SOFTWARE COPYRIGHT

All enquiries to: Trojan Products
166, Derlwyn, Dunvant, Swansea SA2 7PF

```
This program contains the following
routines which may be loaded from TLW.
1:TLW2B_D 2:TLW2_D 3:TEACHER 4:P_TEST
5:DF_SHOW 6:DF_LTR 7:TLW_DF 8:DF_TEST
9:K_BASIC 10:K_TEST 11.TLW_ADPT
12:ADPT_CDE 13:DSYS 14:X_BASIC
```

The Last Word ...

Word-processor for 48k/128k Spectrum computers

INSTRUCTION AND REFERENCE MANUAL

The Word Processor

Additional Utility Programs

Adaptations for Printers, Interfaces and Disc Systems

This manual describes
***"The Last Word"** versions SP2 1.0 and SP3 1.0*
and associated software from Trojan Products

The Last Word ...

Section A
Introduction and Setting Up

"The Last Word ..." is a powerful word processor for use with all Sinclair and Amstrad Spectrum 48k and 128k computers. It provides for the user a wide ranging capability in the construction and storage of 'Text Files', and the ability to process and print them with selectively merged material (stored and manipulated elsewhere in the computer's memory) under the control of instructions lodged in the text file itself.

Additional software provided with the word processor includes a 'Data File' program with which the user can build a range of files, each holding eight five-field records as source material for the 'Mail Merge' facility, an extensive 'Key Define' program allowing the revision of up to 16 designated keyboard characters so they are shown on the screen and sent to the printer as foreign or graphics characters, and an 'Adaptation' program which will modify text files from other less flexible WP's which offer only a fixed line length file system (Tasword 2 etc.) so the files are fully compatible with the much more efficient variable margin format in "TLW".

If you have a Spectrum +3, you will find that the "TLW" disc is fully tailored to suit the built-in disc drive facilities and can be simply adapted to drive virtually any type of Centronics parallel or RS232 serial printer — see Section O.

If you have an earlier 48k Spectrum/Spectrum + or a 128k Spectrum/ Spectrum +2, the cassette tape of "TLW" may require some minor revisions to suit the printer/interface/disc storage system hardware you intend to use.

With some simple changes, the program is fully adaptable to the Rotronics Wafa-drive and most of the extremely effective disc systems such as Rockforts' Disciple and interfaces/equipment from Kempston, TRW/Beta, Opus and the various other companies who have involved themselves in the Sinclair/Amstrad Spectrum scene.

Personalising the program for 48k/128k Spectrum + and Spectrum +2 computers.

"TLW" as supplied will operate without modification with tape and microdrives for mass storage, sending Epson style printer control codes via a Kempston 'E' printer interface. If your equipment is different from the above you should first follow the detailed instructions in Sections O and P to adapt the program appropriately, and then save a 'Master' copy of "TLW" and the other associated software which will be specifically suited to your own system/hardware.

Before you start using the word processor, please note that —

- ☆ *The Command-Summary 'HELP' page is called by keying extended-mode 'H' at any time.*
- ☆ *"TLW"s alarm-clock is silenced and reset by pressing the 'T' key.*

☆ *You can 'Quit' from most "TLW" commands without effect by keying SYMBOL SHIFT 'Q'. If this does not work — press ENTER.*

☆ *To obtain the 'E' keys command mode, use the Spectrum CAP'S SHIFT & SYMBOL SHIFT keys together — the 'Write' in the header display will change to BRIGHT 'E' keys.*

We will assume from here on that any program adaptations you may need to make have been successfully sorted out, and that you are henceforward LOAD'ing your adapted version of **"The Last Word ..."** into your Spectrum directly from your own storage system.

We strongly recommend that you read through at least Sections A to E in this booklet before setting out on the keyboard. You will find the program simple to operate right from the start, and you should soon be able to produce documents of considerable scope, but never forget, it is YOU who must learn how to drive "TLW" to do it . . .!!

◆ ◆ ◆

Section B
Getting Started

Loading "The Last Word ..." into your Spectrum

Connect your printer/interface/microdrives/disc system, etc. to your Spectrum, and switch on. Load the program into the computer by pressing ENTER at the +3 'loader' menu, or by keying LOAD "" and playing the tape or whichever approach your own system demands. The computer will initially get the "TLW" Basic program in, then install the main machine code section and drop to the prompt at the main menu.

The main "TLW" menu

The 'starting-off' point for all the different parts of "TLW" is this six option menu. Some of the choices use software which is already in the computer at this stage, but others will need to fetch further program material from your storage system to provide the response to the task you have chosen. Entry to the word processor is by option-1 and this will now be described in detail. Other choices are described in the sections that specifically relate to them.

The word processor — main menu option-1

Three further questions will be asked by prompt-lines at the base of the screen immediately following the initial entry of this option. These are —

1. Keys [0-3] . . . To start with just press the ENTER key here, at a later stage you should read Section M — The Key-Define Facility — for a full understanding of this feature of "TLW", which allows you to re-define the screen and printer output of up to 16 keys on the Spectrum keyboard. Pressing ENTER leaves the current value of the KDF system unchanged — as shown in the menu at selection-1.

2. Initialise printer [y]: . . . (on non '+3' versions only). A 'y' entry here will lead to your pre-arranged printer/interface line(s) of Basic being run, or as usual a null ENTER passes this by.
3. Date: . . . A date (up to 20 characters long) entered here will be 'fixed' into the "TLW" m/c code, and made available to you as z$ in the Basic variables area. This can be useful when you are printing some letters — a z$ token will fetch and print the date (20 characters) instead of having to key it in each time. Once again just keying ENTER will avoid changing the stored data.

After the date prompt the word processor will be entered, showing you a clear 'page' with the copyright message and the version identification of the software. Press ENTER once more to clear the page, and you are now ready to go.

The screen display

"TLW" uses the top three lines of the screen for a header to display a variety of information about the current status of the program, the text file, and the position of the flashing cursor box within it. It uses the lower 20 lines of the screen to display a page of the text file. The right hand screen margin has some special marks giving information about the end of line status of each complete line of text. The screen (display) page can be set to operate with a width of 40, 48, 60 or 80 characters, entirely depending upon your own viewing requirements. This video page is your view of a section of the text file. The screen pitch you select has absolutely no effect upon the text file itself, it's just a display function and is NOT part of the way that the text is created, stored in the computer's memory, or printed out onto paper.

The cursor

The spot where you write your text is called the cursor position, and just about everything in the program is arranged around it. The cursor is always on the screen somewhere, either on the text page itself, or at some command position showing you where the keyed INPUTs will go. It is usually a flashing box, but will change into four horizontal strips whenever the 'next' character is to decide whether you will get some word wrap, justification or compaction automatically carried out on the text line which you have just finished writing on, i.e. you've just passed the right margin and "TLW" is waiting for the first character of the next line to decide how to handle it.

Allowable cursor movements

Imagine that your lines of text are strips of paper stuck onto a board. You can move the cursor box freely left and right, up and down all over the paper strips, but the screen space which is off the strips does **not** exist and you can never therefore get the cursor into this empty area. If you request such a movement the cursor will go up or down, to the nearest screen column which is legal. A subsequent similar move will remember the original column number and try to get back to it.

Note that the line and column data in the header refer to the text line in the file where the cursor is flashing and to the character column in that text line, NOT to the screen-display line (or column) on the current video-display page.

Script pitches

The program is supplied with the page script width and the right hand margin value set to 80 characters. Type in some text, anything will do at this stage, and see how it looks on the screen. When you get to the right margin just keep on going (don't touch the ENTER key at all yet) and the words will automatically be wrapped and justified into each text line. Now stop after a few lines and revise the script size like this —

— Press both shift keys together to change the word Write in the header to 'E' keys.
— Keep the SYMBOL SHIFT key down and press key 'V', and the word Video will appear in the black part of the header, together with a flashing cursor.
— You may now key in either 40, 48 or 60 for the new characters-per-line script setting instead of the 80 at present engaged.
— Finally, press ENTER and your page will be re-written with the new script size.

Look carefully at the result. The smaller you go, i.e. the higher the number of characters per line you choose, the more important it will be for your TV (or Monitor) to be properly tuned-in to your Spectrum. With a good TV picture the 80 column display will be quite easy to read, although you may find it better to write with 60 or 48 pitch and then view the final result in 80 pitch. The change from one pitch to another revises nothing in the text file itself, only the manner in which the information is presented to you upon the screen.

Left and right margins

You should have noticed by now that whatever you type is always formatted into an 80 characters wide result, from column 0 to column 80. This is a function of the left and right margin values set in the header, these two numbers have the dominant local effect upon your page format. Go into 'E' keys again and key 'M' to be offered a left margin revision or 'E' keys mode then hold down SYMBOL SHIFT and key 'M' for the right margin revision.

You can extend the right margin up to 148 or down to one more than the currently set left margin, and the left margin from 1 to the right margin value minus one. Nothing will have changed except the numbers in the header, but all script which you now create or reform will conform to the new margin format(s) — type some more lines to see the change.

Line end wrap-around

When a text line is longer than the set video-pitch can put on one screen line, the text will continue to be printed — without a break — onto the next screen line, and so-on until the line end control marker is reached, or the end of the text file. As an example, with a left margin of 5 and a right margin of 93 (for full-width 'A4' with Elite script in use) thirteen characters spill into the next screen line at video 80 setting, or thirty three at 60 setting, or forty five at 48 setting, or one whole line plus thirteen at the lowest setting of 40 characters per line. The screen line where the text line ends is shown by the spill or firm marker in the right margin, unless you have turned this feature off to clarify the page.

Line end control markers

Every line of text that you type in will get a special 'control' byte added to its end. This is there initially to tell "TLW" when to start a new line during video and print outputting work, then also to act as a 'marker' for the computer during paragraph reforming. The actual byte used in the text file will be either —

1. A 'spill' byte, shown by a curly-tailed down-left arrow in the screen's right margin, which is to indicate that your text naturally flows past the set right margin and onto the next text line. The byte value used is 14, and it is automatically added by "TLW" immediately following entry of the last character up to the currently set right margin in fresh text — i.e. you **cannot** key it in yourself.
2. A 'firm' line end byte, shown by a reversed 'c' in the right margin, which acts as a paragraph end and quite rigidly separates one line of text from the next by an important barrier. The byte used is 13, the ASCII code for 'carriage return', and this can **only** be generated by the ENTER key when you are creating fresh text. A line may even consist of just a single firm line end marker, when the [Col.] number in the header will be given as Ø — you can later expand such a 'zero-length' line using the Insert command.

The ENTER key

Although most of the keys produce a fairly obvious result when you press them, the ENTER key has a special significance. When you are typing fresh text, i.e. the cursor is leading at the 'front' of the text file, pressing ENTER will force a carriage return into the file at that position, firmly embedding a 'paragraph end' at that point and shifting the cursor to start a fresh line at the left margin on the next text line. This firm line end-marker is most important to the program's subsequent handling of the file.

If however you are somewhere within the text you have already written, pressing the ENTER key will just move the cursor down to the next line on the page, and will **not** alter any of the text which is there. The left margin is automatically fed to you at this point, but you can cursor-left toward the extreme left of the page if you want.

Inserting some 'extra' text

If in the middle of a line of text you want to add a word or two that you have missed-out, you will need to 'insert' the extra text between the other words. Text lines in "TLW" are — as it were — 'locked-in' by the line end markers after the line has been finished, so we need a way to cut the line in half at the correct point to do the job. The 'Insert' command will do this for you, and is used as follows —

1. Position the cursor on the letter (or space) just **before** the place where you want to add-in the new words.
2. Key the command 'E' keys 'I'. All the text to the right of the cursor will be blanked from the screen (it's copied to a 'buffer' elsewhere), and the cursor moved one position to the right into the 'free space' now available.
3. Type-in the 'extra' text you missed-out, with a space at the end if necessary.

4. Key 'E' keys 'I' again, and the temporarily blanked text will be added back onto the screen following the new words you have 'inserted'.

You can now issue a 'Reform' command (the next one described) to re-format the 'spoiled' line to fit properly between the left and right margins again.

Paragraph Reform

Now use the cursor-up control to get to the top line of your page, get into 'E' keys mode again and key SYMBOL SHIFT 'R'. This powerful 'Reform' command will consider your text briefly, and then clear the page from the line of the cursor and re-write it with the new margin settings as far as the **next** firm line-end marker — i.e. to the end of the paragraph. The 'Inset value' will be added to the left margin if the cursor is on the first line of the paragraph, to give it an indented appearance. Change the margins again, add some spaces by typing over what is already there, and do another Reform, once more the script will be neatly fitted to the new circumstances.

The text file — format and layout

"TLW"s text file is initially defined by the system variables at zero size, and it grows to contain only the characters which you put into it — plus of course the control bytes to mark where the text line and paragraph ends are. All the text which you key in and the way in which it is subsequently formatted is strictly controlled by the set left and right margins, and whether you have the Word-wrap/word-split and Justify/ragged functions on or off. There are **no** trailing spaces to fill your text lines out to the right margin. This aspect of "TLW" is fundamentally different to most other Spectrum text handling programs.

