TWIN

reference manual

ARM Evaluation System

Acorn OEM Products

TWIN

Part No 0448,009
Issue No 1.0
21 July 1986

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

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

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

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

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

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

ISBN 1 85250 004

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

Contents

1. Introduction

TWIN is a twin-screen editor designed specifically for the ACORN RISC MACHINE (ARM). Particular attention has been paid to its ease of use and interaction with assemblers, compilers and high-level languages, all of which can be run directly from TWIN.

Compilation errors, reported by their line number, can be diverted to TWIN files and shown on one of the screen windows. The other screen window can then be used to display the offending line of source, and the programmer can make any required alterations, deletions or insertions in the text. Work on the text may continue while some other task is running concurrently.

1.1 Facilities

- Edit text from one of ten internal RAM buffers
- View portions from two buffers simultaneously
- Run background tasks, the results of which can be seen on a screen window
- Copy text from one window to another
- Transfer copied text as an input to a background task
- Go to specific lines in the text
- Quickly find, replace or count text specified in a variety of ways
- View on-screen help and status information, if required
- Use filing system and operating system commands
- Maintain a software clock and date and time-stamp document files
- Print text

When TWIN is first loaded, a window is set to the same line and column dimensions as the screen, and it takes its text from buffer 0. The window size can be reduced, enabling the screen to display both static and dynamic help information. Alternatively, a second window which takes its text from a different buffer may be called on to the screen. TWIN can organise as many

as 10 RAM buffers, of which any two can be viewed by screen windows at any given time.

1.2 Conventions used in this manual

TWIN defines a set of function keys to provide a range of powerful one-key commands. These keys bear the legends f0–f9, but they all have their own names, given by a printed legend case (the key card). In this manual, pressing a function key is indicated by the name of that function key, for example:

Pressing function key one ([f1]) would be shown as;

[command line]

Key combinations involving [SHIFT] and [CTRL] are printed in key-legend form, for example:

[connect buffer]

For consistency all other single key presses involving control keys are printed in boxes, for example:

Press the RETURN key [RETURN]

Press the ESCAPE key [ESCAPE]

Press the DELETE key [DELETE]

Press the COPY key [COPY]

Many language texts typed into TWIN have their own interpretations of the punctuation symbols and special symbols which are available from the keyboard. These are:

```
! " # $ % & ^ @
( [ { } ] ) | : . , ;
+ - / * = < > ? _
```

This often makes it difficult for you to determine precisely which characters on the printed page are explanatory or descriptive, and which (if any) are the ones which the language will accept as having the correct syntax. A typewriter-style typeface has been used to indicate both text which appears on the screen and text which can be typed on the keyboard (for example, a command given to the filing system). This is so that the position of relevant spaces is clearly indicated.

The syntax of commands is shown in a typeface which is distinct from ordinary text. For example: `$.arm.library .`

1.3 Simple use of TWIN

1.3.1 Installing TWIN

TWIN is supplied on a 5.25 floppy disc in Acorn ADFS format. A function-key card is supplied for the TWIN editor and this should be fitted under the transparent strip behind the function keys.

TWIN may be loaded in three ways from the ARM command line:

`TWIN` [RETURN]

creates a blank document in buffer 0.

`TWIN` *filename* [RETURN]

loads TWIN and the named file into buffer 0.

`TWIN` *filename filename* [RETURN]

loads TWIN, the first file into buffer 0, and the second file into buffer 1. Both windows are displayed.

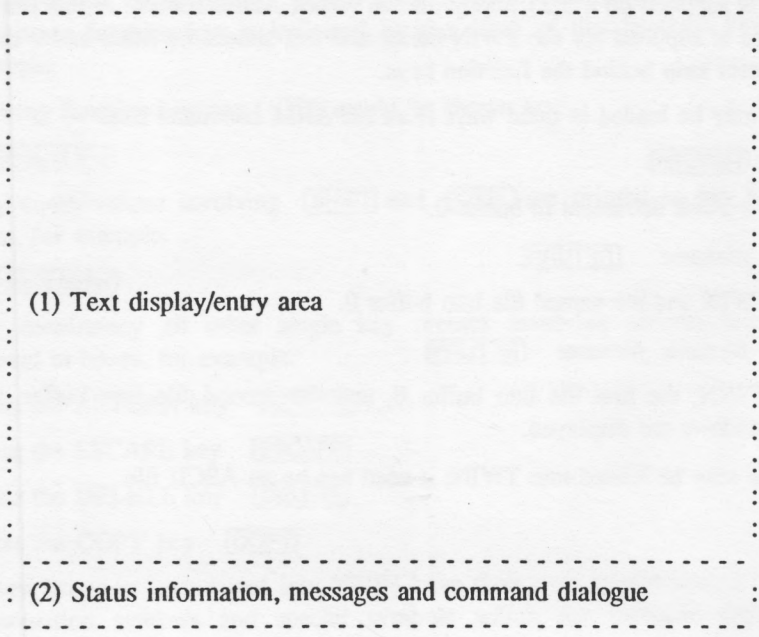
Any file may be loaded into TWIN; it need not be an ASCII file.

1.3.2 The screen layout

When TWIN is entered by typing:

`TWIN` [RETURN]
or
`TWIN` *filename* [RETURN]

