressed on the keyboard and then returns with the ASCII value of that key. It is useful whenever the computer needs to wait for a reply from the user before it continues.

Definition

GET is a function that waits for the next character from the keyboard and then returns that character's ASCII value.

GLF$#

Examples

```
A$=GLF$#C%

PRINT GLF$#E%
```

Introduction

GLF$#, which stands for "Get Line from File", lets you read a whole line of text from a file on your disc or tape in one go. By a line we mean a sequence of characters up to the first carriage return or until a total of 255 characters have been read in, whichever comes first. As with BGET# and GCF$#, whenever GLF$# is used it must be followed by a variable that contains a channel number. This tells the computer which file to use. For example, if we have a file called "file1" that contains a string of 1000 character 'A's without any carriage returns, then

```
C%=OPENIN"file1"
A$=GLF$#C%
```

will put a string of 255 character A's into the variable A$.

Definition

A string function that reads, from a file, a sequence of characters up to the first carriage return or up to a total of 255. The file is specified by a channel number which is held in the variable that GLF$# takes as an argument. This will generate a filing system error if you try to read beyond the end of a file.

GLK$

Examples

```
A$=GLK$
IF GLK$="wrong" THEN PRINT "reject"
```

Introduction

Another way of reading a line of text into the computer is using the function GLK$. This is short for "Get Line from Keyboard" and it waits for the user to type in a series of characters at the keyboard and then press RETURN. The text that is entered then gets returned as a string without the carriage return character on the end.

Unlike GCK$, when you enter text using GLK$, the characters are actually displayed on the screen as they are typed, allowing correction, rather like the INPUT statement in Basic.

Definition

A string function that waits for a series of characters terminated by a carriage return to be typed on the keyboard and then returns the resultant string.

GLT$

Examples

```
A$ = GLT$

UNTIL GLT$=""
```

Introduction

GLT$, stands for "Get Line from Text", and like GLF$# and GLF$ reads a sequence of characters. It does this though from the text you are editing rather than from a file or the keyboard as with GLF$# and GLK$. GLT$ starts reading the characters from the position of the cursor in the text and stops at either the first carriage return or when a total of 255 characters have been read. If the end of the text is reached before either of these conditions happen then a 'beep' will be produced, indicating that it has not managed to get a whole line. This makes little difference as you will still have read all the characters but up to the end of the text instead of a carriage return for example. When GLT$ has finished, the cursor is moved to the first character beyond those that have already been read.

You must be careful not to confuse a line on the screen with what GLT$ actually reads in. This is because if there are no carriage returns in your text then GLT$ will read as much as 6 lines of the screen all in one go.

Definition

A string function that reads a sequence of characters starting from the cursor position of your main document or segment and up to the first carriage return or a total of 255 characters.

LEN

Examples

```
X% = LEN A$

Y% = LEN"FRED"
```

Introduction

Just as in Basic, **LEN** counts the number of characters in a string. For example

```
B%=LEN "Computer Concepts"
```

would give **B%** a value of 17.

Definition

A function that returns the number of characters in a string.

OPENIN

Examples

```
A%=OPENIN("FRED")
Z%=OPENIN A$
```

Introduction

Whenever you need to read information from a disc or a tape you must always tell the computer to prepare itself beforehand. This is done using the function `OPENIN` which opens the named file for reading into the computer. The result you get back is called the channel number and is the number you give to the other file handling commands to tell them which file they should be working on. For example

```
X% = OPENIN("text")
```

will open the file "text" for reading and then put the channel number in X%. This can then be used by a command like `BGET# or EXT#` for example.

Definition

A function which attempts to open a file for input to the computer. If it is successful then the function returns the channel number allocated by the computer's file system but if it is not then the value 0 will be returned.

OPENOUT

Examples

```
Y%=OPENOUT(G$)
B% = OPENOUT"chapter"
```

Introduction

Before you can record data on a disc or tape using the command BPUT# you must first open a file so that the computer knows where to send the information. OPENOUT does this and gives you a channel number that must be stored in a variable which BPUT# uses to direct the data to its correct destination.

Definition

A function which returns the channel number of an output file. If a file of the same name exists then that file will first be deleted. If no file exists then one will be created.

PTR#

Examples

```
PTR#X%=PTR#X%+10
PTR#Y%=0
```

Introduction

This statement selects which character is to be read or written next in a file on your disc or network but unfortunately not your tape. The file is specified by following `PTR#` with a variable that contains the channel number. This is obtained from either the function `OPENIN` or `OPENOUT` depending on whether you are reading from or writing to the file. `PTR#` can appear on both sides of an equal sign and so is both a statement and a function.