Text file size

The end of the space available for the text file is always at RAM address 49479, i.e. one byte below "TLW"s machine code, whilst the start of the file is at a moveable position in the Spectrum's memory at RAMTOP+2, i.e. the Basic CLEAR address plus two. RAMTOP is initially set in BASIC lines 30 and 40 at 30000, so the maximum file size is normally just over 19k bytes. You can reset RAMTOP either by direct command when the program is LOADed or by revising lines 30 and 40 and then SAVE/LOAD'ing it all again, for example CLEAR 48455 will give a text file size of exactly 1k (1024 bytes) and a space for BASIC of about 21k. You therefore can decide how you want to split the available space between that which is set aside for Basic and the space that is for the text file. If you throw away most of the Basic and just leave a RANDOMISE USR 52410 you can squeeze the text file up to around 25k — enough for about 4000 words or maybe 395 lines of average 80 column text. The currently available free text file space in kilobytes is given in the left box of the header next to the word 'Free' followed by a 'k', or — when this would be less than 1k — the figure is given in eighths of a k with an inverts mark, i.e. 3" means that 3x128 bytes are free to be written into.

Note! When you exit to Basic from "TLW" the text file start address is stored in the program for reference. At re-entry the current value of this address is compared with the previous value and, if they are the same, the existing text file will be displayed for you with the cursor at the top of the screen. If they are

different however, the text file will be ERASED and the program restarted in an EMPTY condition. CLEAR changes must therefore be carefully respected, they ALWAYS lead to the destruction of any resident text file material when the program is re-entered.

Keyboard buffering

Each keystroke that you type is initially lodged in a buffer in the computer's memory, and subsequently taken and used as soon as it can be handled. Normally the program is quite capable to keep up with matters in hand, but during justifying work, etc., the buffering operation has an important job to do in saving your keystrokes until they can be used. When there is more than one keystroke in the buffer waiting to be used the [Col.] number in the header left-hand box will be brightened to show that this is happening. The buffer can accept up to 21 printable characters in this way, or a single command. If you get further than 21 keystrokes ahead you will be shrilled at and nothing more accepted until the level drops back to below 21 again.

A 'tutorial' for you

We've written a lesson for you! It's really quite an extensive tutorial file called 'TEACHER', and will help you to see most of the "TLW" commands in a 'working' environment. It is saved on the tape right after the word processor . . . make sure the text file space has been completely cleared-out using 'E' keys SYMBOL SHIFT 'Z', LOAD the 'TEACHER' file in and then try the various video settings to see the way the script is presented on your screen. Besides being able to help you considerably in the task of 'getting-into' the program, the 'TEACHER' shows how easily the 80 column screen can be used to produce a first class full-width result. Experiment with it . . . there is absolutely NO substitute for practice in the process of learning how to really drive a word processor to the limit of its abilities, the more you play the more you'll learn.

Initial summary

It is the margins which principally control the final printed shape of your work, so make certain you get these as you want them throughout. Next in importance comes the Justify or Ragged setting, toggled by 'E' keys 'J', and this is responsible for whether your script is to be neatly lined up at the right side or ragged-edged as would normally be the case with type-written work. After this, well . . . there is a lot to learn, and really only one way to do it. Choose a simple task, like writing a short letter to a friend, and have a go. Don't worry yet about how it will get printed, simply spend some time at the keyboard and gain confidence with the controls. In time, try to get a little adventurous with the other commands to see how they affect the result. It is their scope which lifts the computer from being just a glorified typewriter into a much more powerful machine able to extend your own skills far beyond that of mere typing.

◆ ◆ ◆

Section C
The Commands in Detail

The entry of commands into "TLW" is by one of two approaches. The first covers the group of most often used commands, where a single shifted action (either with CAPS SHIFT or with SYMBOL SHIFT) is important so as not to interrupt the continuity of typing. The second provides the solution where more detail must follow the initial command INPUT to achieve the desired result, and involves the switching on of 'E' keys mode (extended mode) in the header display instead of the usual Write status.

The key-press descriptions given here relate generally to all Spectrum keyboards. On the Spectrum Plus/128/+2 and +3 and also many of the excellent keyboards obtainable from other companies, you will benefit from a number of extra keys whose action is to combine pairs of key-presses from the original 48k model. In most cases, holding the relevant keys pressed will repeat a command, i.e. this could be useful to centralise a complete block of text lines, etc.

In this manual we have arranged the commands into five main groups:—

1. Cursor movement — all the ways to move the cursor around the text on the screen.
2. Text handling — the commands which modify the text in one way or another.
3. Structure and utilities — these set-up the way the text-file is built and screened.
4. Printing — copying the stored text out to the ZX or line printer.
5. File storage and retrieval — data manipulation with tape/micro-drives/wafa/disc, etc.

Group 1 — CURSOR MOVEMENT

Right one character ..... CAPS/SHIFT '8'

The cursor moves one character right, and the line and column references in the left header-box are adjusted accordingly.

Left one character ..... CAPS/SHIFT '5'

Similar to the above, but the cursor goes to the left instead.

Up one line ..... CAPS/SHIFT '7'

The cursor moves to the same screen column in the text line immediately above. If this is not possible, the nearest available column position will be used.

Down one line ..... CAPS/SHIFT '6'

Same as the command above, but down instead of up.

Tabulate right ..... CAPS/SHIFT '1'

The cursor moves to the next multiple of the TAB value, or to the left margin if this is nearer. New space is created when the cursor is at the end of the text file.

Right one word ..... CAPS/SHIFT '4'
The cursor is moved right until a space is found, and then put on the next character.

Left one word ..... CAPS/SHIFT '3'
The cursor is moved back to the first character of the previous word to its left.

Right one sentence ..... SYM/SHIFT 'W'
The cursor moves right until a full-stop is found, then onto the next character.

Right to text line end ..... 'E' keys SYM/SHIFT '8'
The cursor moves to the last character or PCT at the end of the line of text.

Left to text line start ..... 'E' keys SYM/SHIFT '5'
The cursor is put upon the first character or PCT in the cursor'd line of text.

Up/left to screen top line ..... 'E' keys SYM/SHIFT '7'
The cursor moves to the character at the start of the top line of the screen.

Down/left to screen bottom line ..... 'E' keys SYM/SHIFT '6'
The cursor moves to the first character at the bottom line of the screen.

Down to top left of next screen, one link line ..... 'E' keys 'D'
The screen is refreshed with the original bottom line at the top of the screen.

Up to top left of previous screen, one link line ..... 'E' keys 'U'
The screen is refreshed with the original top line at the bottom of the screen.

To start of text-file ..... 'E' keys 'A'
The screen is refreshed with the cursor at the first position in the text-file.

To end of text-file ..... 'E' keys 'Z'
The screen is refreshed with the cursor at the last line of the text file.

To [input] number text line ..... 'E' keys 'N'
The screen is refreshed with the cursor at the start of the INPUT number text line.

Move the screen up by one line ..... 'E' keys SYM/SHIFT '3'
The whole 20 line video-display text-page is moved upwards by one line. Repeats continuously when held.

Move the screen down by one line ..... 'E' keys SYM/SHIFT '4'
The whole 20 line video-display text-page is moved downwards by one line. Repeats continuously when held.

Group 2 — TEXT HANDLING

Delete a character ..... CAPS/SHIFT '0'
The character or PCT (or line end marker if it's a zero-length line) at the cursor position is deleted and the page revised to suit. If a PCT's 'label' is displayed in the

header that PCT can be deleted without affecting any of the written text.

Delete text to end of line ..... CAPS/SHIFT '9'

This is a ZAP! command and requires a 'Y' for confirmation — or ENTER to cancel it. It deletes **all** the text from the cursor to the next spill or firm line end marker.

Delete text to end of file ..... 'E' keys SYM/SHIFT '9'

Another ZAP'ping command as above, but the text-file is terminated at the cursor position. See also the Insert command below.

Delete all text ..... 'E' keys SYM/SHIFT 'Z'

A third ZAP'per. Following a 'y' entry the whole visible text-file is cleared to zero size. See Insert below.

Insert/overwrite text format toggle ..... 'E' keys 'I'

When Insert mode is keyed, all the text from the character to the right of the cursor (to the text file end) is removed and stored safely elsewhere. The screen is revised accordingly, and the cursor is moved one place to the right, and now new text may be added here, or the cursor moved back through the file as usual. This smaller visible file will behave in all ways as though it is the complete and proper one, and all commands will only affect it and NOT the part 'hidden' away. ZAP'ping the file only clears the visible part, and LOAD'ing will as usual be onto the end of the file you can see. You can even 'hide' the WHOLE text file (except the first character), by giving the Insert command with the cursor at the text file start and work on another one brought-in from storage, etc., then print/save/delete this second file, and then still retrieve the first 'hidden' file intact.

When the command is re-used it toggles the header label back to O/write, and the 'hidden' text will all be replaced directly onto the end of the visible file. The cursor can be anywhere, and the screen will be revised if necessary. The Reform command may subsequently be used to repair any spoiled line formats which have resulted from any changes you have made during the Insert.

Push text left ..... SYM/SHIFT 'Q'

Text between the cursor and the line start can be pushed to the left as individual spaces are removed from existing groups and replaced next to the cursor.

Push text right ..... SYM/SHIFT 'E'

Text between the cursor and the line end marker can be moved to the right within the existing line length, as individual spaces from groups are replaced next to the cursor. This command has no effect if there are only single spaces between the cursor and the line end marker, or if the text file end is reached before any space pairs are found. It does NOT extend the line, it DOES re-shuffle the spaces within it. See Section K.

Centralise text in line ..... 'E' keys SYM/SHIFT 'H'

Provided it will fit, all of the text in the cursor'd line is compacted and re-arranged so that the result is centred between the set left and right margins.

Justify text line ..... 'E' keys SYM/SHIFT 'J'

The cursor'd text line is justified to fill the width between the current set margins. If there is no line end marker, i.e. the text file ends, the command will have no effect.

Limit spaces in text line ..... 'E' keys SYM/SHIFT 'L'

The cursor'd text line is compacted from the current left margin, i.e. all spaces are reduced to single spaces to occupy the minimum possible line length space. As above, there must be a following line end marker for the command to work.

Reform text from cursor to next firm marker ..... 'E' keys SYM/SHIFT 'R'

A very powerful command, which will frequently be used to repair blocks of text which have been hacked-about following their original writing. Text from the beginning of the cursor'd text line to the next firm line end marker is completely 'reformed', i.e. continuously Justified-or-Compacted, and the words wrapped-or-split as currently set, between the left and right margins. If a single word is too long to fit into the set line length, the reform will cease at that point and the text will be displayed with the cursor on the failed line.

Insert a Print Control Token ..... SYM/SHIFT 'I'

An INPUT is requested to define the required token, this keyed entry being detailed below. As a result, a single byte representing the token is embedded into the text file to the left of the cursor position, the 'label' describing the token is displayed briefly in the header, and a cursor-right made to continue. The INPUT may be one of —

1. A number from 1 to 24, when the token-associated codes for printer control will be issued during the print run.
2. A number from 90 to 99, when a token to make a print time call to the Basic line whose number is 100 x this value (i.e. 9000 to 9900) will be embedded and acted upon during the print run.
3. A letter from A to Z followed by a $ sign, when a token to fetch and insert there that particular Basic string at print time will be embedded. For more details on PCT's see Sections G and H.

Enter a 'Sinclair' Copyright sign ..... 'E' keys SYM/SHIFT 'K'

A Sinclair copyright code (CHR$ 127) is put into the text file. Note that this is a non-standard code, your printer may not recognise it. Useful for 'key-define' use?

Repeat a block of text ..... 'E' keys 'R'

INPUTs request the definition of a block of text by the numbers of the first and the last lines after which, provided the cursor is not in the block, it will be copied to start at the cursor line and the rest of the text shifted along accordingly. The screen page will then be refreshed with the cursor at the top-left.

Kill a block of text ..... 'E' keys 'K'

Again INPUTs request the line definition of a text block, following which it will be deleted from the file and the screen page refreshed as in Repeat above. There is NO confirmation with this command, once the second number has been ENTERed, the block will be permanently removed from the text-file. You must therefore be very careful in making certain of your first and second line numbers when using this command.

Find some text/optionally exchange it ..... 'E' keys 'X'

The first and last line numbers to search between are INPUT, the 'find' text to be sought keyed in (up to 16 letters) and either a replacement given — the

'substitute' text — or ENTER pressed if no substitute is to be offered, i.e. the command is to be used purely to 'find' something. The program then searches for the 'find' text and displays the line number for every occurrence located. Each time, you may substitute it with the second text string by keying 'S', by-pass and continue searching for the 'find' text by keying 'F', or you can Q'uit as usual.

The screen will be refreshed with the cursor at the top of the page, and on the line of the last 'find' if you Q'uit the search, or at the original cursor'd line if the search has reached the last line or the end of the text file — whether or not the search has actually been successful.

Group 3 — STRUCTURE and UTILITIES

Justify/Ragged text format toggle ..... 'E' keys 'J'

The command will swap the text-format between two modes. When Justify is set newly created text is justified between the set left and right margins (after a striped cursor has been displayed) or when a Reform command is given. When Ragged is set, text will be compacted from the left margin to give a ragged-right appearance, as would be the case with typewritten work. This paragraph is the ONLY one in this manual set with Ragged on — notice the uneven right margin difference.