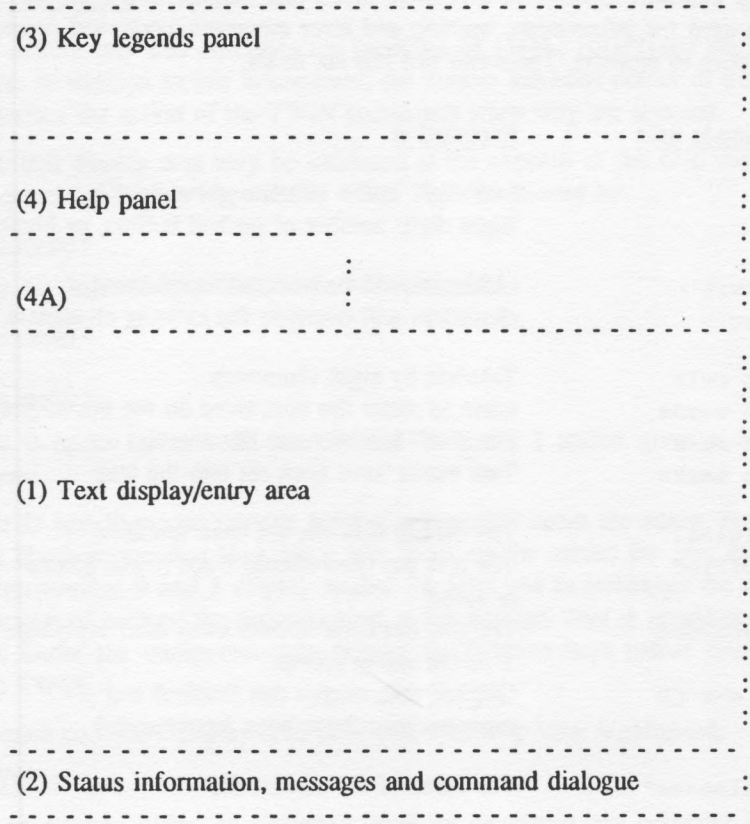
only buffer 0 is active and one screen window is displayed. The screen format is as shown below:



This screen layout is only one of several available, and until you are familiar with TWIN the Descriptive Mode screen will probably prove more helpful. This mode is selected by pressing;

set mode D

There are now four areas on-screen:



(1) Text display/entry area

Displays a portion of the text in the buffer, including the text being typed. This area may contain one or two windows, which need not necessarily be identical in size.

(2) Status information, messages and command dialogue

Usually indicates the current text entry mode and number of marks set, but is also used for information, warning and error messages, command prompts and replies to prompts. The status line has six fields:

Example field	Explanation
`01`	Left digit: buffer number of current file Right digit: number of pushed buffers, or blank.
`Insert`	characters will be inserted into the text
`Over`	characters will overtype the existing characters
`Tab cols`	Tabulate by eight characters
`Tab words`	move to under the next word on the above line
`One mark`	One mark has been set into the text
`Two marks`	Two marks have been set into the text
`Original`	The current text has not been modified
`Modified`	The text has been altered since it was loaded or saved
`Discarded`	The text has been thrown away after replying Y to the safety prompt
`LF <-> CR`	Original text, except that linefeed and carriage-return have been interchanged
`"Filename"`	The name of the current file
`<Date>`	The date of a dated file
`**Command File**`	a file which has a load address of 0 and which has an execution address of FFFFFFFF and which will be read as a series of commands by a filing system
`<Address>`	The load and execute addresses of a binary file

Miscellaneous information, such as the results of global replacements, is also shown in this line.

(3) Key legends panel

Describes the TWIN command provided by each function key.

(4) Help panel

By default this area describes the functions of [TAB], [COPY] and the cursor keys. In addition to this information, the bottom left-hand corner of this area describes the action of the TWIN commands when they are selected.

The text display area may be increased at the expense of the help messages by selecting the key legends (K) mode. This is achieved by:

[set mode] K

The key legends may likewise be removed with:

[set mode] 0
or
[set mode] 3

The 0 option gives a 32 line screen, while the 3 option gives a 25 line screen.

The D and K modes provide helpful information about the editor, but have the disadvantage that they leave less room on the screen for text than the corresponding 0 and 3 display modes. To help you to remember the editing function of each of the function keys, a function-key card is provided: insert this under the transparent strip behind the function keys before starting to use TWIN.

Control codes are displayed as their corresponding letter highlighted:

[CTRL] A

as an inverse video A. However, linefeed characters are normally hidden from view unless they need to be shown. To display linefeeds:

[show newlines]

Linefeeds appear as a inverse video left-arrow rather than a inverse video J. The carriage-return character (CTRL M) cannot be hidden. The end of the buffer marker is displayed as an inverse video asterisk.

(5) The top status line

If you value an on-screen time display, but do not want to use mode D, there is mode T, obtained by pressing;

[set mode] T

This gives a status line similar to the following abbreviated version:

```
TWIN'T'mode  Shift  f5  to  change  mode  TWIN  at  &001D0000
10:16:46 15-May-86
```

Amongst other things, it shows the load address of the TWIN in use, and the time and date.

A screen print-out of a full TWIN help screen with two windows in use is shown in appendix E.

1.3.3 The keyboard

The letter, number and punctuation keys have their normal functions within TWIN, but some of the others have modified functions:

[RETURN]	type new-line character (ASCII &0A), move cursor or complete a command
[TAB]	moves the cursor (2 modes)
[SHIFT]-[TAB]	switches TAB modes
[CTRL]-[TAB]	expands tabs in text
[f0] to [f9]	invoke TWIN commands
[SHIFT]-[f0] to [f9]	invoke TWIN commands
[CTRL]-[f0] to [f9]	invoke TWIN commands
[CTRL]-[SHIFT] - [f0] to [f9]	invoke TWIN commands
[COPY]	deletes the character at the cursor
[SHIFT]-[COPY]	switches cursor to cursor copying mode
[CTRL]-[COPY]	deletes the line marked by the cursor
[CURSOR KEYS]	moves cursor a character/line at a time
[SHIFT]-[L/R CURSOR]	moves cursor left or right by a word
[SHIFT]-[U/D CURSOR]	moves cursor up/down by a window-page
[CTRL]-[L/R CURSOR]	moves cursor to the start or end of line
[CTRL]-[CURSOR U/D]	moves cursor to beginning or end of text
[CTRL]-[SHIFT] - [U/D CURSOR]	moves both windows up/down by a page
[ESCAPE]	cancels an incomplete command or leave cursor editing mode.

The function keys are specially defined by TWIN. Their functions will be explained as and when necessary. A brief description of all of them is given in appendix D.

2. Writing code in TWIN

2.1 Types of file

TWIN is capable of creating:

- Documents which may be any text or language-related source-code. These documents can be date-stamped automatically by TWIN. The current date and time is obtained either by asking you at initialisation time, or by interrogating a network. Both procedures can be incorporated into the early morning start !BOOT file. Should the time not be set, TWIN uses the **Command file** form of document, explained below
- Command files can be used in the same way as a document or date-stamped file, but they are marked as having a load address of -1 and an execution address of 0, enabling the operating system to use them as a series of input instructions. Command Files can therefore be executed under the TWIN* or ARM* prompt and will perform automatic actions on memory or filing-systems, including TWIN itself.