Definition

A statement and function which allows the programmer to move a pointer in a serial file thus enabling random access.

STR$

Examples

```
TYPE STR$(A%)
A$=STR$(A%)+"."+STR$(B%*100)
```

Introduction

Some commands require a string, rather than a number, as a parameter. For instance it is not possible to directly `TYPE` the value of X% into text. The `STR$` function converts a number into its corresponding string. For instance

```
PRINT STR$(10) + STR$(20)
```

would give as its answer `1020`, showing that the result is two strings joined end-to-end.

Definition

A function which takes a numeric expression and converts the result to the equivalent string form.

SOT

Examples

```
REPEAT
  CURSOR UP
UNTIL SOT

IF SOT THEN PRINT "At top"
```

Introduction

SOT stands for "Start of Text" and is a way of telling if the cursor is at the first character position of your text or segment. If this is so then the number 65535 (TRUE) will be returned and if not you will get the value 0(FALSE).

Note that SOT will only return TRUE if the cursor is on the first character of the first line. You may use the command

```
CURSOR AT 0
```

to ensure that the cursor is brought to the far left of the current line, before testing with SOT.

Definition

A boolean function that returns the value TRUE if the cursor is at the start of the text (or segment) and FALSE if it is not.

TIME

Examples

```
LET TIME=0

IF TIME>100 THEN PRINT "One second gone"
```

Introduction

`TIME` is used to set or read the computer's internal timer which counts in one hundredth of a second intervals, just as in BASIC.

Definition

A pseudo-variable which sets or reads the lower two bytes of the internal elapsed time clock.

TRUE

Examples

```
PRINT TRUE
IF G%=TRUE THEN PRINT"Ok"
```

Introduction

TRUE is represented in Wordwise-Plus by the value 65535 which is &FFFF in hexadecimal, the largest number which can be represented in Wordwise Plus.

Definition

A function that returns the value 65535.

VAL

Examples

```
X%=VAL"69"
Z%=25 + VAL(A$(
```

Introduction

As in BASIC, the VAL function takes a string such as "45AB" which contains numeric characters and then returns the number that is represented, 45 in this case. One important point is that the character string must start with a digit otherwise the function will return 0. Take the following for example.

```
PRINT VAL("1234FIVE")
```

will return the number 1234.

```
U% = VAL("A65")
```

will store the value 0 in the numeric variable U%.

An important difference between the Wordwise-Plus function VAL and the BASIC function VAL occurs because Wordwise-Plus does not cater for signed numbers, i.e. no distinction is made between positive and negative numbers. In Wordwise-Plus

```
PRINT VAL("-45")
```

will return 0 because the first character of the string is not a digit whereas in Basic the number -45 would be returned. Similarly

```
PRINT VAL("+32")
```

will also return 0 in Wordwise-Plus but 32 in Basic.

The opposite of this function is STR$.

Definition

A function that extracts digits from the start of a character string and returns them as a number. If the string does not start with a digit then the value 0 will be returned.

VARFREE

Examples

```
PRINT VARFREE
X% = VARFREE
```

Introduction

In Wordwise-Plus string variables like A$ and E$ are always stored in an area of memory which is just over 600 bytes long This means that in all your string variables from A$ through to Z$ the total number of characters cannot add up to more than 613. If you already have this number of characters in your strings but try to add more then you will get the error "`No $ room`" telling you that this area is already full. This can be very annoying to find out when it is too late. The `VARFREE` function determines how many more characters are allowed.

Definition

A function which returns the number of bytes still free in the string variable space.

WORDS

Examples

```
W% = WORDS -13
PRINT WORDS
```

Introduction

When editing a piece of text there is a message in the top left hand corner of the screen which tells you the number of words currently in your document or segment. The **WORDS** function will return this number.

The word count is held in the variable **W%**. Changing the value in **W%** effectively changes the current word count. Asking the value of **W%** is also the same as using the function **WORDS**, except that its meaning is not so clear. The following will both print the current word count:

```
PRINT WORDS
PRINT W%
```

The only real difference is that you can set the value of W%, but cannot assign to **WORDS**. Altering the text using a program does not update the word count. If you know how the number has changed, simply add or subtract it from **W%** but if you lose track of the correct word count altogether, you may use the **RECOUNT** command to re-count the words from scratch.

Definition

A function that returns the word count for your text or segment, whichever is selected.

8. Segments

If you are new to Wordwise-Plus then we suggest that you do not read this chapter now – leave it to later!

There are 10 segments of memory available in addition to the main text area. Available memory is divided between the 10 segments and the main text area – there is no limit on any of the individual areas.

Segments have been introduced for four main reasons:

Firstly: so that you can easily add your own extra editing functions.