Word wrap/word split text format toggle ..... 'E' keys 'W'

When 'W/Wrap' is set, no words will be split at a text line end — the program will automatically move them onto the next text line to ensure complete word integrity. However, when in 'W/split' mode there is no such intervention and all text will be simply cut up to fit between the set margins. Note that this process is NOT reversible! Also when 'W/split' is set an '^' is displayed next to the headers' Justify or Ragged as an extra indication that neither function will be active during text creation, but will still naturally work normally when the Reform command is used.

Case-set alpha characters to the line end ..... 'E' keys SYM/SHIFT '2'

Text from the cursor to the line end is reset per the current CAP's or low/c mode.

Include a Basic string ..... 'E' keys SYM/SHIFT 'I'

You can enter 'a$' to 'z$' to define a simple Basic string, which must already exist in the Basic variables area — and can be up to 4k long. The first eight string characters are displayed as a visual 'check' that you've got the right one, followed by a '#' symbol and the numeric string length. The command will NOT pass bytes outside the ASCII range of printable characters, i.e. they should all be between chr$ 32 (space) and chr$ 127 (the Sinclair copyright sign).

Outline on/off ..... 'E' keys 'O'

A toggle that simply switches on/off an outline to the video text page sides and top.

Visual markers on/off ..... 'E' keys 'V'

This toggle controls the automatic generation of margin markers and the character page/ink colour inversion next to PCT's. When the toggle is on, a firm

paragraph marker (a reversed 'c') is shown next to the 'Video' width number in the header.

CAPS/lower-case toggle ..... CAPS/SHIFT '2'
Flips the keyboard from CAP's (upper case) mode to low/c (lower case), and vice-versa.

Left margin revision ..... 'E' keys 'M'
Requests a numeric input, which can be from 1 to the right margin value −1.

Right margin revision ..... 'E' keys SYM/SHIFT 'M'
Requests a numeric INPUT, which can be from the left margin value +1 to 148.

Video-pitch revision ..... 'E' keys SYM/SHIFT 'V'
The input here can be 40, 48, 60 or 80, and the screen-display is refreshed with the characters-per-line value input, at approximately the same place in the text file.

Tabulation size and inset value revisions ..... 'E' keys SYM/SHIFT 'T'
The 'Tab by' value may be changed in the range 1 to the right margin value −1, up to 80, or ENTER leaves it unchanged. The paragraph 'inset' — i.e. the 'indent' to the first line, may then be revised in the range 0 to 8.

Screen colours revision ..... 'E' keys SYM/SHIFT 'B'
The command gives colour-access for the text page (0 to 7), the ink (0 to 7 — but add 64 to this number if you want to make the text page BRIGHT), the border (0 to 7), and finally the header (1 to 7). The revisions are permanent if the program is SAVE'd after they have been made.

Command summary 'help' page ..... 'E' keys 'H'
The text page is scrolled down to present a complete summary of the commands available and the key-press combinations required to call them. The relevant command modes are denoted by 'c', 's', 'e' or 'E' — followed by the correct key to press:—

Mode of entry:	**Key-press to make:**
Directly entered	Caps shifted 'key' — c 'key'
('Write' mode)	Symbol shifted 'key' — s 'key'
Indirectly entered	Extended mode 'key' — e 'key'
('E' keys mode)	Extended mode symbol shifted 'key' — E 'key'

Word count and cursor address ..... 'E' keys SYM/SHIFT 'W'
This command counts all of the groups of characters in the text file which are separated by spaces or line end markers (regardless of whether they are words or not . . . we regret "TLW" can't yet spell) and displays the result together with the current visible file length in bytes. Press a key and you will then be shown the cursor address in RAM, a useful feature if you want to analyse a text file from Basic.

Calculator operation ..... 'E' keys SYM/SHIFT 'C'
The text page is scrolled down to allow you to write in a mathematical expression, which will be evaluated when ENTER is pressed. A full description of

the calculator and all its functions is given in Section J of this manual.

Time setting ..... 'E' keys SYM/SHIFT 'X'

"TLW" runs an alarm clock which counts all 'free' waiting time. The alarm period may be set from 1 to 250 minutes (it's initially 45), or it may be set at zero minutes to switch it off. Using the command initially displays the set period and the remaining minutes to the alarm. You may then input a new alarm period, or Q'uit to leave the current values.

The alarm is silenced (and reset) by pressing the 'T' key. This feature is intended to assist you in the vital self-discipline which is necessary to re-SAVE all text files at regular intervals . . . Note that you will NOT be able to carry out any operation (apart from switching the alarm off) when it sounds.

Quit the current command ..... SYM/SHIFT 'Q'

This combination of keys is probably one of the most important — it will allow you to Q'uit most of the "TLW" commands at any stage without the result being binding.

Back to Basic ..... 'E' keys 'B'

This command transfers control of the computer directly back down to the main Basic program from whence the "TLW" m/c code was initially called. The menu will be displayed, and you are free to proceed as you require.

Group 4 — PRINTING

Print Guide ..... 'E' keys 'G'

A setting-up menu to drive the printer/interface as needed. Key 'R' (for Revise) and enter another 'R' if you wish to use the Sinclair ZX-printer, or set the 'output call', 'C/ret' and 'L/feed' bytes as required for line printing — see Section E.

Print-out some text ..... 'E' keys 'P'

A menu is presented which shows the first and last line numbers of the whole file, the current values lodged for 'copies' and line 'spacing', the 'pause' on/off status, and the current setting of the PCT's toggle. Section E explains more fully.

The Print Control Token file ..... 'E' keys 'T'

The 24 tokens are shown with their 'labels' (names) and their sets of four code bytes, together with a copy of the current standard PCT 'prefix'. See section G for a full explanation of this most important part of "TLW".

Group 5 — FILE STORAGE and RETRIEVAL

Save some text ..... 'E' keys 'S'

A menu requests the first and last line numbers to inclusively SAVE between, the choice of tape or micro-drive/wafa/disc unit etc., and the file-name to SAVE with. The SAVE will be done as soon as the details have been ENTER'ed. See Section D as well.

Load some text ..... 'E' keys 'L'

A menu requests whether tape or a mass storage unit is to be accessed, and the file name that is sought. The LOAD will add the new material to whatever is

already held in the computer's text file space — it will NOT replace it. Section D explains more fully.

Catalogue a cartridge/disc ..... 'E' keys 'C'
An input for the number of the mdv/disc unit to be CAT'alogued is requested, and the computer will present the data to you. The information remains on the screen until any key is pressed, then the page will be scrolled back or re-written.

Erase a file from a cartridge/disc ..... 'E' keys 'E'
The drive number and the name of the file to be ERASE'd are requested, and the job carried-out as instructed. With microdrives, if the file you have named is not present the message 'Absent!' is given and the scree held until ENTER is pressed.

Format a microdrive cartridge ..... 'E' keys 'F'
The drive number and the new cartridge name are requested, and the job carried-out as requested. With other types of storage units the command is inoperative. See Section O if it doesn't work.

◆ ◆ ◆

Section D
Loading and Saving Text

Once "TLW" has been set up onto your own mass-storage system (see Section P for full details), the commands for LOAD, SAVE, ERASE, CATALOGUE and FORMAT all use the same type of screen based menu and will handle both tape and your own system. LOAD'ing and SAVE'ing are explained here in detail, whilst Section C — Group-5 will give you the information required for the other three.

SAVE'ing a text file

This routine is called by 'E' keys 'S' and asks for four pieces of information —

1. The first text line to SAVE from. Give '1' for the file start, or you may put in a higher number if you want. The entry is validated when the next piece of data is in.
2. The last text line to SAVE to. This can be an over high number (i.e. 9999) if the file is to be SAVE'd to the end, or you can define the last of a block of lines within the file to be selectively SAVE'd. Both these first and last line numbers are checked for suitability after this second INPUT, and the length to be SAVE'd is given.
3. The drive unit to be SAVE'd to. This is alays Ø for tape, 1 to 8 for micro-drives, 1 for the '+3' disc drive, or 1 to 4 for other disc drives (for systems with drives A ... B ... you should use 1 for A, 2 for B etc.). The drive range displayed and maximum number acceptance can be limited by POKE'ing the system variables at 50470 and 64501 — see Section Q.

4. The file-name to be SAVE'd with. Most of the keyboard characters are available to SAVE with, press CAPS SHIFT for upper-case letters. Most systems allow a ten character name, if yours requires less then only the first eight or so will be used. Using 'a:', 'b:' or 'm:' as the first two characters of a Spectrum +3 file-name specifies the drive to be accessed — read the Spectrum manual for further details.

The SAVE will be carried out immediately following the last entry. If the file-name already exists on a micro-drive cartridge, the message 'Found! New? [y]' will be given, to which 'Y' will perform a complete ERASE/SAVE cycle and ENTER will drop you straight back to the text file. Other units will erase under these circumstances, except the '+3', which will naturally save a back-up copy with the file suffix '.bak'. Microdrive cartridge saving is in DATA format, whilst tape and discs are in CODE. Note that you can always LOAD some CODE from a micro-drive into "TLW" using option-2 on the menu — Pre-load text — if this is desirable instead.

LOAD'ing a text file

This routine is called by 'E' keys 'L' and just two pieces of data are required —

1. The number of the drive to LOAD from. This will be Ø for tape, 1 to 8 for micro-drives or the disc drive designation as appropriate.
2. The file-name which is sought, up to ten characters can be entered. For the '+3' the usual 'a:', 'b:' and 'm:' pre-fixes can be used.

The file will be fetched and LOAD'ed immediately following the second entry. All tape file-names are shown until the one sought is found and LOAD'ed. On cartridge any missing file-name is signalled as 'Absent!', whilst a found-but-corrupted file will error into Basic with 'File not found' as usual but all the sectors before the 'bad' one will have LOAD'ed successfully. If the cartridge file is too long for the space available, as much as is possible will be LOAD'ed before the error message 'space!' is given, i.e. the file space will be filled and the excess refused. Only CODE bytes will be accepted from tape and disc, and DATA from cartridge. If you have some CODE text on cartridge which you wish to get into "TLW" you can load it using menu option-2 instead, then re-save it as Data.

Text LOAD'ed into "TLW" won't overwrite anything already there, but is always added into the file space immediately following the existing text file end. In an empty file this means that the material is LOAD'ed in at the start of the memory space available, but if the program already contains a file the new script from outside will be added onto the end of what is already there — some other programs might call this MERGE. If the Insert command is in use so there is some text tucked away separately from the visible file, material LOAD'ed into the program will add to the end of the visible text file, allowing you to MERGE 'outside' text at any place within an existing file. You can thus add or MERGE into an existing text file any 'standard' pieces of text, provided they are already appropriately stored.

◆ ◆ ◆

Section E
Printing from the Text File

The 'print-out' function of a word-processing package is, in the final analysis, the most important part. It must allow you to select what to print, control the printer at all times exactly as you have programmed it, and handle the commands it meets during the print run so as to manipulate the connected hardware and put onto paper the results which you have directed it to produce. Think of the routine as a 'Program Controller'. The program material is the text file, comprising the printable text itself and the Print Control Token commands which you have embedded to direct the printing the way you want. These tokens can be either for direct Printer Control (tokens 1 to 24), for Including Basic strings (tokens a$ to z$), or for calling Basic lines in the 9000 to 9900 range (tokens 90 to 99), all whilst the print run is taking place.

The Print Guide

This is a setting-up routine to drive the printer/interface as needed. To obtain this use 'E' keys 'G', then key 'R' to get-in to revise something. Enter firstly another 'R' if you want to flip to the Sinclair ZX-printer, or set the 'output call' (for interfaces that normally require a 'RANDOMISE USR xyz' to initialise them before printing), carriage return 'C/ret' and line-feed 'L/feed' bytes as required for your interface and line printer. Use a 0 to put them back to a null [-] if necessary. Exit the routine with the usual Q'uit, or press 'P' to directly enter the print-out command.

The Print Routine

This routine is called by the command 'E' keys 'P', and the program will display for you the existing initial and final line numbers in the file, together with the last used set of 'Spacing', 'Copies', 'Pause' and 'PCT's' data lodged, so there is a chance that you can go straight to printing without any changes. If some revisions are necessary, press the 'R' (Revise) key to be let in to cycle through the data and input your changes or ENTER to by-pass etc. —

1. The 'from line' number (to start printing from) must be correct but the second 'to line' can be any higher-than-possible number if the last line is the one you wish to finish printing at.
2. The line-'Spacing' can be set to any number from 1 to 254, this simply feeds that many carriage-returns/line-feeds to the printer at the end of each text line.
3. The number of 'Copies' to be printed may be set to any number from 1 to 254, so that "TLW" cycles through the text file and prints the defined number of copies, without any further input.
4. The status of the 'pause' between copies may be either on or off. When on, the computer will wait for a key-press between each copy. Input an 'R' here to flip the toggle (on/off switch) to the required state.
5. The status of all the sixty Print Control Tokens can similarly be switched on and off by the same method as above when the cursor is on 'PCT's'.

When the above details are correct, press 'P' (print) to be offered the first printed copy, and then ENTER if the 'pause' is on. You can abort the output to the printer at any time with the SYMBOL SHIFT 'Q' (for Q'uit) control, or BREAK into BASIC if required.