In addition, TWIN is capable of loading a pure binary file. The file may be inspected, and it is permissible to edit it by overtyping. Usually it would only be sensible to edit recognisable ASCII strings. If edited in insert mode the file would be increased or decreased in length with fatal consequences upon branch and jump instructions. The load and execution addresses of the loaded binary file are shown on the status line.

2.2 Date stamping TWIN documents

In order to allow advanced tools to run, files can be date stamped. The date is maintained in the software clock on the I/O Processor.

TWIN is loaded with the correct time and date on a stand-alone system by using the file DATE. On a system connected to a network, you can type `Date-net`.

The filename DATE expects to be followed by parameters in the form:

```
DATE DD-MMM-YY HH:MM:SS
```

for example:

```
DATE 14-May-87 10:37:08 [RETURN]
```

The system has been dated as 14th May 1987, at a time of 10:37am and 8 seconds.

TWIN will start up with the current date and time shown on the status line. The date of the file will be updated when the file is saved. If the date has not been set, TWIN will initialise a command file. However, a dated file may be loaded into a running version of TWIN which has not had its date set. If an attempt is made to save a dated file from an undated TWIN, a prompt message will advise that the file cannot be marked with the current date and time and will seek permission to save it as undated.

2.3 TWIN and the operating system

Any of the filing and operating system commands may be used from the editor, by first pressing

[command line]

The bottom of the screen will display the TWIN* prompt, and the operating system can be used. There are two options to return to TWIN:

[ESCAPE]
or
[RETURN]

when the TWIN * command line is blank.

While in the command line mode, the function keys will revert to their function as soft keys so will generate any text assigned to them previously using the command

`*KEY` *n string* [RETURN]

(see your micro user guide for details of this command)
Note also that the text assigned to these keys is also available to TWIN when cursor editing (see section 3.3.2) has been selected.

2.3.1 Command files

Although TWIN is normally used directly by the keyboard operator, it can also be driven by command files which have been created for the purpose.

Command files are best used when you wish to repeat the same editing operations on several files or when the editing is such a complex sequence that it is best done in stages.

Since command files execute without your interaction, their construction and testing should be done in stages so that careful checks can be made to ensure that a command file does what it is supposed to do and no more.

The first line of a command file should usually be blank, to take TWIN out of command line mode.

The following example sequence of commands makes a language the background task in a twin-window environment.

Enter TWIN in **Command File** mode and type into the editor the following:

[RETURN]
[enter character]
[toggle window]
[enter character]
[push]
[enter character]
[task status]
language name
[RETURN]
[enter character]
[toggle window]

This enters six characters (plus the number of characters in the language name) into the TWIN command file. These commands are typed into TWIN using the special [enter character]-[function.] For example, [enter character] then [toggle window] places the toggle window character into the command file, and so on. The screen will show whatever symbols are associated with these characters: they are likely to be an assortment of graphic, mathematic and foreign language symbols, and have no particular significance.

Note the use of [push] to ensure that the language file doesn't overwrite any TWIN file when it is loaded.

Such command files don't always have to run under TWIN, as shown in the example given above. They always start in the operating-system environment, but may call, enter and leave language environments. TWIN will often be involved because the command sequence is likely to want to make use of TWIN's editing features.

This type of command file is often used as a filing system !BOOT file, to select the editor and set options (such as display carriage returns, release scroll margins, select text colour, obtain date and time from a network and so on) at the start of editing.

The text of TWIN can be placed into a language by pressing [exit to]. After checking to see if any altered text needs to be saved to the filing system, TWIN will go to a named language or to the operating system if [RETURN] on its own was pressed.

Loading and saving with the filing system is described in chapter 5.

2.4 TWIN concurrency

To run a task concurrently with TWIN, it is necessary to select a buffer to run it in. Pressing [task status] will supply the prompt command on the new window's status line, and a task can be called – it may be a language or a utility. All output from the task will print to the buffer for as long as the task runs, and the contents of the buffer can be viewed in the TWIN window. Editing on another of TWIN's buffers can take place while the task is running, but the task will stop occasionally as TWIN loads and saves files, or gives prompt and error messages which require some immediate response from you. TWIN's clock will not be updated.

When the task's buffer is on-screen, the window is the output screen for the task, and keyboard information (characters typed in the range &00 to &7F) can be passed to the program. All the editing and cut-and-paste facilities of TWIN are retained, and are available to assist in entering commands into the task program. A [copy block] operation can be performed between the TWIN window and the task window which directs all the copied characters to the task.

The task window normally follows the information flowing into the end of the task buffer – in this way the latest output is always on view. This is a task-linked situation, the default. However, pressing [auto bottom] unlinks the task so that the task window no longer follows the data streaming into the

buffer. The task-unlinked state allows you to move the cursor all over the task buffer.

[auto bottom] toggles between the task-linked and task-unlinked state. The task window can be re-sized, and may be removed from the screen altogether. If you are in a TWIN window and it is inconvenient to go into the task window, pressing key

[task bottom]

forces the task-buffer cursor and window to the end of the task buffer.

When a task is started, half of the available memory is allocated to it for the output: this will be used up gradually, but there is no check for running out of memory. Only one task may run at one time, and a new task cannot be started until the one running has finished or is cancelled.

TWIN can read the error messages from compiler output and go to the offending lines in the source file. [next line] looks for the text `line` (in any case of characters) in the other text buffer: if found, it then treats the following text as input to a [goto] operation but obviously only the absolute line number option is of real interest.

In the task-linked mode of operation, TWIN must redraw its window to display any information arriving from the task: this slows down the task in hand. If faster operating speeds are required, the task window may either be reduced in size to its smallest dimension (five lines), or turned off completely. Repeated pressing [expand window] while in the TWIN window (as opposed to the task window) reduces the size of the task window, and pressing [close window] while in the task window removes the task window from the screen.

A simple worked example of how to run the ARM assembler from TWIN is given in appendix A.

3. Editing functions

3.1 Insert and overtype modes

TWIN offers the choice of typing in overtype or insert mode. In overtype mode the characters typed replace those currently at the cursor position on the screen. In insert mode they push existing text to the right to make room for themselves. The mode in use is displayed in the bottom-left corner of the screen (the status area). To switch between these modes, use [insert/over]. Certain keys behave differently in the two modes – these differences will bementioned as the functions of those keys are introduced.