Secondly: so that you can use each segment as a "jotter" for notes – the jotter can be saved and then re-loaded into the main text area to form the basis of another chapter or set of notes.

Thirdly: so that you can generate programs within Wordwise-Plus to "drive" the word processor. These programs can handle simple mathematical as well as string operations and can deal with random access files.

Fourthly: since it provides a powerful language programming capability within a word processor, application programs can be published which can completely change the way the product behaves. It can be a word processor to one person and a quite sophisticated invoice generator to another person.

The new language capability allows more powerful features such as mail merging and index generation. Some examples of new features are included on the Wordwise-Plus cassette.

Because the concepts are new, this section is more tutorial than the rest of the manual. Probably the easiest way to start is with a practical and useful example.

Idea Number 1 – a notepad facility

Type the following text into Segments 0, 1 and 2 respectively.

Type into segment 0:-

```
SELECT TEXT
DISPLAY
```

Type into segment 1:-

```
SELECT SEGMENT 2
DISPLAY
```

Type into segment 2:-

```
REM notepad
```

Having done this, go to the Main Text area and type in a few notes. Then press **SHIFT f1**. This executes the program in segment 1, which simply selects the notepad area, segment 2. You will find that you are in Segment 2 – your new notepad. Press **SHIFT f0** to return to your Main Text. These simple programs have added two new functions to Wordwise-Plus and these functions are activated by use of **SHIFT** together with function keys. **SHIFT f1** will try to run the program in Segment 1 and so on.

Idea Number 2 – transposition

When typing text in, a very common mistake is to trasnpose two letters – "transpose" or "trasnpose"? It is easy to create a new command and asign it to **SHIFT f5** to transpose two letters – to correct the mistake. Type this program into segment 5

```
A$=GCT$
DELETE LEFT
CURSOR LEFT
TYPE A$
DISPLAY
```

To correct the word "trasnpose", position the cursor on the letter "n" and press **SHIFT f5**.

Idea Number 3 – an unformatter

Have you ever wanted to take a text file and strip out all the RETURN codes at the end of each line – but leave in the RETURN RETURN at paragraph breaks?. Files that have been spooled – or come over electronic mail – are usually split into lines about 80 characters long. If you wish to edit the document then you will need to "unformat" it first.

This program does just that.

First it searches for all RETURN RETURNs and replaces them with "ZCZC".

Secondly it replaces all remaining RETURN characters with a SPACE.

Thirdly it restores the RETURN RETURN paragraph breaks.

Type the following program into segment 3:-

```
REM unformatter – removes single
REM Return characters.

SELECT TEXT
CURSOR TOP
REPEAT
  REPLACE "||R||R","ZCZC"
UNTIL EOT

CURSOR TOP
REPEAT
  REPLACE "||R"," "
UNTIL EOT
```

```
CURSOR TOP
REPEAT
 REPLACE "ZCZC","||R"
UNTIL EOT

CURSOR TOP
DISPLAY
END
```

Note the use of "||R" to represent a **RETURN**. Other special characters are represented in strings thus:

||T – represents **TAB**
||R – represents **RETURN**
||G – represents **f1**
||W – represents **f2**

If the document that you are unformatting starts each line with a number of spaces – as it would if it were spooled with Left Margin setting of 5 (LM5) – then you will need to search for "||R ||R" – that is ||R, 5 spaces and another ||R to replace with "ZCZC".

Idea Number 4 – cash calculations

If you wish to ask a question from the program in a Segment then you will need to

First: move the cursor to the correct place on the screen – for example the top line -
this can be easily done with `VDU 31,12,0`
But note that this "cursor" is not the editing cursor which is controlled by the "`CURSOR`" command.

Second: delete the text there by overprinting with a `PRINT` statement.

Third: `PRINT` a message asking the question.

Fourth: use Get Line from Keyboard (`GLK$`) to input a line of text.

Fifth: convert that line to a numeric.

etc.
Try typing the following program into segment 4:-

```
VDU 31,12,0
PRINT "Cost in pence? ";
PRINT " ."
VDU 31,14,0
A%=VAL(GLK$)
A$=STR$(A% MOD 100)
IF LEN(A$)=0 THEN A$="00"
IF LEN A$=1 THEN A$="0"+A$
A$=" "+A$
A$="'"+STR$(A% DIV 100)+A$
TYPE A$
DISPLAY
```

Notice also the use of the commands `PRINT` and `TYPE`. `PRINT` causes output to be placed on the screen but not in the text, whereas `TYPE` will place the output in the Main Text.

The Segment can be called from the Main Text display by pressing **SHIFT f4**.

Since the questions asked can be varied and the user's response can be held in a variable, it is possible to keep totals and so on and thus generate a variety of cash handling programs such as invoicing.