On the Trojan cassette/disc of "TLW" you will find a text-file called 'P_TEST'. This is a simple test file which contains a number of Print Control Tokens to demonstrate many of the printer styles that are available from most dot-matrix printers. It will help you to load this file into your Spectrum and go through the 'print out' procedure with it, both to demonstrate the "TLW" command and to examine the result at your printer in conjunction with the screen display.

For further information about the PCT's, read Section G — Printer Control etc.

◆ ◆ ◆

Section F
Headers, Footers, Page Numbers

On some occasions you may wish to add a standard group of words to the top of your printed pages of text (a header) or to the foot of your pages (a footer), and of course also the current page number — starting at '1' or whatever is required. These features are 'built-in' to"TLW", and are reached through menu item-4. The 'extra' work this requires makes use of "TLW"s unique and flexible ability to access Basic whilst a text file is being printed — the program will fetch and insert the additional data at the appropriate places by intercepting each line-end (via a call to Basic in the 'line-feed' byte slot) so they can be counted to provide the result you are seeking.

Menu option 4 — Page numbers/headers/footers

The six pieces of data you must key-in to drive this feature are all held as Basic variables, so you are initially asked whether you want to examine 'existing data' at a [y/n] prompt. To set-up new data use the 'N' response, whilst to check some pre-existing material just key ENTER or key-in a 'Y'. The prompts and answers go as follows:

1. Whole page length [66]: . . . If you use 'American' 10" long paper this is the right number of lines, so just press ENTER — if not, input what you want (A4 is 70 etc.).
2. Print lines/page [59]: . . . This allows seven lines clear at the page base — again ENTER for this answer, or input what you really want (63 for A4 etc.).
3. First page number [1]: . . . ENTER gets you the '1', or input an alternative.
4. Header: . . . Here you should key-in the standard repetitive text that you require at the top-left of each page of print-out, or just key ENTER to leave this as a 'null' which will give no header at all. The length is not limited, but you must make it fit the circumstances to hand. It will be printed as 'line-1' of the output, with a blank line below.

5. Footer: . . . Same rules as for the header, but with one blank line above as it will be the last on the page.
6. Status 'off' — OK [y/n]: . . . Finally, there is a 'switch' that makes it all work (if 'on') or not (if 'off'). ENTER doesn't change the given status, 'N' swaps it to the 'other' condition.

In some of the prompts you will see that **one** of the answers is 'bright' — this is the one that just ENTER will provide, only a 'N' response will revise the existing state.

Following the last input, you can view the total state of play. If a change is required, the final 'Is this all OK [y/n]?' . . . allows you to ENTER to get the menu or 'N' to have another go.

One final and vital point. Above the six lines of entered data — below the bright line that says 'SCRIPT PRINT FORMAT DETAIL' — is the legend '(1st T/F print byte: PCT '9Ø')'. Other parts of this manual will explain in more detail, but in essence this is a reminder that the **first** printable character of a text file which is to use this page/headers/footers feature **must** be a '9Ø' Print Control Token. What this does is 'call' down to Basic line 9ØØØ when "TLW" is initially assessing the print aspects of the file, and it establishes the key start variables of the page/head/foot system. You can either put this at line-1 col.1 (followed by the rest of the text), or make the 1'st line contain **only** the PCT '9Ø' and start the 'real' text-file at line-2.

During 'simple' printing you will probably have set the Print Guide carriage-return to 13 and — depending on your system — the line-feed to 1Ø or [-], and your text files may also start with PCT's 23 and 24 to initialise the printer to the correct number of lines per page. When using this page/head/foot system however it is "TLW" itself that must determine the page-line arithmetic and **not** the printer — printer page-sizing commands must therefore be strictly avoided. Switching 'on' the status of the page/head/foot system will also install a PCT 91 to 'call' down to Basic line 91ØØ at the end of each line of text. If you normally use the line-feed with a '1Ø' etc., you will need to manage this another way — the easiest will be to put a PCT 92 as the carriage-return and add a line in Basic — **9200 LPRINT CHR$ 13; CHR$ 10;: GO TO wp** instead.

When you print your text file, the extra Basic computing will be achieved far faster than the printer can manage — and so the output will be quite continuous. The initialising PCT 9Ø will establish all that is required at the start, and the PCT 91 in the line-feed will do all the automatic line-counting and header/footer/page-number printing. The numeric variables used in Basic are viewable via the 'Calculator' — 'pgno' for page number etc. (have a look at the Basic), and you can use the 'Insert string' command to look at the stored header (h$) and footer (f$) data. At the end of the text file the Basic will feed blank lines until the foot of the final page has been reached, so that the final page number can be printed.

If you do any 'normal' printing after some headed/footed text in one session, be sure to go back down to the option-4 menu to turn the status 'off' and put back your normal line-feed byte — or a 'Ø' for [-] as necessary.

◆ ◆ ◆

Section G
Printer Control & Basic Calls

The Printer Control Tokens

The text bytes which are output by your computer to your printer for it to actually print onto paper are all in the range 32 to 127, i.e. the ASCII standard set of values which are universally used to represent the letters of the alphabet and the other keyboard characters. Most line-printers will however accept from the computer some special sequences of code bytes to revise specific features of their printed output format or style. "TLW" is well equipped in this respect with a bank of 24 extremely adaptable Printer Control Tokens, which can be embedded anywhere in the text file to cause a pre-set print-style change to occur during the print-run. The whole set of tokens can also be preceded by a single common byte which in "TLW" is called the 'prefix'.

The table of PC Tokens

"TLW"s PCT table is reached by 'E' keys 'T', and this displays for you the current format of the Printer Control Token file, the 24 associated 'labels' and the decimal values of the bytes lodged if between 0 and 254 — the value 255 is reserved as a null which is not sent to the printer but is displayed here as a '-'. The file looks like this in its 'as-purchased' form —

No:	Label:	Control tokens:				No:	Label:	Control tokens:			
1	Frm.F	—	12	—	—	2	Elit+	—	27	77	—
3	Pica+	—	27	80	—	4	Enlg+	—	27	87	49
5	Enlg−	—	27	87	48	6	Cond+	—	15	—	—
7	Cond−	—	18	—	—	8	Undl+	—	27	45	49
9	Undl−	—	27	45	48	10	Emph+	—	27	69	—
11	Emph−	—	27	70	—	12	D.st+	—	27	71	—
13	D.st−	—	27	72	—	14	Sup.+	—	27	83	48
15	Sub.+	—	27	83	49	16	Sp/b−	—	27	84	—
17	Ital+	—	27	52	—	18	Ital−	—	27	53	—
19	NLQ.	—	27	120	1	20	Ppr.−	—	27	56	—
21	Ppr.+	—	27	47	—	22	Initl	—	27	64	—
23	A4/70	—	27	67	70	24	Prf.6	—	27	78	6
	PCT prefix 27										

These labels and values suit the Kempston 'E' interface and the Epson standard range of printers, and will probably be similar to those which drive whichever printer/interface is attached to your own computer. All the labels and code bytes associated with them may be revised as follows to give a virtually unlimited range of results —

1. Input the number of the token you wish to modify (1 to 24), or 0 to gain access to the prefix which may be used to issue a standard byte ahead of each individual token byte to tell your printer/interface for instance that the next few

bytes are NOT text, but a print change command. (Note — the prefix will be a null '–' for the Spectrum +3).

2. Now input your revision, or just key ENTER to pass on to the next element. When you have finished, use the SYMBOL SHIFT 'Q' (Q'uit) command to scroll the PCT table (window) away and return you to the text file again.

Placing a token into the text file

A Control Token is entered into the text file by keying SYMBOL SHIFT 'I' and then inputting the token number from 1 to 24. The token will be embedded to the left of the visible character on the screen, the label or name associated with that token is then displayed briefly (in the black header section), and a cursor right automatically made. No screen-space is taken by the token, and your lines of text will always be the right length with the characters across the screen in the correct places so that you can judge the format. You can embed as many tokens as you like in one place, they will all be read and their associated code bytes output consecutively at print time if the PCT's toggle is switched on.

To show where the PC tokens are on the screen the character to the right is displayed inversely, i.e. it will have the ink and paper colours swapped. If this is not possible, i.e. when a token is at the very end of a text line and there are no characters to the right of it, the line end marker will be underlined to show that the line ends with a token. This character inverting feature can be turned off or on again by using the video-marker command 'E' keys 'V', which works like a toggle, and then refreshing the page to show the result.

Whenever the cursor is later placed onto a token, the label associated with it will again be displayed in the black header. You may if you wish delete just the token whenever the label is being displayed, without otherwise affecting the text.

The 24 Printer Control Tokens are never used with the ZX-printer, but the other two types of tokens — the 'Print-in strings' and the 'Call a line of Basic' are — provided the PCT's status toggle is on at the time of printing.

The token file may also be printed out to give you a permanent record — use option-6 on the KDF menu, itself reached via option-5 from the main "TLW" menu.

Calling Basic whilst printing a text file

You may want to use some lines of Basic to do some computing whilst the printer is working through the text file, for instance to revise some Basic strings that are later to be 'included' in the print-out (see Section H), to operate a page numbering/heading/footing system (already built-in to "TLW" when you purchase it), or even to install a replacement text file for 'chain-printing'. These ten PCT's are installed just like the 24 'direct' printer ones, by keying SYMBOL SHIFTED 'I' to get 'PC Token:' in the header and then inputting a number from 90 to 99 — the Basic line number which is 'called' will be 100 times the numeric value of the token used.

When the text file is being printed, "TLW" will output the text bytes to the printer until a PCT 90 - 99 is met. At this point the computer will jump directly to the line number required, where you should have installed some Basic to carry out the task you have in mind. At its simplest this can be a special printer token — for instance "TLW" has the Epson tokens for double-height and single-height

printing installed at lines 9800 and 9900, so PCT's 98 and 99 can be used specifically for this purpose in addition to the standard set of 24 printer tokens. Alternatively you could fetch-in some new string material from your storage system for 'inclusion' by an a$ to z$ token, or call your hardware for a screen-dump etc. — the options are almost endless.

The Basic lines that you use during these 'calls' can do almost anything as long as they don't compromise the security of "TLW" system software. All that is necessary is that they end with a **'GO TO wp'** statement, 'wp' being a Basic variable which is set to 9999 every time the word processor is operated. Whenever a 90 - 99 token is used by "TLW" during a print run, the system will conclude the print-out by completely refreshing the screen in case some TV-display corruption has occurred during the print cycle.

◆ ◆ ◆

Section H
Incorporating 'Outside Text'

There are two ways that you can include 'outside text' in "TLW". The first is for when you are typing into the file and you want a pre-set sequence of letters to be included as an integral part of the file itself, the date maybe or an often used name or even a paragraph of boiler-plate text. The second is strictly for use actually during the print run, and allows text in the form of Basic string material to be read-in and included as 'print-items' at the points in your text where Print-in tokens have been embedded.

How to Include a Basic string whilst typing

Provided that the necessary simple Basic string has been LET as a variable, key the 'Include' command 'E' keys SYMBOL SHIFT 'I' and you can then input the name of the string, a$ to z$, for assessment. You will be shown the first eight characters of the Basic string as a check, with a '#' plus the total string length.

You may then key 'U' to use the string, 'N' (Next) to do it again, or Q'uit as usual. If used, the string will be read into the text file as though being keyed in, and will be justified/word-wrapped etc. as normal. "TLW" will not allow you to read the string into a 'zero-line', you must therefore take care where you are going to do the Include.

How to Include a string whilst printing

Provided that some simple (i.e. not DIM'ensioned) object strings have been LET and exist as a Basic variables, there are 26 string printing tokens available to include the strings as print items. The tokens are called 'Print-in a$' to 'Print-in z$', and are embedded into the text file by using the PCT insert command SYMBOL SHIFT 'I' and giving the chosen string name letter and a '$' to follow. Note that the program makes NO allowance for the length of the string in relation to the printed space, so you must take care to ensure that it will fit as you require when it is printed.

A simple example using the "TLW" 'system date'

When you select option-1 from the main menu to go into the word processor, the final prompt is to revise the system date. As well as being POKE'd into the m/c code part of "TLW" so that it is 'permanent', it is also LET as a 20 character string z$ and is thus available to be included either whilst you are typing or during printing of your documents.

Whilst typing into the file therefore, keying 'E' keys SYMBOL SHIFT 'I' and then entering 'z$' followed by 'U' will read the system date into the text file for you. If however you initially key just SYMBOL SHIFT 'I' and enter 'z$' you will add a 'Print-in z$' token into the file, which will include the date only whilst the printer is operating. In the former case you can easily judge the space that the date takes because it is there to see on the screen, but of course then it is permanently 'fixed' and if the text file is printed on another day the date will be wrong. In the latter case however you must judge the 20 characters space that are required during printing, but now the date will always be correct (provided you set it on entry to the word processor) because it is freshly 'created' each time the file is printed. If you consider this feature in the light of the 'Mail Merge' and Data File options that are available in "TLW", you can see that it is easy to print repetitive personalised documents with 'merged' data — the date being an obvious candidate for inclusion.