When the program being typed reaches the right-hand side of the screen it overflows on to the next screen line; no special character (such as hard or soft returns) is placed in the text at the point where the text breaks to start a fresh line on the screen. Therefore, to start a new line, it is necessary to press [RETURN]. TWIN imposes no limit on the number of characters that can be typed, other than the maximum buffer size, dictated by the available RAM.

Pressing [RETURN] in insert mode puts a new-line character (ASCII 10) at the current cursor position and the cursor moves to the start of the next screen line. The return character is important as it marks the end of a line to the editor. Return characters are normally invisible. To make TWIN display return characters press [show newlines]. New lines show as left-arrows on a highlighted square.

Pressing [RETURN] in overtype mode moves the cursor to the start of the next screen line.

To correct minor errors while typing, use [DELETE] to erase characters to the left of the cursor. In insert mode, this removes the character from the screen and buffer; in overtype mode it doesn't remove it, but replaces it with a space.

3.2 Moving around the source text

The cursor keys have a repeat action if they are held down. The screen window can only show part of the program text; this is regarded as a page. To see another page, use the cursor keys as shown on the next page.

To move through the TWIN buffer a line at a time (scrolling), use the up or down cursor keys – the screen will scroll when the cursor comes within a few lines of its top or bottom (this distance is the scroll margin, and can be changed using [new margins]).

To move through the text buffer a page at a time, use the up or down cursor keys with [SHIFT].

To move to the start or end of the text buffer, use the up and down cursor keys with [CTRL].

To move the cursor under the start of words on the preceding line, press [SHIFT]-[TAB] until `TAB words` is displayed on the status line at the bottom of the screen, then use [TAB].

To move the cursor to the next tab position to the right, press [SHIFT]-[TAB] until `TAB cols` is displayed on the status line at the bottom of the screen, then use [TAB].

To move to a particular line in the buffer, use [goto]. This displays the current line number (`At line` ***nnn***, `character` ***xxx***, `new line:`), where ***nnn*** is the number of returns between it and the start of the buffer. You may go directly to a line number, or to a mark, or move relative to the current cursor position forwards +***nn*** or backwards –***nn*** .

To move to a known piece of text in the buffer, use [find and replace] – see chapter 4.1

The screen will normally scroll when the cursor is moved to within four lines of the top or bottom of the screen (the scroll margin) to ensure that text surrounding the cursor can be seen properly.

The scroll margins

It is possible to alter the action of scrolling by the [new margins] key. Using [new margins] brings the prompt:

```
set Bottom margin, set Top margin or Remove margins ?
[B/R/T]
```

on to the status line.

[new margins] B

sets the bottom margin to the cursor position

[new margins] T

sets the top margin to the cursor position

[new margins] R

removes the scroll margins altogether.

The margins will be reset to TWIN's preferred positions if the size of the window is altered by pressing [expand window] [toggle window] [close window] or changing screen modes. Toggle window operations which do not reset the size of the windows and do not reset the margins.

3.3 Changing the program text

3.3.1 Small alterations

Small changes to text in the buffer are made using the [DELETE] or [COPY] keys for single-character deletions backwards and forwards, or by overtyping in overtype mode.

In insert mode, [DELETE] deletes characters to the left of the cursor; [COPY] deletes characters to the right of the cursor; typing inserts the new text at the cursor.

In over mode, [DELETE] replaces characters to the left of the cursor with spaces and moves the cursor to the left; [COPY] behaves as it does in insert mode. Typing replaces existing text with new text.

To remove a blank line or join two lines, the new-line character, together with any spurious spaces which might be present, should be removed. This is often easiest to do when the new-line character is made visible and this is simply achieved by pressing [show newlines].

Whole lines may be removed by pressing [CTRL]-[COPY] simultaneously.

3.3.2 Cursor editing

To copy short sequences of text from the screen, place the cursor at the position where the copied text is required and press [SHIFT]-[COPY] to select cursor editing mode. Next, move the cursor to the text which is to be copied. Pressing [COPY] now copies the text, character by character. Extra text may be typed in and existing text may be deleted using the [DELETE] key. The copying process cannot continue past a new-line character. Press [ESCAPE] to return to insert or overtype mode.

Two drawbacks to cursor editing limit its usefulness: control characters (including the new-line character) cannot be entered as part of the copy, and, if the destination line is before the source line, copying may invoke a line-scroll and so impair a successful copy from that point onwards.

The singular advantage of cursor editing is that it disables the editor commands on keys [f0]-[f9] thus enabling those keys to be used as soft keys to type complete words, phrases or ASCII codes 127-255 with a single keystroke.

Although these techniques are very useful, even more powerful editing commands are provided to delete, move and copy blocks of text.

3.3.3 Cutting and pasting text

These blocks are only limited in size by the text in the buffer. The first step in using any of these commands, is define the block of text to be used. This is done by moving the cursor to the start and/or end of the block and marking its position with [mark place].

To delete a block of text, mark the start or end, then move the cursor to the other end of the block and use [delete block].

To move a block of text, mark the start and the end (in either order), then move the cursor to the destination outside the block and use [move block]

To copy a block of text, mark the start and the end (in either order), then move the cursor to the destination and use [mark place]. From now on, each time [copy block] is pressed, a further copy will be placed at the cursor position, until the marks are erased using [clear marks].

To copy a file from the filing system into existing text in the TWIN buffer, position the cursor where the text is to go, then use [insert file], typing the file name in response to the prompt `Type filename to insert:`

TWIN removes marks automatically after the block delete or move commands. Marks are not removed automatically after the copy command, to allow multiple copies of marked text to be made easily. Marks can be cleared at any time with [clear marks]. It is best to do this (as a matter of good practice) as unwanted marks can modify the action of certain commands such as [global replace] and [save file], and can prevent operation of some commands, causing the error message `Bad marking`.

The number of marks currently set is displayed in the status line at the bottom of the screen. A maximum of two marks is allowed in any of TWIN's buffers: buffers have their own set of marks. This allows copying between one buffer and another, providing that marks have first been cleared in the destination buffer. The text marked should be in the source buffer, and the cursor placed in the destination-position of the current buffer. A copy (but not a move) is then made between the source and destination buffers.