It is also possible to read and write from data files, so customer numbers could be used to access name and address data whilst invoice details could be written to file.

Idea Number 5 – editing multiple files

Often it is useful, or necessary, to split a large piece of text into a number of smaller files. It is easy to arrange a function key to move "back" a file or "forward" a file.

In this program a list of files is held in Segment 6, the "back" program in Segment 7 and the "forward" program in Segment 8 The variable F% holds the number of the file in the list that is currently being edited. So if the Intro is being edited then F% holds 1. F$ is used to hold the current filename, since menu load and save options automatically copy the last used filename into F$.

List of files in segment 6:-

```
INTRO
CHAPT1
CHAPT2
CHAPT3
CHAPT4
CHAPT5
CHAPT6
INDEX
```

'Backwards' Program in segment 7:-

```
REM Move back one
REM first save the present file
REM assume F$ is already set up
SELECT TEXT
SAVE TEXT F$
REM get previous filename using F%
SELECT SEGMENT 6
```

```
CURSOR TOP
DOTHIS
 CURSOR DOWN
TIMES F%
F$=GLT
REM and load the file in
SELECT TEXT
LOAD TEXT F$
DISPLAY
```

A similar technique is used in Segment 8 to move "forward" a file. The difference being that an extra line

```
CURSOR UP 2
```

would be inserted before the line

```
F$=GLT$
```

in order to move back to the filename before the current one.

This technique does work – but a much more complete version of these routines and other related ones are included on the cassette.

9. Variables and Labels

Variables

There are two types of variables that can be used in Wordwise-Plus: numeric variables and string variables.

Numeric variables

In Wordwise-Plus a numeric variable name is just a single upper case letter followed by a percent sign, for example `A%` or`T%`.

This means that there are only 26 of them, of which it is only really safe to use 24, but more of that later.

These variables can hold any integer between 0 and 65535 inclusive. They cannot hold negative numbers nor can they hold real numbers (i.e. those with a decimal point). If you really need to use reals do it in BASIC!

Like BASIC these variables are always present and if one has not been used before then will automatically be set to zero. In fact, the values of these will still remain if you go from Wordwise-Plus to BASIC or vice versa but, be warned, if you change to BASIC and then back again your numeric variables will remain but your text might not.

As mentioned above, there are 26 of these variables but only 24 can be used with complete safety as normal variables. This is because the variables `W%` and `P%` are used by Wordwise-Plus itself.

`W%` is used to hold the word count dispayed in the corner of the screen in Edit Mode. This means that every time you insert a

word into your text during editing, W% will be increased by one and of course if you delete a word then it will be decreased by one. You can also change the value of W% just like any of the other variables but this will mean that you will lose your total word count. However,this can be retrieved using the command RECOUNT. See the function WORDS for more details on W%.

P% is used to hold a copy of the current page number when a document is being printed, previewed or spooled. This means that as soon as you use one of these commands then the value of P% will be changed. Remember that P% is not reset at the start of PREVIEW, PRINT, or SPOOL commands (except when performed from the menu) unless preceded by a DEFAULTS command.

String variables

Like numeric variables, a string variable is also a single upper case letter but followed by a dollar rather than a percent sign. The valid variables are therefore A$, B$, etc. to Z$. These are also permanently present and have a default value of "", i.e. a null or empty string. But, unlike numeric variables, string variables are destroyed if you enter BASIC and so cannot be transferred.

Any one string variable can hold a maximum of 255 characters, but the total number of characters that are allowed in all of the string variables, A$ through to Z$, cannot be more than 613. In most programs this limit will never be noticed, but in a few which store a large number of strings a careful watch should be kept on the number of characters stored. Exceeding the string storage limit will generate the error 'No $ Room'. This can be avoided by using the VARFREE function to determine the number of characters which may still be stored before over-running the limit.

When string comparisons are performed, the operations take place within a limited space of about 250 bytes. The consequence is that the total number of characters in the strings being compared must not exceed about 250. It precludes, for instance, the command

```
IF A$=B$ THEN PRINT "YES"
```

when the length of the two strings added together is more than about 250 characters.
This constraint rarely matters, but usually shows itself when using commands to read a line from text or from file. These commands will read up to a RETURN or 255 characters, whichever comes first. Therefore when reading text which contains no RETURN characters the strings read will always be 255 characters long. They cannot then be compared with another string. This can be overcome by inserting an extra test into the program to see if the string length or total length of several strings will be greater than 255 characters, *before* carrying out any comparison. Use the LEN function to find the length of a string.