◆ ◆ ◆

Section I
The 'Mail Merge' Process

The expression 'mail merge' is used in various differing ways to describe the business of 'including' in some controlled way selected items of data from 'somewhere else' in repetitive automatically produced documents. In "TLW" the availability of the powerful 'call a Basic line' and 'fetch/print-in a Basic string' tokens makes it possible to use the computer's data manipulation and logic assessment functions to extremely good effect in the production of printed documents incorporating (merging) specific stored data.

In Section L of this manual you will find a full description of the "TLW" Data File program, which allows you to construct and save special files which are directly compatible with "TLW"s mail merge system. Each file comprises eight separate records with five 24 character fields and a linked 8 character header which "TLW" uses as a comparison against user entered 'criteria' to determine whether or not the record should be 'merged' and printed with the current text file document. The facilities built-in to the "TLW" Basic system on option-3 will allow you to make full use of these data files from within the word processor, so that you may repetitively print documents which have merged into them only the specific data that you require.

The Mail Merge text-file 'document'

This is the "TLW" text file which is in effect the basic 'template' for operating the mail merge system. It must be carefully planned to achieve a consistently

formatted result bearing in mind the physical size of the data that will be fetched and included in the printed output. The inserted text or data can **not** be 'justified' into the text file document before it is incorporated into the print-output stream, and therefore you must construct your text so that any trailing spaces in the data fields are naturally acceptable in the finished print-out. The "TLW" Data File program provides records with five fields each of 24 characters standard length, this arrangement being normally quite suitable for names and addresses.

The 'practice' files provided on tape

The "TLW" cassette has three files which are provided simply to help you understand and practice with the mail merge system.

1. A very simple text file called 'DF__SHOW' which merely comprises lines each containing a single 'print-in a string' token, starting with v$ to show the 'header' from each record, then a$ to e$ in sequence. This file will print out just the contents of each record from the Data File program, one field per line, with the header at the top. Note that the string identities given here are not changeable to any others, they are 'fixed' for many reasons within various parts of the program.
2. A slightly more adventurous text file called 'DF__LTR', which is in effect a 'circular' that can be mailed to a range of people or companies once the name, address and system date (available as z$) have been merged by the tokens which are embedded.
3. To supply these two text files with some 'test' data, there is a file of data for 'merging' purposes from the "TLW" Data File program called 'DF__TEST'. This file is just a collection of eight names and addresses, for use in items 1 and 2 above.

When a file from the "TLW" Data File program is to be used as the source material for merging into a "TLW" document, you must initially use option-3 from the main menu and then enter 'd' to get a request for your data file to be fetched into the "TLW" Basic variables area. These files are 1024 bytes in length, and the program will raise the RAMTOP level accordingly to make sufficient room. This will necessarily lead to the destruction of any text file material in "TLW" when used for the first time, so beware. The filename and system input stream number are requested, then the data is LOAD'ed as asked. Since the data has to be 'sliced-up' into its individual 24 character fields when used by "TLW", there is some Basic at line 9500 which must **always** be engaged by a 'Basic call' token 95 at the start of the text file to carry out this work.

A practical worked example using the above test files

1. At the main menu select option 3 — 'Data File work'. A three element prompt will be given — 'Get data [d], D/prog [p] or C'teria [c]: . . .'. For this exercise, the 'd' answer is given, leading to a request for the data filename sought ('DF__TEST') and the source stream number (0 for tape etc.).
2. When the data has loaded, the 'Criteria' selection list is presented. Here you have the opportunity to establish a 'reference string' against which the 'header' from each of the eight records in the 'DF__TEST' file will be compared in order

to determine whether or not the data should be 'merged' by "TLW" during the print cycle. You have four choices — to use all the records without question [Ø], or to select [1] to [3] if you want to enter a subsequent 8 character string which will cause the data to be merged (and the document printed) only if the record header is 'less than', 'equal to' or 'greater than' the reference string you give. For the purposes of this exercise, enter a 'Ø' now to use 'all' of the data.

3. With the data 'installed' and a 'use-all' status established, the main menu is returned. Now enter '1' to go into the word processor and initialise the printer as you go, and from within "TLW" load the 'DF__SHOW' text file from tape (or your own system if you have transferred it there). Move the cursor through this file to see what it contains — the mandatory token-call to line 9500 at the start to 'set it up', then a series of 'print-a-string' tokens starting with v$ to give the record header followed by a$ to e$ to print the five fields in turn. This file will basically just 'show' you what is in the data file, in other words you can consider it as the 'print-out' function of the Data File program.
4. Turn on your printer, give the "TLW" 'Print' command, then key 'R' to move the cursor through the print data so you can re-set the copies to 8, and also the pause to 'off' unless you're using a single-sheet feeder in your printer.
5. A 'P' at the end will set the printer going, and the eight records will be printed out without a pause — with the header as the first line, then a blank line, then the five 24-character fields of data on consecutive lines.

Now you can have a good look at what is available in this 'test' data file. In spite of the rather trivial nature of the data, the underlying principle of the 'criteria selection' process can now readily be shown. Go down to Basic and change the criteria to a 'positive' value — say '< criteria 'P''. This will give you just five records, all starting with a 1st character of ASCII value less than 'P' in the Spectrum's character set. Try other criteria settings — the result is that YOU are now in a position to choose exactly what you want to select and print in your 'Mail merge' documents.

When you are reasonably happy with the above, clear the text file out and load in the 'DF__LTR' file. This is a complete 'circular' letter which requires to be fed with data from a file such as the 'DF__TEST' file already in Basic from your previous practise work. You will also need to set the system date for this, so key it in on the way into the word processor from the main menu. Try printing the letter with the same sort of criteria as you did for the 'DF__SHOW' file, and in very little time you will appreciate just how easy it is to selectively print a controlled series of documents with merged data. You are now 'Mail Merging' . . . QED!

◆ ◆ ◆

Section J
The Calculator

Whilst the program is being used to write a text file, The Last Word's calculator can be employed to carry out almost any sort of mathematical computation. It allows an approach to expression evaluation which is somewhat beyond the

abilities of a normal calculator, the style being as though the input is being handled directly in Basic by the Spectrum's VAL'uing function. The computer's logical and scientific functions can be readily employed, and the Basic variables area is freely available to you, from which you may pick numeric or string variables by name. The results of your sums can be retained as Basic numeric variables with the names m0 to m9, as though the calculator has ten reserved memories to store them for you, and the 'current' result can be written into the text file at the cursor position.

The calculator is called by the 'E' keys SYMBOL SHIFT 'C' command, and you can now write into the bright window any logical expression and then press the ENTER key — an '=' is not required. Your expression can range from just a number, to which the result will be the same number (!), to a simple sum such as 4×3+8×5 — and be careful of the computer's usual 'operator' priorities, to any collection of numbers/functions/logical operators/variable names/sliced string valuations — which you'd normally expect to be resolved by the computer's Basic expression evaluating system. If you have a string expression established in Basic such as LET c$="m1+125.5×(INT(m2/m3)×(1+vat))", you may expect VAL c$ to be calculated either when given on its own, or when included in any other expression provided all of the included variables m1, m2, m3 and vat etc. exist in Basic.

Whilst using the calculator, the keyboard decoding is modified to make available the Sinclair logical and scientific functions by using the SYMBOL SHIFT or CAPS SHIFT keys. The range of calculator functions which are available are listed below —

Function	Key	Shift	Function	Key	Shift
+	K	SYMBOL SHIFT	LN	Z	CAPS SHIFT
−	J	SYMBOL SHIFT	VAL	J	CAPS SHIFT
×	B	SYMBOL SHIFT	SGN	F	CAPS SHIFT
/	V	SYMBOL SHIFT	ABS	G	CAPS SHIFT
^	H	SYMBOL SHIFT	PI	M	CAPS SHIFT
SIN	Q	CAPS SHIFT	<	R	SYMBOL SHIFT
COS	W	CAPS SHIFT	=	L	SYMBOL SHIFT
TAN	E	CAPS SHIFT	<>	W	SYMBOL SHIFT
ASN	S	CAPS SHIFT	>	T	SYMBOL SHIFT
ACS	C	CAPS SHIFT	>=	E	SYMBOL SHIFT
ATN	T	CAPS SHIFT	OR	U	CAPS SHIFT
EXP	X	CAPS SHIFT	AND	Y	CAPS SHIFT
SQR	H	CAPS SHIFT	NOT	S	CAPS SHIFT
INT	R	CAPS SHIFT			

Handling the Result

When a result is given, you can store it into a memory by keying 0 to 9 as required. Any other key will by-pass this, so that you can 'Use' the result (key 'U' and it will be written into your text), or 'N' for next (will discard it and re-present the calculator for you to use again), or SYMBOL SHIFT 'Q' for Q'uit will return you to the text file. Note that you can for instance use the calculator to have a look at the current value of the Basic variables used in the page numbering and other routines — the lines/page is 'lppg', the current line-count is 'lcnt', the page number to print 'pgnop' etc.

If a result cannot be given, i.e. the calculation 'fails' and would normally produce an error report in Basic, you will be asked [what?], and the statement you have written will be held until you press a key to initiate another go.

◆ ◆ ◆

Section K
Hints and Tips

Extending/adding-to an existing text-line

If you have a line of text 'locked-in' by line end markers, the 'push text right' command wil **only** shuffle text through the spaces to the right of the cursor, it will not actually extend the length of the line itself to accommodate **more** text (i.e. it won't add-in spaces etc.). If you want to get some more into the line therefore, you can —

1. Extend the space around the cursor using the left/right pushers. This is necessarily limited in capability to the extent of 'free' spaces (ASCII 32) you have in the line to shuffle around. See also 'Margins' below.
2. Artificially expand the line to the length given by the right margin value — using the single-line justify command 'E' keys SYMBOL SHIFT 'J'. This will 'give you' some more space to play with the pushers. Afterwards you can tidy the line using the single-line space limiter 'E' keys SYMBOL SHIFT 'L'.
3. (2) above can be greatly improved by temporarily extending the right margin to 20 or so more than the 'current' setting, and popping it back aftwards.
4. Use the 'Insert' command. This temporarily slices the text file from the character after the cursor, and gives you all the room you could possibly need to add-in further text at any place. After the extra text is inserted — toggle the 'Insert' again, use the 'Reform' to re-paragraph the extended text, and off you go . . .

Addresses etc. at the 'top right' in letters

The 'word-wrap' feature can make these very difficult to do! Set 'W/split' and a right-margin one higher than you really want. Then TAB each line to the end and key ENTER to force a firm line-end at the correct margin. Do this for as many lines as you need to build a 'bank' of space, then return the right margin to the correct value. Now position the cursor for the start of the address and key it in. Your text will stay exactly where it is put, and you can use the push text left/right commands to get it all lined up as you want. Finally, switch back to 'W/wrap' again to continue.

Margins

Even if you want your final printed result to appear to be from column 1 etc., set the left margin to some small but positive value, e.g. 5, and adjust the paper in your printer accordingly. This will allow you to use the push text left command to create spaces in the line when your typing has lost a character or two, because there will

always be a store of 5 'spare' spaces to push the text left into when necessary. If even more room is required in a line, temporarily set the right margin to a much higher value and Justify that line to create more spaces within the line, and then use the text left/right pushers to clear an area for what you want, reset the margins, and lastly Reform to clear-up the disarray. The Insert command will also allow you to add-in anything you want, but if it's already in use the above approach is rather more helpful.

Video Clarity

In video 80 and even 60 pitch display, when the video markers are turned on, it can sometimes be difficult to interpret individual letters when they follow PCT's embedded in the text. Temporarily, turn the video markers off and — to show the characters more clearly — move the cursor back and forwards through the unclear spot. Then re-toggle the markers back on again and ENTER to the next line to leave the area clean.

Pause the printer

If you want to stop the printer for a daisy-wheel or a paper change etc., you can do this easily at any given place by setting-up for example a PCT 94 to call a Basic line —

9400 INPUT "Pausing! ENTER continues ..."; LINE p$: GO TO wp

so you can carry out the changes you want, then press the ENTER key to continue.

Cursor to a unique word in the text file

Use the 'Exchange' command to find the particular word or set of characters in the file, then Q'uit at the find to refresh the page with the cursor on that line.

Underlining with a daisy-wheel printer

Some types of printer can only underline words by going back over them and using the ASCII '_' under-bar character 95. In such cases you must usually specifically program the result that you require, probably by issuing a carriage-return immediately after the words to print the line thus far (**not** with a line feed!) and going back to do the underline job. This you can program into "TLW" by using a Basic line call — for example PCT 93 at the end of the words to be treated, backed up by a Basic line in the form —

9300 LPRINT CHR$ 13; TAB 50; LPRINT "_____";:GO TO wp

with the 'TAB 50' specifically set to suit the circumstances to hand.

Long and grouped PC tokens

Some printer control sequences may be longer than the four bytes (or five with the prefix) that the PCT table will hold. The trick is to use a string print-in PCT (in this case use t$), and to LET a string in Basic as required, i.e. for a ZX-LPRINT III interface and a Citizen 120D printer, the 'Master Graphics Command' would be —

LET t$ = CHR$ 1 + CHR$ 5 + CHR$ 27 + CHR$ 42 + CHR$ 0 + CHR$ n1 + CHR$ n2

which totals seven bytes, but is called by one token only.