TWIN will use as its source buffer either the buffer which occupies a screen window, or, in cases where only the destination window is on the screen, the buffer which has just been removed from the screen using the [close window] key.

3.4 Clearing and restoring text

To delete the entire contents of the text buffer, use [clear text], then use one of the options in response to the following prompt:

```
Clear text Y (dated),shf-f9 (auto-exec),D {discard}/N
```

`Y` kills the current file and begins a fresh one which will be date-stamped.

[clear text] kills the current file and begins a fresh one which will be executable as a command file.

`D` removes the altered status from the current file. The file remains in memory but it will now be possible to remove it by a LOAD *newfile* command which does not ask permission before overwriting the file with the *newfile*.

`N` means do not clear text. [ESCAPE] may also be pressed, in which case the text is not cleared.

If no action is taken, the command times-out and automatic escape occurs.

Never press [BREAK], since this resets the computer.

Use [clear text] very carefully, there is no command to restore lost text, although appendix C offers some advice in this direction.

4. Finding and replacing text

TWIN provides two very powerful find-and-replace commands, which can perform the following types of functions:

- rapidly changing all or selected occurrences in the TWIN buffer of one string into another string - correcting spelling mistakes, improving consistency, expanding or shortening variable names in programs, expanding abbreviations, removing comment lines from debugged programs to make them shorter
- finding the next time a text string appears in the TWIN buffer - moving to a line where the content is known, but the line number is not, checking procedure/function parameters in programs and so on
- counting the number of times a string occurs in the TWIN buffer - checking the size of a document, guarding against over use of particular phrases, analysing style, looking for under-used variables/procedures/functions in programs and so on.

4.1 Find and replace

This is the interactive find-and-replace function [f4]

Remember when using the find-and-replace function to place cursor at start of text. When [find and replace] is pressed, the function keys take on new meanings, and these will be explained later. [find and replace] gives ample prompts which should prevent incorrect replacements occurring (the results of which might be difficult to reverse). [global replace] assumes that all occurrences will be replaced and will go ahead and act upon the whole file (if no marks are set). Global replace on the ARM processor is extremely fast.

After selecting one of these commands, a prompt requires you to specify the text to search for in the text buffer, and what to do with any matching textfound. Matching text may be deleted, counted, replaced or moved. The specification of the text to match is the target string, occurrences of matching text are found strings and the specification of text with which to replace found strings (if supplied) is the replacement string.

The find-and-replace commands can be used simply, as described immediately below, but to make the best use of these powerful commands, refer to the section following.

When [find and replace] is selected, the prompt `Find and replace:` is shown on the status line. The find string should then be typed, terminated by [RETURN], whereupon TWIN will look for the string. If it finds it, the prompt:

`C(ontinue), E(nd of file replace), R(eplace) or ESCAPE/RETURN`

will be given.

Pressing `C` causes TWIN to keep looking for more occurrences of the string.

Pressing `E` causes TWIN to prompt for a replace string and then the command behaves like a global replace from the present position to the end of the file.

Pressing `R` causes TWIN to prompt for the replace string which is used only to replace the string at the cursor position.

Note that a replace string need not actually be typed in the above two examples.

Pressing [RETURN] instead of a replace string will delete the find string, while pressing [COPY] or [SHIFT]-[COPY] will call up a previous find string or a previous global string.

Pressing [ESCAPE], [RETURN] or [DELETE] aborts the command.

Markers have no significance to the find-and-replace command.

The following functions can be achieved easily using [find and replace] with the appropriate string:

To find the next occurrence of the target string use:

[find and replace] *target string* [RETURN]

To selectively delete occurrences of the target string use:

[find and replace] *target string* [replace by] [RETURN]

To selectively replace occurrences of the target string with a single replacement string use:

[find and replace] *target string* [replace by] *replacement string* [RETURN]

To selectively replace occurrences of the target string with different replacement strings use:

[find and replace] *target string* [RETURN]

and follow the prompts.

To use the previous target and replacement strings:

[find and replace] [RETURN]

To use the previous target with the option to alter the strings in some way use:

[find and replace] [COPY]

at this stage you may edit the find string

[RETURN]

Pressing [SHIFT]-[COPY] recalls the string used by the other function global replace or find and replace, which can likewise be altered if required.

To cancel an incomplete string on the command line, press [ESCAPE]. [ESCAPE] will abort a find-and-replace command while it is operating, but the command is likely to finish before an escape can be effected.

All characters may be used freely in target and replacement strings as they have no special meaning to the search and replace commands.

Capital and lower-case letters are equivalent in the target string. They will both match the same letters of either case (for example, cat will match CAT or Cat or caT and so on).

4.2 Special find-and-replace functions

[f0] - [f9] have new meanings once [find and replace] or [global replace] have been selected. They are used to introduce meta characters into either or both the find string and replace string. On the status line a brief description of their job appears in reverse video.

key	brief meaning
[new line]	the new-line character, usually LF, 10
[found section]	followed by 0-9, the nth wild character found
[found string]	the entire found string
[most items]	the most possible matches for the following item
[many items]	as many matches of the following item as necessary
[replace by]	begin the replace part of the string
[not item]	do not match the next item
[any character]	matches any character
[alpha-numeric]	matches A to Z, a to z, 0 to 9 and _
[digit]	number matches 0 to 9
[start set]	begin set of items to match
[end set]	end set of items
[sub range]	range, for example A to Z
[control]	control–code the following item
[set top bit]	set top bit of the following item
[exactly]	match exactly the next item
[toggle case]	toggle case sensitivity.

At the beginning of every [find and replace] or [global replace] command, case is taken to be insensitive.

4.2.1 Differences between find and replace and global replace

The global replace function takes the target and optional replacement string typed in response to the prompt and acts on all matching strings found in the text buffer, giving a count of the number found. If a single mark is set, it will only search between that mark and the cursor.

The find string function takes the target and optional replacement string and searches from the present cursor position towards the end of the text buffer. When a match is found the cursor is moved to the found string and the screen updated, if necessary, to show the found string.

4.3 Global replace

[f5] performs an automatic count of occurrences (with no replace) or a fully-fledged, non-selective find and replace. The whole text, or just a block of text, may be specified.