The only string variable that has other uses is F$. This holds the last filename that you have used for loading or saving from the Main Menu and will appear on the screen when selecting either of these options from the Main menu. For example with the Main menu in front of you if you type

```
:F$="INTRO"
```

and now if you select an option from 1 to 4 you will get the prompt

```
Previous filename — INTRO

Please enter filename
```

Labels

A label is a full stop followed by a series of characters and marks a point in the program that the computer must jump to. BASIC has line numbers to tell the computer where to `GOTO` but Wordwise-Plus needs labels.

```
.this-is-one

.2

.&label
```

These are all perfectly valid.

.but this is not
.and-this-one-would-be-longer-than-the- width-of-the-screen

The two above would not be valid. The first one includes spaces. A label must not have any other words following it on the same line. Any program statement in Wordwise-Plus cannot span two screen lines and so the second of these examples would be invalid as well.

The only commands that jump to labels are `GOTO` and `PROC`. See these keywords for further details.

10. Operators

Operator precedence

As in the BASIC language, so in the Wordwise-Plus language there are a number of operators which can be used with numbers, numeric variables, strings and string variables. An operator is the part of an expression that 'operates' on other parts. For example the '+' addition sign.

When Wordwise-Plus is calculating a string or numeric expression such as

```
A% = D%*3 + 3456/23 - ASC(A$)
```

or

```
F$ = "B.chapt"+STR$(F%)
```

it works out some parts of the total calculation before other parts – as we do in normal arithmetic. For example multiplication is always performed before addition so that

```
3+7*5
```

is 38 and not 50. There is a strict "pecking order" for operators and it is given below. You will find that the order is the same as for BBC BASIC and you may wish to refer to the BBC Microcomputer User Guide page 144 for more information.

This is the order of precedence for all the Wordwise-Plus operators.

Group 1
functions (e.g. `GET`, `PTR#`)

brackets

? indirection operator

Group 2
none in this group

Group 3
`*` – multiplication
`/` – division
`DIV`
`MOD`

Group 4
`+` – addition
`-` – subtraction

Group 5
`=`
`<>`
`<`
`>`
`<=`
`>=`

Group 6
`AND`

Group 7
`OR`
EOR

Boolean operators
The operators

```
AND
OR
EOR
```

are provided in Wordwise-Plus. They can be used in two ways: as logical operators or as bitwise operators. These are described in the BBC microcomputer User Guide on page 205. The following examples illustrate the two uses.

Logical:

```
IF A$="YES" OR A$="yes" THEN A%=A%+1
IF X=5 OR Z=7 THEN PRINT "completed"
```

Note that `PRINT 6=6` will print "`65535`" which is the numeric value of `TRUE` in Wordwise-Plus.

Boolean:

`PRINT &66 AND %00001111` (would give the result "6")

```
G% = CHR$((ASC GCF#F%) And &7F)
```

Integer arithmetic operators

Wordwise-Plus provides the following integer operators

`+` ... addition
`-` ... subtraction
`*` ... multiplication
`/` ... division
`MOD` ... modulus
`DIV` ... whole number division

String operator

The only string operator is "`+`" which is used to join two strings together. For example

```
A$="day"
B$="Tues"+A$
PRINT B$
```

will print "`TUESDAY`".

Comparison operators
Both string and numeric values may be compared using the relational operators

= .. equal to
< .. less than
> .. greater than
<= .. less than or equal to
>= .. greater than or equal to
<> .. not equal to

Thus

```
IF "X">"A" THEN PRINT "Yes"
```

will print "Yes".

Indirection operator
The query indirection operator is available for reading from memory only, thus

```
PRINT ?&F4
```

is acceptable but

```
?&f4 = 11
```

will not be accepted. See the BBC Microcomputer User Guide section 39 on page 409 for more information.

Format effector
The tilde character (~) can be used in the PRINT command to print numbers in hexadecimal format. Thus

```
PRINT ~495
```

will print "01EF". The tilde does not work in other situations – for example with STR$. Note that the tilde character is displayed on the screen as ÷.

Base indicators
As well as being able to enter numbers in decimal and hexadecimal, Wordwise-Plus can accept numbers in octal and binary. The available prefixes are

& – hexadecimal
@ – octal
% – binary

Thus

`PRINT %10101111` will produce `175`
`PRINT @177` will produce `127`

11. Files

If you wish to edit a very large file then it is strongly suggested that you break the file down into a number of smaller sections. Not only is this advisable because is will prevent you losing a large file in the event of accindent, but it is also necessary because of the limited memory on the BBC microcomputer.

It is frequently necessary to process long files however and the cassette supplied with Wordwise-Plus contains programs which will enable you to do this easily. Details of how to load these Wordwise-Plus programs are given on the instructions accompanying the cassette.