Note also that the tokens for double-height and single-height printing are built into the "TLW" Basic as PCT's 98 and 99 — calls to lines 9800 and 9900 respectively. The double-height fully compensates for the correct line feed count in the page heading/footing/numbering.

Refresh the screen

If you do manage to outwit the "TLW" screen-handling code and get the cursor stuck where it really shouldn't be (pretty unlikely, but it's all rather complicated), do a 'cursor to top-left' and use that line number to re-page with, i.e. 'E' keys 'N' and input the line number in the header box, then move back to where you were to continue.

SAVE and [space!]

If you have a text file full of priceless material and you get the [space!] message when you try to use a microdrive, this is because there is insufficient room to create the required 595 byte microdrive channel, so "TLW" is preventing a potential lock-up by disallowing the command. To make more space available, go into Basic and clear out all of the variables (RUN 3160 will do this) and try again.

An alternative method is to put the cursor at the text file beginning and use the Insert command to 'hide' all the text away, revise the RAMTOP address up a bit from Basic by issuing a higher value CLEAR statement than was previously given, then get back into "TLW" and re-type the first text file character and de-insert to retrieve all of your 'hidden' text. The former method is preferable, but the latter may get you out of trouble of a more serious kind — if there is enough space.

File-chaining whilst printing

Since the m/c code calls are available from Basic to 'zap' the text file space clear (USR 64978) and to reset the file pointers after installing a new text file (USR 65010) it is not difficult to use a Basic line call PCT to replace the existing text file and go straight into printing another — whilst the printer is operating. For instance —

```
9700 LET s=USR 64978: LOAD "m";3;"new-file"CODE s
9710 RANDOMISE USR 65010: GO TO wp
```

will — if you put a PCT 97 on a separate line at the end of the 1'st text file and then print it — print out the initial file, drop to Basic and 'zap' the file space and return the file base address in the variable 's', load the file "new-file" into the text file space from drive 3, re-set all the file pointers to the 'start to print it all' condition, and finally the 'GO TO wp' statement continues the print process at the 1'st byte of the new file.

Operating problems

A great deal of care and attention has been taken in checking and mastering the current issue of "TLW", and it is very unlikely that any problems will be the result of a 'rogue' cassette. If however you repeatedly experience a problem it is most likely that it is somewhere in the 'personalised' code, so be sure that the problem has **not** occurred during the adapting process. The only certain way to be sure that you are working with good original code is to start again from scratch with the

original tape. If that does not solve your problem then one of two situations probably exists —

1. You have either misunderstood the correct way in which to tackle the task you are attempting, or you are getting the right result according to "TLW" but it is not what you expect. Read the appropriate manual text again to make certain of your approach.
2. You have indeed got a bad cassette or disc. Try the other side of a tape as a check — this is also recorded with the same material but at a different setting, and is most unlikely to exhibit the same problem. If this doesn't resolve things, ask your dealer for a replacement or send it back to us with full details of where you got it etc. and a replacement will be sent to you.

Other problems ...

If you are stumped by some problem over which you would like some help, write to us with a clear description of the situation and PLEASE!! enclose a stamped/addressed envelope so that we can reply. We don't guarantee a solution, but we'll certainly give it some thought and then let you know the result.

◆ ◆ ◆

Section L
The "TLW" Data File Program

The "TLW" suite of software includes a Data File Program allowing you to construct and save a series of 'files' which each comprise eight 'records' carrying five 24 character 'fields' of any sort of mixed numeric and/or character data, each record also being linked to an 8 character 'header' (i.e. 8 headers per file) which can be utilised by "TLW" through a user-driven 'criteria-comparison' procedure to provide you with selective inclusion of the data in mail-merged documents.

Entry to the Data File Program is obtained through menu item-3 in "TLW", then giving 'p' to use the program and '0' to get it from tape etc. The program is driven entirely through input prompts at the foot of the screen, the paragraphs below describing the procedure to adopt to either review an 'existing' file or create a fresh one to your own requirements. The "TLW" cassette carries an example file called 'DF_TEST', which it will help you to load into the Data File program by entering 'r' for review / 's' for stored / 'DF_TEST' for the filename / '0' to specify tape as the source (if not using a +3), and then running your cassette until the file is loaded so you can view the program in an operational context. There are also two "TLW" text files to help you to practice in the word processor with this data file, one called 'DF_SHOW' to simply print the data file without any further manipulation, and a second called 'DF_LTR' with which you can make more serious experiments to see just how the whole 'Mail Merge' business works.

When initially loaded the program asks you to 'review' an existing file or 'create' a new one. We shall here consider the new file creation process first, then the review of it after it has been stored.

Creating a 'new' data file

When the 'Create [c] Review [r] "TLW" [t]: . . . ' prompt is given, answer by entering a 'c' and the program will 'dimension' a fresh 1Ø24 byte data file for you to fill with the information that you wish to use in "TLW". Eight records will be presented to you in turn, for you to make an entry into firstly the 8-character 'header' and then into the five 24 character fields of each record. You may enter a null field simply by pressing the enter key, or data as appropriate for your needs. Be careful not to invoke any of the Sinclair Basic key-words, as these will **not** be printed properly by "TLW". You can use the fields for any purpose, for instance for names and addresses the 1'st line will be the name and the subsequent 4 for the address. The header is really only there to act as a 'match' with the criteria you later use in "TLW" to judge whether to 'merge' it or not, and need not be filled unless you wish to use this feature. Alternatively you can employ it to make a broad descriptor for the data being held etc. — for instance, put an 'A' in it and then put your 'A. . . ' names and addresses on file for simple reference when required . . . by "TLW" or not.

After each record has been finished, you are asked whether you wish to review the file as it is — in which case the 'review' procedure is operated as discussed in the next section — or just pressing enter will get to a prompt allowing you to save the file to tape, microdrive or — if you have a '+3' or have revised the Basic as given at the end of this document — to your disc system. After this the program goes back to the initial 'review' or 'create' prompt, from which you can now of course select 'review' to peruse the data that you have just keyed-in — it's 'e' for 'existing' etc.

Reviewing an existing or stored data file

At the initial 'Create [c] Review [r] "TLW" [t]: . . . ' prompt, select the 'r' option to have a look at some existing data in the program — or to fetch another file from your storage system. The next prompt is 'Existing [e] or stored [s]: . . . ', to which you answer as required. Whilst 'e' gets you straight to the 'review' stage, the 's' response leads to the usual requests for a filename to load and the source stream to access.

When the file is installed, the first record will be displayed. You may now 'step through' the file, either keying ENTER to pass-by a record, inputting a 'p' to move back to the previous record or 'e' to skip to the end of the sequence. Entering a 'c' naturally leads to input prompts allowing you to enter a new 8 character 'header', then in turn each of the five 24 character fields is replaceable by a new entry. If your entry is too long it will be refused and re-prompted. After the eighth record the 'Save the file [y/n]: . . . ' prompt is seen again, and the system has rejoined the sequence at the conclusion of the 'create-a-file' stage.

To return to the "TLW" word processor after some work in the Data File program, simply answer 't' to the 'Create [c] Review [r] "TLW" [t]: . . . ' prompt and give the stream number required for the load — and back you go.

◆ ◆ ◆

Section M
The 'Key-Define' Facility

What 'Key-Define' means

The purpose of this fairly complex 'extension' to the "TLW" word processor is to give you access to two of the fundamental aspects of the program —

1. To re-define the links between some of the keys on the keyboard and the pattern of dots that make up the associated character on the screen, so you can substitute your own character shapes for selected letters and alternative byte values for printing.
2. To provide some extra sets of printer control tokens that can be sent to your printer before and after the re-defined characters, so that 'alternate' character sets may be switched-on each time to produce a printed result to match that of the screen image, and switched-out immediately after.

The 'key-define' program thus allows you to define two lists of any eight keys with entirely new 'characters' on the screen, and passes them through an extended print-output subroutine so that — as long as your printer can provide the alternative fonts — it will switch them in and employ them appropriately for each individual character. As a very simple example for instance this will allow you to use the USA set in your printer (set Ø in most Epson style printers) for the majority of the time — and thus make available the '$' in the usual way — and to automatically switch to the English set (set 2) for individual use of the pound-sign when desired instead (the pound-sign is further discussed as a specific 'font re-definition' item in option 4 below). The 'KDF Basic' program includes eleven stored sets of eight international screen characters — and a graphics set — plus a complete 'font designer' option so that you can 'do your own thing' in all the four character pitches which "TLW" screens. You may thus re-define up to sixteen keys as eight foreign characters plus eight graphics, or two sets of eight international or graphics characters, or sixteen graphics characters, etc.

After the 'key-define' program has been used and the 'modified' version of "TLW" saved for future use, the status of the two banks of re-defined keys is shown by the number appended after 'keys/' at the end of the message on "TLW"s menu option-1. If neither set of re-defined keys is currently 'active' the menu will say 'keys/Ø', for set-1 'active' there will be 'keys/1', for set-2 similarly 'keys/2', and when both lists of keys are 'active' it will bear the legend 'keys/3'. You can revise this setting at the first prompt after you have entered '1' to go into the word processor, or simply press ENTER to leave the setting unchanged. Whilst using the word processor, active re-defined characters are screened only in their 'new' form on the text page of the program and not in the header or any of the command bright patches as this could quite possibly render them unreadable.

The sequence of events per character screened/printed

When you press a key, "TLW" buffers it at once and — at the first opportunity — fetches it from the buffer for use. After it has been put into the text file **unchanged**,

a check is made to see whether it is in either of the 'active' lists. If not the standard character font is employed. If it **is** an 'active' re-defined character, however, the alternative 'KDF' character font data and the correct pitch micro-print mask is employed to write the 'new' shape onto the screen.

When the text file is being printed, every text file byte is similarly checked to establish whether it requires special handling. If not, the printer receives it unchanged. If on one of the 'active' lists, however, four actions are invoked —

1. A further 'Translation' list is checked to see whether you want the byte swapped for a different one. This is to allow for instance the '£' key (non-standard Spectrum byte value 96) to be swapped for byte value 35 which is the '#' on a Spectrum but the '£' in ASCII — and it's ASCII that your printer will want to 'speak'.
2. Depending upon which of the two lists that the original text file byte to be printed has matched with, an additional PCT is provided by the 'key-define' code (up to 4 bytes, preceded by the standard 'prefix' as usual) whose purpose is to 'switch' the printer to the alternative appropriate character set.
3. The byte to be printed is now sent.
4. A second additional PCT is fetched from the 'key-define' code and sent to the printer to change the operative character set back from the 'alternative' one to the original set that was in use at the start.

The "TLW" printer-reader routine then returns to carry on with the next character to print . . . This may sound to you like a pretty extensive operation, but of course in machine code it represents a very standard transposition manoeuvre which will make no discernable difference to the speed at which your printer churns out the result.

We now assume that you have used the "TLW" menu option-5 to load the 'KDF Basic' program into your Spectrum 'underneath' "TLW" itself. The 'KDF' menu is quite different from that of "TLW", this section is based on the former — **not** the latter.

Option 1 — Use "The Last Word" — keys/0

This option initially offers a revision of the KDF status — '0' indicates both lists are off, '1' or '2' shows that list only is active, whilst '3' engages both on. For no change just press ENTER, after which you will pass straight up into the word processor. Note that in this situation you will have a text file space of about 1490 bytes — just enough to test it and your printer.

Option 2 — Load a character set from the data bank

Twelve options are provided, the usual eleven 'International' character sets provided by most Epson type dot-matrix printers, and a set of graphics characters which are our personal choice of eight from the 32 available as chr$ 128 to 159. Enter the set number, then the list (1 or 2) you want to use. The four extra Printer Control Tokens (see item 5) are arranged to suit list-1 for 'foreign' characters and list-2 for graphics, but you can change them. The character set will be copied to the chosen list, and the main menu returned.

Testing — the KDF test-file

Whilst in "TLW" you can load into the word processor a short test text file called 'K_TEST' which shows both standard lists on one 'page'. Use it constantly to display the current state of the KDF system, it will quickly clarify the screen appearance and the printer handling of the current key-material being defined. The '40' video setting shows this at its clearest — but of course the other pitches are of equal interest. Note that for the graphics bytes to be passed to the printer the interface must be set to handle bytes above 127 and below 32 without changing them into Sinclair word-tokens, this may require a different approach to that which you have set up for "TLW" and — if you do not normally use it — the line-feed byte may need to be invoked as well.

Option 3 — Display one character — four pitches

This item shows you one character at eight-times screen size in its fundamental 'character-set' form as well as in each of the four pitches — 40, 48, 60 and 80 — that micro-print runs in "The Last Word". All the individual byte values are shown, and the micro-print character masks which 'slice-out' selected bit columns from each letter. It vividly shows the speed penalty of Basic compared to code . . . please be patient!

Option 4 — View/change character font bytes

The list (1 or 2) you want to review is requested, then whether you want to see the 40/48 or 60/80 pitch data — they are separately handled from different source information by micro-print. All 8 bytes from the eight characters on the list chosen are displayed (although please note that the top byte is not used) together with the micro-print mask for each one. Enter 1-8 to access/revise a characters' font bytes, it is their binary values that are the bit-pattern you see on the screen as dot-lines in each 'letter'.