To replace all occurrences of the target string with the replacement string, use:

[global replace] *target string* [replace by] *replacement string* [RETURN]

To delete all occurrences of the target string, use:

[global replace] *target string* [replace by] [RETURN]

To count the number of occurrences of the target string:

[global replace] *target string* [RETURN]

The target string may consist of an arbitrary mix of characters and meta symbols but excessively lengthy or complex target strings may be processed very slowly or be rejected by TWIN as too complex. Mistakes are reported by a syntax error message before the find and replace takes place.

4.3.1 Specifying single characters

(1) To match a letter which isn't case sensitive, such as `a` or `A`, it is typed as normal.

(2) To match a letter of a specified case use [exactly] any letter

(3) To match the new-line character use [newline]

(4) To match a control character, for example, 12 use [CTRL] `L`

(5) To match ASCII 128-255 , for example, 129 use [set top bit] [CTRL] `A`

(6) To match 257 use [set top bit] `A`

(7) To match any letter, digit or underscore use [alpha-numeric]

(8) To match any digit (0 to 9) use [digit]

(9) To match any character (ASCII 0-255) use [any character]

(10) To match any one of a set of characters, for example: xyz, use [start set] `xyz` [end set]

(11) To match any one of a range of characters, for example: f-i, use `f` [sub range] `i`

(12) To match any character other than that specified, for example: don't match `n`, use [not item] `n`

NOTES:

- the case of letters used in specifying groups is not interpreted ambiguously as it is with explicit specifiers (for example, a will match a or A, but e-g will match e but not E
- control characters can also be typed as [CTRL] *any key*, with the exception of [CTRL]-[M], [CTRL]-[U] and [CTRL]-[|] (these are [RETURN], [delete line] and [ESCAPE])

The three character specifications that define groups of characters, [start set] [end set], [sub range] and [not item] may use other single character specifiers to define that group, so that the following examples are all valid specifications of single characters:

[start set] - [alpha-numeric] $ [end set]

matches any alpha-numeric character or the dollar sign

[start set] @a [sub range] z [end set]

matches any lower-case letter or @

[not item] [start set] [digit] [end set]

matches any non-numeric non-space character

[not item] 3 [sub range] 6

matches any character other than 3, 4, 5 and 6

[start set] a [sub range] ce [sub range] z [end set]

matches any lower-case letter other than d.

4.3.2 Multiple-character target strings

A multiple-character target string consists of a combination of single character specifiers as described above. Note, however, the use of [toggle case] This is used within the find-and-replace commands to toggle between case insensitivity (the default situation) and case sensitivity.

4.3.3 Specifying variable length strings

Using the keys described already, any strings in the program text that are of a fixed and known length may be matched. Two extra keys are provided to allow matching of variable length strings, provided that the variable length portions can be defined as repetitions of single character specifications. The extra keys are:

[most items] taken to mean the most of a group of identical characters

[many items] taken to mean as many of a character as necessary.

These keys are used in the target string by placing them immediately before a single character specification as defined above; the resulting object is a string specification.

The first key, [most items], creates a string specification that will match as many repetitions (one or more) of the following character specification as it can. Thus:

[most items] `a` matches `a` or `AA` or `aAAAaAAaaaa` and so on

[most items] [alpha-numeric] matches any word

[most items] [digit] matches any number (6, 66, 3454545)

[most items] [not item] [newline] matches a non-blank line.

The second key, [many items], by contrast, creates a string specification that will match as few (down to zero) repetitions of the following character specification as it can.

The [many items] key is most useful for specifying unimportant filler characters, that is, characters that need not be there at all to match the target string.

The [most items] key is generally easier to use, since the use of [many items] often requires a specification of one more character than is actually in the match, causing occasional skipping of the next found string in the buffer. Searching using it is much faster than using [many items].

For example:

target	match
`cat`	Cat or CAT or cat etc.
`c` [many items] `at`	Cat or CT or caaaaT etc.
`d` [most items] `og`	Dog or dooog or DOG etc.
`ca` [many items] `t`	CA, ca cat etc.
`do` [most items] `g`	dog, DOGGG, etc.
[most items] [alpha-numeric]	a word

To count words of three or more characters use the syntax:

[global replace] [ALPHA] [ALPHA] [most items] [ALPHA] [RETURN]

To find uses of multiple spaces use:

[find and replace] [most items] *(space)* [RETURN]

Note that while variable length strings are defined using single-character specifiers, they cannot be treated as single-character specifiers. For example, although:

[most items] [alpha-numeric]

matches words,

[not item] [most items] [alpha-numeric]

does not match non-words, but any character other than

[most items] followed by an alpha-numeric.

Use [most items] [not item] [alpha-numeric]

to match non-words.

4.3.4 Specifying the replacement string

If the replacement string is absent, as in the case of:

target string [global replace] [RETURN]

then the replacement string is null and found strings will be deleted from the buffer after the safety message `Delete the text found?` has been answered with `Y` or [global replace].

Like the target string, the replacement string may consist of normal text or also include the special functions called by the function keys.

4.3.5 Using text from the found string

The found string may be repeated in the replace string simply by pressing the [found string] key. For example, to replace all occurrences of micro with microcomputer the command would be:

[global replace]
`micro`

[replace by]

[found string]
`computer`

[RETURN]

The whole find string `micro` is incorporated into the replace string.

Selected characters from an ambiguous string specification may also be used in the replacement string by quoting their position in the ambiguous field:

[global replace]

[ALPHA]

[ALPHA]

[ALPHA]
`abc`

[replace by]

[found string]
`0abc`

In this example the found string refers to the ambiguous field [ALPHA] [ALPHA] [ALPHA] The 0 indicates that the character in the first position (0) is the one to be incorporated in the replacement string. The characters in the second and third positions are thrown away. If, for example, the third character was also needed, then the line:

[global replace]

[ALPHA]

[ALPHA]

[ALPHA]
abc

[replace by]

[found string]
0

[found string]
2abc

would capture it.

Therefore, the parts of the found string which can be used in the replacement string are those that were specified by ambiguous character or string specifications in the target string. To identify which parts of the found string to use, the ambiguous elements are numbered according to their position in the target string, from left to right, starting at 0.