Technical information
Listed below are a few technical points which may be helpful for those writing Wordwise-Plus programs and procedures.

Using an Acorn DFS, Wordwise-Plus can support the maximum of five data files open at the same time. Obviously it is not possible to use more than one file with cassette.

Some of the Acorn file systems close all open files when an error occurs – others do not. As a result you should take care to close all files whenever possible, at the start or end of a program for instance. Use the command

```
CLOSE#0
```

Wordwise-Plus does not leave its own files open during editing – it only opens its own files briefly when loading or saving files on menu options 1,2,3,4 and 8 and the corresponding commands provided in the programming language.

12. Printers

Menu option 6 is used to print your text and it is probable that it will work with your printer immediately. However you may find that there are a few problems with some characters – for example the pound sign may not print correctly and effects such as underline and italic may not work. This chapter tries to deal with some of the common printer related problems that may arise.

Serial or Parallel

If you plug your printer into the socket under the computer marked "printer" then it should at least print normal text – though there may still be problems with some characters. That socket is known as the "Parallel printer" socket – or the "Centronics compatible" socket.

If you have a printer with a "serial" interface then it must be plugged into the socket at the back of the computer marked "RS423". This is a serial interface. Printers which plug in here may be said to have a serial interface variously named- "RS232", or "RS232C", or "V24". All these names mean pretty much the same thing.

If you have one of these printers then you must type

```
*FX 5 2
```

from the main Wordwise-Plus main menu every time that you turn the system on – or if you press **CTRL** **BREAK** on the computer.

You may well have quite a few other problems initially, but your dealer should be able to sort these out for you without much trouble – persevere!!

These are the two main problems:

firstly an electrical problem of getting the right cable to connect the printer to the computer and getting all the plugs in the right way round.

secondly getting the "baud rates" of the computer and printer the same.

If, when you try to print a document, your printer won't do anything, (and assuming you have plugged it in and switched it on!) then there is probably a problem with the lead between the computer and the printer.

If your printer does print characters but they are rubbish then you have probably got the "baud rates" set to a different speed on the printer and computer.

If the problem is the lead then:

either get out both the BBC computer User Guide and refer to Chapter 38 starting on page 404 and the printer's technical manual and try to sort out the problem yourself.

or take the printer and computer to your dealer and get him to solve the problem – it will probably cost you the price of a new lead!

If you are getting garbage on the printer then it is probably a "baud rate" problem. These are rather easier to sort out than lead problems and by reference to the User Guide and the printer manual you will probably be able to work out which magic incantation of `*FX 7` and `*FX 8` you will need. See page 407 of the User Guide.

Underline and Double-strike print
With most modern printers, dot-matrix or daisy wheel, it is possible to underline letters and to produce a bold effect by double-striking the paper. Turning these effects on and off is achieved by sending a command to the printer. The problem is that different printers need different commands.

Wordwise-Plus is set up to send the correct commands for an Epson Rx-80 printer and related models. If wish to use another model then you may have to tell Wordwise-Plus to send different commmands, ones suitable for your printer.

Here are the sequences that are generated by Wordwise-Plus by default. They are for an Epson Rx-80.

Action	codes are	Printer manual description
Underline Start	27,45,1	ESC "-",1
Underline End	27,45,0	ESC "-",0
Double-strike Start	27,71	ESC "G"
Double-strike End	27,72	ESC "H"

The commands can easily be re-configured to drive any printer supported codes. If you wish to drive a Qume Sprint 9 then you can re-configure Wordwise-Plus by entering the following commands at the top of your text.

Action	Enter	Printer manual description
Underline Start	RPS 0,27,73	ESC "I"
Underline End	RPS 1,27,74	ESC "J"
Double-strike Start	RPS 2,27,75,51	ESC "K"

13. Changes from Wordwise

These are some of the updates from Wordwise.117 not including any commands or features for use with segments.

1 version number and date
2 Menu options 1 and 3 now `*SAVE` all text on DFS, ADFS and NFS. A side effect of this is that any markers are deleted from the text.
3 Menu option 2 and 4 now `*LOAD` all text from DFS, NFS and ADFS. After having loaded the text this then changes all illegal characters to || codes.
4 The word count is now held as a 16 bit integer in W% and P% holds a copy of the page number.
5 **CTRL R** deletes any markers in the text
6 **CTRL W** causes a recount of words
7 **CTRL F** toggles on-screen formatting
8 It is not possible to insert a marker when there is no room
9 It is not possible to scroll down when there is no room.
10 The "`No Room!`" error occurs at 4 characters free
11 When loading text pressing **ESC** after the filename has been entered will not destroy text
12 Loading a null file will not delete text
13 Menu option 2 prompts before loading text over existing text
14 Menu options 1 and 3 will ask if a file is to be overwritten
15 The menus are tokenised to save space but are a little slower
16 Wordwise plus works with Aries B20 board, Watford Electronics RAM board and 6502 2nd processor
17 Beeps when asking "`Marked section only`"
18 embedded command `BP` at end of text will not now start a new page but will end current page
19 `HP` and `FP` are now one line different (6)
20 The event vector is no longer used
21 visible page breaks
22 Default left margin now 0 (different from SOME versions)