For instance — the byte whose decimal value is 179 is written in binary as 10110011, the 1's representing powers of 2 starting with 2 to the power of zero at the rightmost bit (**bi**nary-digi**t**), then to the power of 1, then power 2 and so on up to the power of 7 at the leftmost bit. From left to right the bits have individual values therefore of 128, 64, 32, 16, 8, 4, 2 and 1. Byte '179' is made-up of 128+32+16+2+1, hence the binary form 10110011. The 1's show on the screen as the black dots and the 0's as clear space — thus the 8 seemingly random decimal byte values in each character's font. The process is best appreciated by using item-3 in conjunction with this one. Note also that the micro-print masks need three set bits (i.e. the 1's) for 40/48 fonts, five for 60/80, and the program will refuse wrong entries. Screen characters should be thought of as being built-up from right to left, the mask for the character having set bits where a column of bits from the 'original' character form is to be 'missed'. The rightmost two mask bits are used in 6-bit wide 40 pitch letters, and the rightmost four mask bits in 4-bit wide 60 pitch.

As a practical worked example — suppose that you want a proper answer to the perennial hash/pound sign problems, where for most of the time you need the '#' for Basic listings etc., but it's on chr$ 35 in the USA character set — whilst the pound sign is the same code but in the English set instead, i.e. no **one** set has them **both**. In short, what is required is the ability to run primarily in the USA set

but to divert to the English set to print sterling values when necessary. In the Spectrum the pound sign is fortunately chr$ 96 — normally designated as the ASCII reversed apostrophe — so we need to print the pound on the screen with the usual key and put it into text files as chr$ 96 but, when printing is called, use KDF to substitute chr$ 35 for it and swap momentarily to the English set to achieve the desired result.

The way to do this is to nominate List-1 as the English set using item 1, and use item 5 to revise the first character to chr$ 96 by selecting '1' and entering the '£' there, then ENTER'ing through the rest of the list. Still in item 5, enter '3' to use the translation list and there install chr$ 96 to be translated to chr$ 35 on the top line. All that remains now is to create a font for the 'new' chr$ 96, since whenever the '£' key is pressed with [keys/1] engaged the font for the character will be drawn from within the KDF extra code rather than from the "TLW"s normal character bank. This you can achieve by going into item 4 twice — once for 40/48 pitch and once for 60/80 — and keying in the following font bytes: 28, 34, 120, 120, 32, 126 and 0. For 40/48 pitch the mask byte should be 73, and for 60/80 it is 211. When these are in, use item 3 to have a look at the screen appearance of all four pitches of the new chr$ 96, nearly the same as the Sinclair pound but with a thicker centre-cross bar to identify it. Finally you should save it all back onto your own system for future use. And there you are . . .

The '£' sign as given here is in fact incorporated in "TLW" as a standard feature, but if you would rather do something else — using the stored character fonts etc. — you have only to reset the 1'st byte of list-1 to suit.

Option 5 — Lists/print translations/Print Control Tokens

The screen displays the two lists of eight characters which are currently assigned to the KDF system, a print character translation list, and the two pairs of four-byte PCT's which are issued following the existing "TLW" prefix byte (which you can revise from inside "TLW") immediately before and after a 'listed' character. Enter 1 or 2 to peruse/revise the lists, 3 to go through the print translation list — note that this one works **after** the other internal operations of "TLW" but is not caught out by stray matching bytes from the other 24 PCT's — and 4 to 7 to access the new PCT's. If you want to 'null' a list-1 or 2 character use a space — chr$ 32, in the translation list go **from** chr$ 128 **to** chr$ 128, and '255' produces the usual '–' byte in Print Control Tokens.

Option 6 — PCT table and Help screen print-outs, byte swaps

Two simple routines are provided so you can (1) print out the "TLW" table of Print Control Tokens and (2) the Help page. Your printer will probably require that you initialise it first — use option-1 from the KDF menu or set it up in "TLW" before you go to the KDF program.

Four simple character swap-lists are provided in (3) so that straightforward byte substitutions can be made within the "TLW" printer-reader and before the more serious KDF interceptions are put into action. The item screens two lists — the first giving eight opportunities to swap screen letters for designated alternatives, and the second doing the same thing to send to your printer. The swaps will be made *prior* to the KDF translations, and are suitable for simple transpositions on a 'one-to-one' basis.

Option 7 — Load/re-save the "TLW" code

If you load the KDF Basic program away from "TLW", you are here able to load the "TLW" m/c code so that the KDF routines can be used on it.

When you have used the KDF Basic program to revise some detail of one or both of the two 'translation lists', you will want to re-save the "TLW" machine code in its new arrangement so that whenever you use it again the KDF set-up will come back as it is now. Simply enter 'y' in answer to the 're-save' prompt, or ENTER to pass by back to the menu. The program gives two places where you can add lines to LOAD and SAVE the code with systems other than tape or microdrives, the screen will prompt you appropriately. Note that it is saved with the text file base at 48002, and for successful "TLW" menu-2 operation you will need to re-enter and re-save it by RUN 80 in "TLW".

Option 0 — Back to "TLW", stop in Basic

A 'y' answer to the prompt will re-start you back in "TLW", or ENTER by to be allowed to stop in the Basic program. This will allow you to make whatever changes you want: to save it all back to cassette for instance RUN 9998, or add your own system 'SAVE' code at 9999 (KDOS disc lines are installed as standard). Note that if you have a Wafa system the extra 2k space required to run them just cannot be accommodated with the KDF Basic in your Spectrum as there simply is not enough room — you must do it all from cassette and then transfer the new extended "TLW" to your Wafa via a temporary tape copy.

◆ ◆ ◆

Section N
Text Files from Other WP's

You may have some text files from other word processing programs saved on tape, microdrive or even disc that you would like to use in "TLW". This is possible provided that the file format is fully compatible, but there is usually a problem. Many other WP's make files that have no carriage returns embedded, but simply rely on their lines being a constant 64 (or whatever) bytes long and the WP's print output reader 'insert's the necessary c/r's where required. This makes for lots of blank 64 character lines, whereas "TLW" creates exactly what is required in an efficient 'collapsed' style but with the two different types of carriage return — spill and firm — already installed in the correct places. You need therefore a special program to 'stretch' your other WP files to 'include' the additional c/r bytes at the required intervals, after which you can 'tidy them up' in "TLW" to achieve the more elegant 'collapsed' format.

The 'TLW/ADAPT' program

In the Spectrum +3 version of "TLW" this 'adaptor' program is reached by answering 'y' to the prompt following the selection of menu item 0. For all other Spectrums this 'stand-alone' program is at the end of your "TLW" cassette and should be loaded into an 'empty' Spectrum running in 48k mode. The questions and answers on the screen are fairly self-explanatory, and will provide you with re-saved files of slightly longer size because they include the extra carriage return

bytes at regular intervals so that "TLW" can screen the result in separate lines rather than as one gigantic long single line.

You will generally find it best to put 'spill' markers (14's) into the files, and when they are eventually loaded into "TLW" to work back from the end of the file paragraph by paragraph using the 'Insert' command to add firm paragraph ends and blank lines where appropriate. If they are micro-drive files you will need to use the menu option-2 loading approach to get them into "TLW" because they will be CODE whereas "TLW" saves to micro-drive as DATA.

◆ ◆ ◆

Section O
Printers and Interfaces

There are two or three aspects of "TLW" that will need to be correct for your printer to output text as you require —

1. The initialisation of the printer stream — done in the "TLW" Basic.
2. The appropriate arrangements — in the "TLW" m/c code section — to send carriage return and line feed bytes to the interface/printer.
3. An 'interpreter' to act as intermediary between "TLW" and the printer input port, if you are **not** using a Spectrum +3.

The first two are simply resolved in the software, the latter requires either a ROM based interface (Interface-1 RS232, Kempston 'E', ZX LPRINT-III etc.) or for older units an extra section of code to be held in "TLW" — either in the ZX-printer buffer or in RAM immediately below the "TLW" code — to 'feed-to' the interface. The Spectrum +3 of course has this capability 'built-in' although if you want to use an RS232 printer there are some minor revisions to be made.

After the requisite modifications have been made — and those if necessary in Section P for disc systems etc. — you can save the whole "TLW" program to your own system in its 'new' form by the direct command RUN 50 (tape) or RUN 70 (mdv/disc etc.).

Spectrum 128 RS232 port — non Spectrum +3

Select whichever mode is correct for your printer. In 128 mode you can send the text and control bytes for printing via the RS232 port by adding the following lines —

3140 FORMAT "b";x: REM x = printer baud rate.
3145 POKE 23349,39: POKE 23350,1

The Spectrum +3 RS232 port

Although the program is already set to run the centronics parallel port on a +3, it is simple to invoke the RS232 serial port by adding the line

5 FORMAT LPRINT "R": FORMAT LINE baud-rate

where 'baud-rate' is the value required by your own printer. If you don't put the second part of the line, the baud rate of 9600 will be assumed — note though that this may be too quick for some printers.

KEMPSTON 'E' centronics parallel interface

This is already set up. If your printer feeds two lines instead of one, then to stop this revise the line-feed value in the 'Print Guide' section of the program (whilst in "TLW") to 0, or change the BASIC line 3140 to —

3140 COPY: REM/0

The ZX LPRINT centronics parallel interface

This unit requires that the PCT Prefix is set to 1, and that the first byte of each of the PCT's contains the number of code bytes that follow. The 'Print Guide' [output call] should be set to 0, and the BASIC line is —

3140 LPRINT: LPRINT CHR$ 3: POKE 23679,148

and delete line 3145.

KEMPSTON 'S' centronics parallel interface

There are two possible approaches here. If you only require to print simple files from "TLW" then you can use the short piece of code already installed to do the job. Otherwise go to section (ii) to install the full version.

i. The short version.
Type the direct command **POKE 64954,245: POKE 64955,1: POKE 64964,5**
Change line 3130 to —
3130 PAUSE 50
and delete lines 3140 and 3145. Set the PCT prefix, [C/ret], and [L/feed] bytes, as you require them. Then set the 'Print Guide' [output call] to 0.

ii. The complete version. Add/modify the following basic lines —

```
30 CLEAR VAL "30000": GOSUB VAL "8000": PRINT ' ':
   LOAD "TLW2_D"CODE: LOAD "BUFF1"CODE: GOTO NOT PI
40 CLEAR VAL "30000": GOSUB VAL "8000":
   LOAD *"m";1;"TLW2_D"CODE: LOAD *"m";1;"BUFF1"
     CODE: GOTO NOT PI
55 SAVE "BUFF1"CODE 23296,256
85 SAVE *"m";1;"BUFF1"CODE 23296,256
3140 RANDOMISE USR VAL "23370"
3145 POKE VAL "23079", VAL "148"
```

Type **LOAD "BUFF1"CODE** and play your KEMPSTON 'S' interface cassette, to LOAD in the interface/printer software.

When it is LOAD'ed then RUN 3160 and set the [output call] in the 'Print Guide' to 23370, sort your [C/ret] and [L/feed] bytes for your printer, and the PCT's and prefix.

TASPRINT centronics parallel interface

There are two types — A and B. Most type B units regrettably have chips fitted that do **not** conform to Z80 standards, and these will probably prevent the program from running at all. Type A units require a section of code in memory to operate them — this is called 'tasbuff' and it should be LOAD'ed into the computer's printer buffer, after "TLW" has been LOAD'ed. Some changes should then be

made to the Basic and the m/code part. These are similar to the Kempston 'S' but of course use the CODE name 'tasbuff' instead of 'BUFF1' and an [output call] in the 'Print Guide' of 23296.

MOREX type — requires a simple output driver

This code is not installed in "TLW" but the 18 bytes you will require will easily fit into the space set aside for the Kempston 'S' code. You must write a short Basic program to do the work. MERGE and RUN it when "TLW" has been loaded —

```
3130 PAUSE 50
7000 DATA 245,219,251,230,1,32,
        250,241,211,251,62,1,211,127,175,211,127,201
7010 FOR N=0 TO 17: READ A: POKE 64954+N,A: NEXT N: STOP
```

Delete lines 3140 and 3145.
RUN 7000 when "TLW" is loaded.
Then delete lines 7000 and 7010.

Now use the 'Print Guide' command and set the [output call] to 0, and the [C/ret], [L/feed] and PCT prefix bytes as required by your printer. This code unfortunately will not LLIST or LPRINT from Basic.

dk'tronics type

This unit requires some extra software which is too large for the ZX printer buffer to hold, so it must be 'added-in' below the "TLW" m/c code. Different printers need different software, the Epson and AP-100A codes go into 541 bytes whilst the MCP671 needs 130 bytes more. The Epson installation is given as an example here, revise the figures accordingly for other printers.