For example:

To bracket words (one or more alphabetic/numeric characters, preceded and followed by anything else) the syntax would be:

[most items] [alpha-numeric] [replace by] ([found string])

or

[most items] [alpha-numeric] [replace by] ([found string] 0)

To remove trailing spaces from lines the syntax would be:

[most items] *space* [newline] [replace by] [newline]

To put a plus sign (+) in front of all single-digit numbers (a single digit preceded by a space and followed by a space or common punctuation mark) the syntax would be:

[digit] [start set] *space*`,:;.` [end set] [replace by] +

5. Files

5.1 Saving and loading files

5.1.1 Saving text files

The [save file] key is used to save a TWIN document to the filing system. [save file] produces a bleep to warn you that a file of the same name in the current directory of the filing system may be about to be overwritten. This is in case [save file] was selected by mistake. There are a number of ways in which to give a filename to the document being saved.

(1) Type a filename then press [RETURN] in response to the prompt `Type filename to save:`

(2) Save the document using the same filename as last typed for save, load or insert by pressing [COPY] [RETURN] after `Type filename to save:`
N.B. After pressing [COPY] but before pressing [RETURN] the filename may be altered.

(3) Place `>filename` on the first line in the buffer and then press [RETURN] when prompted for the filename. This line must be less than 129 characters long but `>filename` need not be at the start of the line.

To save just part of the buffer in a file, mark one end of the part required (using [mark place]), move the cursor to the other end and then use [save file]. The prompt

`MARK TO CURSOR save:`

will be given instead of

`Type filename to save:`

There will be no bleep given.

5.1.2 Loading text and program files

The contents of the currently selected buffer may be replaced at any time using [load file]. The prompt `Type filename to load:` will appear and the filename followed by [RETURN] should be typed.

The file named should be an existing file on the current filing system. Any type of file currently supported by the filing system or language will load into TWIN. The wildcard * character may be used where the filing system allows. For example, [load file] `P*` instead of [load file] `Paintpr4` , but to ease the typing of load names which are complex due to directory routing, the special *path load command is available (see below).

Some types of file such as ARM object code will load correctly but will not be very readable, as they do not consist of many ASCII strings. They can be edited, however. Some programming languages convert keywords in their programs into ASCII codes above 127. Programming languages may provide commands to pass their tokenised programs to the editor, expanding them to readable plain text in the process. For example, ARM BBC BASIC V5. Refer to the language's documentation for guidance.

A file can also be loaded using methods (2) and (3) outlined in section 5.1.1.

5.1.3 Combining files

Files can be added to the text in the buffer rather than completely replacing it by using [insert file] . The file is inserted at the cursor position.

5.1.4 The *path load facility

[*path load] uses a text file containing directory information, giving TWIN the ability to search for files in places other than the current directory. To understand th e working of [*path load] imagine there exist three files, *filea, fileb* and *filec* which are regularly used. Creating a TWIN file called *path* in your currently selected directory similar to the example below will then allow TWIN to load any one of these files without the need to move out of the currently selected directory, or give a full file path name in response to [load file]

RETURN
```
$.arm.library
```

RETURN
```
$.arm.peter.utils
```

RETURN
```
$.armdoc.current
```

RETURN

This example assumes *filea* has already been saved to $.armdoc.current, *fileb* to $.arm.peter.utils and *filec* to $.armdoc.current. The first line is blank thus allowing TWIN to also look in the currently selected directory.

Once the path file has been set up press *path load and enter the filename in response to the prompt:

```
Type filename to load (via *path):
```

When the file has been found it will be displayed and the full path name will be placed in the file name position on the status line. If the file path cannot be found, the file system's error message will report `Not Found` . If, on the other hand, one of the three files cannot be found, then TWIN will report `File not Found` .

Names and delimiters can be built up as required. Use only |, ASCII 10 or ASCII 13 as delimiters. Using | (which can be read as or), the example path file would look like this:

```
|$.arm.library|$.arm.peter.utils|$.armdoc.current
```

There is still the null name before the delimiter, so the current directory would be searched first. The directory name(s) given can be relative to the current directory. The *path file must not be longer than 1024 bytes.

5.2 Printing files

TWIN will print the contents of the currently selected buffer directly to a printer, and the output will simultaneously appear on the screen, scrolling at a rate determined by the printer's baud rate or the size of the printer's internal buffer or both. If a single mark is in the text, the printed output will consist only of that text which is between the mark and the cursor. If two markers exist, the output will be the text between the marks.

TWIN normally makes no attempt to format lines and consequently words tend to become split at the right-hand edge of screen lines and printed lines. Word splits tend not to occur very often in source code where the average line length is less than 80 columns, but for sections of explanatory text or for documents word splitting is not acceptable. [format] will act on a paragraph of text and remove any split words by selectively replacing some spaces with a new-line character. The new line is initially placed after the 80th character, but if this is not a suitable point to break the text, a search is made backwards until a space is found. The paragraph is taken as the text between the cursor position and the next new line which is followed by a character, a space, full stop, comma, semi-colon or colon.

TWIN cannot right justify the text.

6. Appendix A

6.1 Concurrency: a worked example

6.1.1 TWIN and AAsm, the ARM assembler

Switch on the computer and set the time.

The ARM prompt appears.

Load TWIN from the logged on drive by typing `TWIN`

Type the following into TWIN

```
; -> test
ORG &1000
SWI 1
= "Hello World",10,13,0
SWI 17
END
```

This short piece of text is an AAsm file, now residing in TWIN's buffer 0. The source file can be saved using the name taken from the ; -> line at the top of the file using [save file] [RETURN].

Next, obtain a second buffer by pressing [toggle window]. A second, blank, window is now on the screen. AAsm can be called from this window by pressing [task status]. This brings the prompt `Command:` on to the status line of buffer 1. If AAsm is not in the currently selected directory or library, the full pathname must now be given. However, if AAsm does reside there then the response should be:

```
AAsm
```

AAsm will load and in the window of buffer 1 you will see:

```
ARM stand alone Macro Assembler Version x.xx
Entering interactive mode
Action:
```

The status line will show:

```
1 Task running in this window
```

The response to the prompts should be:

```
Action: ASM
Source file name: test
Code file name: testc
```

Whereupon the file test will be assembled:

```
Pass 1
Pass 2
Assembly complete
No errors found
Action:
```

Both passes ran, with no errors reported.