23 Error messages are now removed at next key press
24 Menu option 8 now spools control codes
25 Get file now gets control codes
26 EM does not send an extra carriage return on every page
27 default footing "f1 CE f2 PAGE f1 PP f2"
28 Menu option 5 allows searches for carriage returns, greens, whites and tabs
29 When printing, previewing or spooling a marked section all the embedded commands up to this section are taken into account
30 Print buffer is only cleared on menu options 6,7,8
31 GF will now produce an error if it cannot open a file
32 OC and ES commands can now accept characters as well as character codes.
33 All embedded command numbers can now be any valid numeric expression
34 The number of tab stops has now been increased to 14 or as many is fit in one DT
35 Word count =0 if there are no words in the text
36 Search and Replace now accepts wild card character # as well as ||R, ||G, ||W and ||T
37 Saving text onto tape and pressing ESC leaves cursor at top
38 New menu option 9
39 Pressing Shift and function key now runs segment program
40 Pressing space bar when previewing pauses the output
41 PC command accepts a string expression. A literal character must be between quotes.
42 New Embedded commands:

*	Operating system call	PA	Pause
DE	Double-strike end	PF	Print File
DS	Double-strike start	PS	Print string
ES	Escape sequence	RPS	Redefine print sequence
FI	Fully indent	SEG	Execute segment
LNE	Line number end	UE	Underline end
LNS	Line number start	US	Underline start
OPS	Output sequence		

14. Error Messages

Bad argument
The argument or parameter is missing or wrong.

Bad channel
Either the syntax for file handling command is wrong or the channel has not been opened.

Bad command
Cannot understand command

Mistake
This can occur for several reasons but usually indicates that an operand such as a number, variable or function is expected but cannot be found.
eg.`PRINT 6*hello`

Bad operator
Computer expects an operator but cannot find one.
eg. `PRINT 6Z`

Bad Params
This occurs if the file is not recognised as a valid parameter file when loaded with `LOAD PARAMS`

Can't execute
This usually occurs when a program tries to change itself which includes even moving the cursor.
eg. the following program in segment 0:-

```
SELECT SEGMENT Ø
DELETE TEXT
```

Can't open file
File can't be opened using `OPENIN` or `OPENOUT`

Escape
The **ESCAPE** key has been pressed.

Error
This occurs when an expression is malformed.
eg. **PRINT 6***

File too long!
The length of a file indicates that it will not fit into available memory. This cannot be checked on cassette.

MARKERS!
The computer expects two and only two markers to be set.

Maths error
A mathematical operation is out of range. Only integers between 0 and 65535 are allowed.

No REPEAT
This occurs if the interpreter comes across an **UNTIL** or **TIMES** statement without a previous **REPEAT** or **DOTHIS**.

No room!
The computer's memory is full.

Missing "
The computer expected to find a double quote.

Missing)
The computer expected to find a closing bracket.

No such file!
This occurs when you try to load a file that does not exist.

No such label
The label for a **PROC** or **GOTO** cannot be found.

No $ room
The total number of characters in all of your string variables cannot exceed 613.

`Syntax error`
The computer doesn't understand!

`Type mismatch`
This occurs if th computer is expecting a string and gets a numeric or vice versa.
eg. `X%="FRED"`

`$ too long`
The maximum length of a string is 255 characters.

`/ by zero`
Division by zero is not allowed.