Clear the computer out, type CLEAR 30000: LOAD "EPSON"CODE 48939 and run your tape of dk'tronics codes. When the correct code has loaded, type RANDOMISE USR 48939 to 'set' the code at this address. Now load "TLW" from the cassette and drop to Basic via option-0. Type the direct command POKE 65252,42: POKE 65253,191 — this will reset the 'upper limit to text file' in "TLW" at 48938 (42 + 256 × 191 = 48938) to keep the dk'tronics code all safe. Now go into the word processor via option-1 and set the 'Print Guide' [output call] to 48939, the [C/ret] and [L/feed] as required by your printer and if necessary reset the PCT Prefix to null (255). Go back down to the Basic listing again, type CLEAR 30000 and once more re-enter/exit from "TLW" — the m/c code is now 'fixed' together. The following Basic lines should be revised —

```
60 SAVE TLW2_D"CODE VAL "48939", VAL "16597": GO TO NOT PI
80 SAVE *"m";1;"TLW2_D"CODE VAL "48939", VAL "16597":
      GO TO NOT PI
3140 POKE VAL "23679", VAL "148"
```

and delete line 3145.

The Opus Discovery printer port

Add the following lines —

```
5 CLOSE # PI: OPEN # PI; "t"
3140 CLOSE # PI: OPEN # PI; "b"
```

and delete line 3145. If you find that the line-feeds are inadequate during page numbering/header/footer printing, delete the 'REM' from line 9160 as well.

Rockfort Disciple

This very comprehensive device incorporates not only the printer driving code but also the disc operating system as well. It is probably most advisable to make up a separate disc for "TLW" work, and this should incorporate the Disciple system-track with the correct setting-up approach for your own printer. You can then delete the printer initialising lines 3140 and 3145 in "TLW" — they are redundant. In "TLW" itself, the PCT prefix will need to be '–' (use 255) and the line-feed probably set to 10.

The printer aspects are best handled by Basic such as —

```
20 POKE @ VAL "6", NOT PI: GOSUB VAL "8000":
     GOTO VAL "1000"
3130 POKE @ VAL "6", SGN PI: POKE @ VAL "11", NOT PI
```

Interface 1 'FORMAT' problems

Sinclair issued two versions of the ROM in their interfaces. CLOSE#0: PRINT PEEK 23729 gives '0' for edition 1, '80' for edition 2. For edition 2 the "TLW" m/c code must be modified for the FORMAT command to work properly, as follows —

```
POKE 50415, 93
POKE 50416, 27
```

◆ ◆ ◆

Section P
Disc System Adaptations
(This section is not relevant for Spectrum +3 owners — all the work is already done for you)

"TLW" loads automatically from cassette after you enter LOAD"". The program in its standard form is written to work correctly with cassettes and Sinclair microdrives. To revise it to operate with a disc system or the Rotronics Wafa Drive, it is necessary to 'overlay' "TLW"s area of m/c code that does the microdrive work with one that lets all of the storage system computing be done in Basic, and then some lines must be added to the Basic to carry out the DOS computing.

The code overlay is held on the "TLW" cassette with the filename 'DSYS', and is installed as follows —

1. Get "TLW" into the computer in the usual way.
2. From the main menu, exit to Basic from option-0.
3. Enter the direct command **LOAD "DSYS"CODE** and run the cassette. When the code has successfully loaded, that's it for the m/c code aspects.

The Basic comprises a type-specific set of lines numbered 4000, 4100 and 4300 for LOAD, SAVE, ERASE and CAT'alogue — with a second standard line at 4500 to act as a common 'GO SUB' to the main four to provide the filename and all the numeric data that is required. The type-specific lines for the common disc system DOS handling are given below, followed by the '4500' line that all systems

require. The lines for the Disciple are on the cassette as 'X_BASIC', and can be MERGE'd into the TLW Basic program 'TLW2B_D' and then — if necessary — revised as given below for alternative systems as appropriate.

Opus Discovery

```
4000 GO SUB dtl: LOAD *"m";d;n$CODE base,space: RANDOMISE
     USR rtn
4100 GO SUB dtl: SAVE *"m";d;n$CODE strt,lng: RANDOMISE USR rtn
4200 GO SUB dtl: ERASE d;n$: RANDOMISE USR rtn
4300 GO SUB dtl: CLS: CAT d: RANDOMISE USR rtn
```

TRW Beta

```
4000 GO SUB dtl: RANDOMISE USR 15363: REM: LOAD n$CODE base
4010 RANDOMISE USR rtn
4100 GO SUB dtl: RANDOMISE USR 15363: REM: SAVE n$CODE
     strt,lng
4110 RANDOMISE USR rtn
4200 GO SUB dtl: RANDOMISE USR 15363: REM: ERASE n$
4210 RANDOMISE USR rtn
4300 GO SUB dtl: RANDOMISE USR 15363: REM: CAT n$(TO VAL "2")
4310 RANDOMISE USR rtn
```

Kempston KDOS

```
4000 GO SUB dtl: PRINT #VAL "4": LOAD n$CODE base,space:
     PRINT d: RANDOMISE USR rtn
4100 GO SUB dtl: PRINT #VAL "4": SAVE n$CODE strt,lng: PRINT d:
     RANDOMISE USR rtn
4200 GO SUB dtl: PRINT #VAL "4": ERASE n$: PRINT d: RANDOMISE
     USR rtn
4300 GO SUB dtl: PRINT #VAL "4": CAT: PRINT d: RANDOMISE USR
     rtn
```

Rockfort Disciple

```
4000 GO SUB dtl: LOAD *"m";d;n$CODE base,space: RANDOMISE
     USR rtn
4100 GO SUB dtl: SAVE *"m";d;n$CODE strt,lng: GO TO wp
4200 GO SUB dtl: ERASE "m";d;n$: RANDOMISE USR rtn
4300 GO SUB dtl: CAT d: RANDOMISE USR rtn
```

Rotronics Wafadrive system

First — **without** your Wafa system attached — load the whole "TLW" program into your Spectrum, revise the CLEAR address in lines 30 and 40 to '32000'. Re-save it all to a new cassette by RUN 50, switch off the computer, add and initialise the Wafa, and load in the **new** copy of the program. Then add the Basic —

```
4000 GO SUB dtl: LOAD # n$,base,space: RANDOMISE USR rtn
4100 GO SUB dtl: SAVE # n$,strt,lng: RANDOMISE USR rtn
4200 GO SUB dtl: ERASE # n$: RANDOMISE USR rtn
4300 GO SUB dtl: CLS: CAT # n$(TO VAL "2"): RANDOMISE USR rtn
```

The 'standard' GO SUB line at 4500 ...

The GO SUB subroutine to be written in as line '4500 is common to all the systems, although modified slightly for the TRW Beta and Rotronics Wafadrive hardware.

```
4500 LET d=PEEK VAL "65313": LET fp=FN p(VAL "50436"): LET
     n$="": FOR n=NOT PI TO VAL "9": LET n$=n$+CHR$ PEEK
     (fp+n): NEXT n: LET base=FN p(VAL "65268")+SGN PI: LET
     space=FN p(VAL "65193")-base: LET strt=FN p(VAL
     "65246"): LET lng=FN p(VAL "65248")-strt: LET rtn=FN
     p(VAL "50438): RETURN
```

For the Beta and Wafa systems, the line should start —

```
4500 LET d=PEEK VAL "65313": LET fp=FN p(VAL "50436"): LET
     n$=CHR$(d+VAL "96")+":": FOR n=NOT PI TO VAL "7": LET
     n$= ... etc.
```

Adapting the rest of "TLW" to suit your system

There are a few other lines of Basic in "TLW" which are 'specific' to the storage system that you use, and these will also require to be changed to suit it. Rather than describe here the exact new Basic, the 'intent' of the lines is explained so that you may use the instructions with your system to revise the lines to whatever is needed.

Line 40. Loads the 'TLW2__D' m/c code to its 'natural' address from the current drive.
Line 70. Saves the 'auto-run' Basic for 'TLW2B__D' to the current drive.
Line 80. Saves the 'TLW2__D' m/c code block from 49480, length 16055 bytes to the current drive.
Line 3230. Loads a code text file called 'n$' to address 's' from drive 'd'.
Line 3330. Loads a program 'n$' from drive 'd'.
Line 3370. Loads data file 'n$' into string 'x$' from drive 'd'.

Adapting the "TLW" Data File Program

This will also require about five changes, the lines to check being —

Line 102. This is to erase a pre-existing copy of a file before it is saved again. If your system will happily over-write an existing file with another of the same name, you can delete this line. Otherwise it should be tuned to suit your hardware, etc.
Line 105. This saves the 1024 byte data file held as Basic string 'a$' to drive 'd' with the name 'n$'.
Line 1070. This loads the 1024 byte data file called 'b$' from drive 'd' into the Basic string 'a$'.
Line 3025. Saves the Data File Program 'TLW__DF' to auto-run from line 10 onto drive 'd'.
Line 3040. Returns you to "TLW" — 'TLW2B__D' or 'Auto' etc.

To carry out this work, you should 'STOP' into Basic by keying CAPS SHIFT '6'.

Adapting the Key Define Program

The main job here is to add the extra Basic lines in the 4000 - 4500 region exactly as above, so that your disc system will operate in the same way whilst the Key-Define program is being used. Apart from this —

Line 620. Returns you to the "TLW" word processor — 'TLW2B_D' or 'Auto' etc.
Line 8027. Loads the "TLW" m/c code if not already in the Spectrum.
Line 8210. Saves the newly modified 'TLW2_D' m/c code back to your system.

Adapting the 'TLW_ADPT' adaptor program

Here there are seven lines that you should revise so your system can use them —

Line 60. Loads the m/c code block 'ADPT_CDE' to its natural address from drive 1.
Line 390. Loads the text file 'f$' to address 32000 from drive 's'.
Line 435. Erases pre-stored filename 'f$' from drive 's' before re-saving it. You may not require this feature — if not, just delete this line.
Line 440. Saves the text file 'f$' from address 32000, length 'lng' to drive 's'.
Line 1050. Loads "TLW" from drive 's' — 'TLW2B_D' or 'Auto' etc.
Line 9998. Saves 'TLW_ADPT' to drive 1, to auto-run from line 60.
Line 9999. Saves the m/c code block 'ADPT_CDE' from address 65000, length 75 bytes, onto drive 1.

Saving your personalised copy

When you have sorted out all of the items in the ADAPTING sections that you require, you should then ensure that the text file is EMPTY (use 'E' keys SYMBOL SHIFT 'Z') and then re-SAVE the "TLW" program onto your own system by typing the direct Basic command RUN 70 for mdv/disc/wafadrive or RUN 50 for cassette, so that it is all available when you require it. You will also need to save the other programs to your system — the modifications detailed above should have given you an insight into how to do that.

◆ ◆ ◆

Copyright and all that . . .

◆ ◆ ◆

Notes

Section Q
"TLW" System Technical Details

To get a value in Basic and screen it

LET y=PEEK address: PRINT y *for the single byte ones, and*
LET y=PEEK address+256* PEEK (address+1): PRINT y *for the double byte ones.*

To re-set a value from Basic

LET y=value: POKE address,y *for the single byte ones, and*
LET y=value: POKE address,y−256* INT(y/256): POKE address+1, INT(y/256) *for doubles.*

Data or variable description	Numeric values, etc.
"TLW" machine code block	49480 to 65535.
Call address —	52410.
Re-entry address during printing —	65154.
Processor operating mode —	Interrupt mode 2.
Text file size —	0 to 25k, base at RAMTOP +2.
Printer control token file —	52193 to 52408 inclusive.
Format: 24 separate tokens, each with —	
Bytes 1 to 5 — ASCII token bytes.	
Bytes 6 to 9 — code bytes, value 0 to 254 (255 not sent).	
Standard printer token prefix — 0 to 254 —	52409, 1 byte (255 not sent).
Printer Control Tokens 1 to 24 —	Byte values 128 to 151.
Print-time Call Tokens 90 to 99 —	Byte values 152 to 161.
Print-in $ Tokens —	Byte values 230 to 255.
Printer interface code block, Kempston 'S' etc. —	64954 to 64977 inclusive.
Calculator memories —	10, BASIC variables m0 to m9.
Calculator variables —	ALL BASIC variables usable.
String inclusion length —	Up to 4k-bytes.
String name inclusion range —	26, A$ to Z$.
Some system variables —	
Max screen drive no (ASCII) —	50470, 1 byte.
Drive no limiter —	64501, 1 byte.
Text file clear-out call —	64978, returns base address.
File-pointers reset call —	65010.
Alarm setting —	65189, 1 byte.
Top of text file space —	65193, 2 bytes.
Bottom of text file —	65195, 2 bytes (RAMTOP +2).
Text file upper limit —	65252, 2 bytes.
Text file top byte —	65268, 2 bytes.

Print tr. list, from —	65173, 8 bytes.
Print tr. list, to —	65181, 8 bytes.
Carriage return byte —	65241, 1 byte.
Line feed byte —	65242, 1 byte.
Basic line to exit to —	65244, 2 bytes.
Print-fetch start adr —	65246, 2 bytes.
Print-fetch last adr —	65248, 2 bytes.
Print-fetch current adr's —	65250, 2 bytes.
Text file upper limit —	65252, 2 bytes.
Text file top byte —	65268, 2 bytes.
Cursor address in file —	65270, 2 bytes.
Cursor column in text —	65272, 1 byte.
Cursor line in text —	65273, 2 bytes.
Left margin —	65281, 1 byte.
Right margin —	65282, 1 byte.
Print-out line spacing —	65285, 1 byte.
Number of print copies —	65286, 1 byte.

Printed by MEIRFIELD PRINTERS Ltd.
Swansea (0792) 461102