Embedded Commands Reference List

Command	Range	Default	Break	Functions
BP		no	yes	Begin new page
BS A%	0 to 50	6		Bottom Space
CE A%	1 to 200	1	yes	Centre line(s)
CI		no		Cancel Indent
CO		yes		Continuous Output
CP A%	0 to PL		yes	Conditional Page
DE		27,72		Double-strike End
DF		f1 CE f2 PAGE f1 pp f2		Define Footing
DH				Define Header
DM				Disable Message "Paper"
DP	32 to 255	96		Define Pound
DS		27,71		Double-strike Start
DT	0 to 180	10,20		Define Tabs
EM		no		Enable Message "Paper"
EP		no		Enable Paging
ES		none		Escape Sequence
FI		no		Fully Indent
FP A%	0 to BS	3		Footing Position
GF		no		Get from File
HP A%	0 to TS	3		Top Space
IN A%	0 to LL-10	0		Indent
JO		no		Justify On
LL A%	10 to 180	70		Line Length
LM A%	0 to 150	0		Left Margin
LNE		no		Line Number End
LNS		no		Line Number Start
LS A$	1 to 50	1		Line Spacing
NJ		yes		No justification
OC A%,A%		none		Output Control code(s)
OPS A%	0 to 9	none		Output Print Sequence
OS A$		none		Operating System call
PA		off		Pause
PC A$	"!" to "z"	"|"		Pad Character
PF A$		no	yes	Print File
PL A%	10 to 200	66		Page Length
PN A%	0 to 65535	1		Page Number
PP		no		Print Page number
PS A$		""		Print String
RPS A%,A%	0 to 9			Redefine Sequence
SEC A%				Execute SEGment
SP A%	0 to 200	0	yes	Space down lines
SS		yes		Single Spacing
TI A%	0 to LL-10	0	yes	Temporary Indent
TS A%	0 to 50	6		Top Space
UE		no		Underline End
US		no		Underline Start
★		none		'Star' command

'Break' column indicates if it forces a new line when printing

15. Reference tables

Command List

There follows a list of all the commands that may be included in programs or entered from the menus by preceding the command with a colon.

Under some circumstances the "Can't execute" error will occur. These are commands which, if executed, would corrupt or alter the running program. This includes cursor commands – i.e. it is not possible to say CURSOR RIGHT if the currently selected segment is the same as the one that is running.

<Expr> means the command should be followed by a numeric expression i.e. a number, integer variable, function etc.
<$expr> means the commands should be followed by a string expression i.e. a string variable, text in quotes etc.

```
BPUT#    <%var,>     <Expr>
CALL      .........<Expr>
CLOSE#    .........<Expr>
CLS
CURSOR   RIGHT       <Expr>
CURSOR   LEFT        <Expr>
CURSOR   UP          <Expr>
CURSOR   DOWN        <Expr>
CURSOR   AT          <Expr>
CURSOR   TOP
CURSOR   BOTTOM
DEFAULTS
DELETE   AT          <Expr>
DELETE   LEFT        <Expr>
DELETE   MARKED
DELETE   MARKERS
DELETE   TEXT
DELETE   WORD        <Expr>
DISPLAY
DOLINE    .........<$expr>
```

```
DOTHIS
END
ENDPROC
FIND      .........<$expr>
FKEY      .........<Expr>
GOTO      .........<Label name>
IF THEN   .........<Expr>
LET      <%var>      =<Expr>
LET      <$var>      =<$expr>
LOAD     PARAMS      <$expr>
LOAD     TEXT        <$expr>
LOAD     TTC         <$expr>
NEW
OSCLI     .........<$expr>
PREVIEW  FILE        <$expr>
PREVIEW  MARKED
PREVIEW  PAGE        <Expr>
PREVIEW  TEXT
PRINT     .........<Expr>(;)
PRINT     .........<$expr>(;)
```

```
PRINT    ~          <Expr>(;)
PRINT    FILE       <$expr>
PRINT    MARKED
PRINT    PAGE       <Expr>
PRINT    TEXT
PROC     .........<Label name>
PTR#     .........<%var>
RECOUNT
REM
REPEAT
REPLACE  .........<$expr>,<$expr>
SAVE     PARAMS     <$expr>
SAVE     TEXT       <$expr>
SAVE     MARKED     <$expr>
```

```
SELECT   TEXT
SELECT   SEGMENT    (<Expr>)
SPOOL    FILE       <$expr,><$expr>
SPOOL    MARKED     <$expr>
SPOOL    PAGE       <$expr>
SPOOL    TEXT       <$expr>
SWAP     .........<Expr>
TIME=    .........<Expr>
TIMES    .........<Expr>
TYPE     .........<$expr>
TYPE     SEGMENT    <Expr>
UNTIL    .........<Expr>
VDU      .........<Expr>(,/;)
★        .........<TEXT>
.        .........<Label name>
```

Functions

Functions are commands that return a value, and so they would usually appear an equals sign. E.G. X%=len(A$)

```
ASC      <$expr>
BGET#    <%var>
CHR$     <%var>
EOF#     <%var>
EOT
EXT#     <%var>
FALSE
FREE
GET
GCF$#    <%var>
GCK$
GCT$
GLF$#    <%var>
```

```
GLK$
GLT$
LEN      <$expr>
OPENIN   <$expr>
OPENOUT  <$expr>
PTR#     <%var>
SOT
STR$     <%var>
TIME
TRUE
VAL      <$expr>
VARFREE
WORDS
```

Index