The program can now be executed by leaving AAsm:

```
Action: quit
```

which ends the task and leaves buffer 1 as a TWIN buffer, rather than a buffer shared by AAsm. The message `Task running in this window` is removed from the status line. To run *testc* from TWIN, it is made another Task:

Press [task status]

In response to the prompt `Command:` type `*testc` . The task will run, printing "Hello World" to the buffer and causing the status line to briefly report:

```
1 Task running in this window
```

before the task ends and control passes back to TWIN.

6.2 Using a task

The commands needed to recompile a Modula–2 environment, for example, would be:

`m2dep`	modula 2 dependency list
`m2make`	use dependency list to make m2comp, list of things to compile
`m2comp`	compile all programs

To build a command file to create this sequence would require:

(1) clear text from the command file

(2) type in the list activities

(3) save it as with an appropriate filename

Execute the command file as a task by pressing:

[task status] *filename*

Now files can be edited in the usual way in the other window; the text in the task window being updated almost as quickly as output arrives. As errors appear, the relevant source file can be loaded and [next line] used to move to the error line in the source.

7. Appendix B

7.1 Error messages

TWIN always reports detected errors and the command which is in error is not obeyed. The text in the buffer will be left unchanged but the editor may clear marks or move the text cursor to the start of the buffer or screen line.

`Bad mark number`

You have tried to set more than 2 marks or have not set the right number of marks (with [mark place]) for use with this command, for example you are trying to delete a block with 2 or 0 marks set rather than 1. The marks will be cleared.

`Bad number`

You have typed a number outside the allowed range of letters rather than digits, in response to an editor prompt for a number. Note that numbers are always decimal - you cannot use & to specify a hexadecimal number.

`Bad use of stored name`

Using [insert file] you have either pressed [RETURN] or [COPY] [RETURN] in response to the prompt for the file to insert, instead of supplying a filename.

`Line not found`

You have specified a destination line with [goto] which is beyond the end of the file. (To count the number of lines in the buffer, use [global replace] [newline] [RETURN].)

`No infile name found`

When using the load, save or insert file commands you have responded to the prompt for filename by pressing [COPY] or [RETURN] without previously typing a filename. If you did type a filename before pressing [RETURN], the file is either missing or an invalid name was used.

`No previous string`

After using the [find and replace] or [global replace] keys you have responded to the prompt by pressing [RETURN] but have not previously specified a find-and-replace string for that command. (Note that the string previously used with [find and replace] is available for use with [global replace] or vice versa, by pressing [SHIFT] [COPY].)

`No room`

There is not sufficient room in the text buffer for the file that you have tried to load or insert or the text buffer is full so that there is no room left for you to type more text. To release more memory, the contents of a buffer can be saved and the buffer closed.

`Insufficient memory to (re)start TWIN at this address`

This error message is most likely to occur when you have forgotten that the language in use is running under TWIN and attempts to edit the program. In these circumstances the request to load TWIN (or edit a BASIC program) meets with system disapproval, since the TWIN already in memory is seen as an obstacle. To recover from this error, the language can be reloaded, and commands such as >OLD and >LIST (for BBC BASIC) issued to obtain any missing program.

`Relative move out of file`

On a goto command, the forwards or backwards jump specified would take the cursor out of the file.

`No more lines in other window`

[next line] has been pressed, but there are no more error lines specified for TWIN to go to.

`Unsplittable line`

A line has been found which has more than 78 characters, none of which are spaces or new-line symbols.

`Invalid option to continue prompt`

The `continue` prompt has been answered by an illegal response.

8. Appendix C

8.1 Recovering lost text

Text recovery is not guaranteed in any circumstance: there is no OLD TEXT command. Text may be lost in a number of ways, and the loss can be fatal.

Fatal loss
Deleting a large block of the text by mistake using [delete block]. After a block delete, the text closes up and the block no longer exists in RAM memory.

Partial or fatal loss
Overwriting a file by another file using [load file]. If the new file is smaller in size than the one overwritten, some of the old file will still be present in RAM memory, otherwise the loss is fatal.

By pressing [BREAK] or [CTRL]-[BREAK]. The text is still present in RAM memory, although TWIN cannot reconstruct its pointers to reclaim it.

Partial loss
If text is lost and its reconstruction cannot be accomplished by re-typing a back-up file, the following action may be partially or totally successful.

(1) Note the starting point of TWIN by selecting TWIN's 'T' mode.

(2) Leave TWIN using [command]

(3) SAVE memory from the start of TWIN to the physical end of RAM. This should be done in two or more sections such as `TEMP1` and `TEMP2`, for example: `*SAVE filename twin start address arbitrary length`. The arbitrary length should not be so large that it cannot be loaded back into TWIN.

(4) Re-enter TWIN and [load file] `TEMP1`

(5) Use [find and replace] to search for the lost text. If you find it in `TEMP1` or `TEMP2` you can use [delete block] selectively to remove everything except the text which is needed.

However, TWIN will need to search many megabytes of memory and there is no guarantee that the lost text will be found. Always make frequent back-up copies of documents and files to avoid the consequences of losing text.

9. Appendix D

9.1 TWIN function keys

[goto]

TWIN computes line numbers by counting new-line characters and three different goto jumps are allowed:
Move to a given line-number. For example, [goto] 765
Move to a mark by using m1 or m2. For example, [goto] m1
Move by a number of characters by +<n> or –<n>. For example, [goto] -50

[show newlines]

The new lines can be shown as a special character, a highlighted left-arrow, so that they can be seen clearly. Press this key to either reveal or hide the new lines.

[toggle window]

Toggle between windows. If the second window is not already on screen then the screen is split and the second window is opened. The selected window's status line is highlighted.

[connect buffer]

Connect a window to a different buffer from the one currently using it, by specifying the buffer's number. There are 10 buffers, called 0 to 9. Alternatively, the plus and minus sign (+ and –) will increment or decrement the buffer attached to the current window. Any response other than 0-9, + or – lists the buffer directory on the screen. Buffers cannot be connected while a pushed buffer is present.

[command]

Commands to the computer's operating system can be given following the TWIN* prompt which appears on the screen. For example: obtain a CATalogue, run BBC BASIC, and so on. Afterwards, re-enter TWIN by pressing [RETURN].

[insert/over]

Toggles between insert and over modes for the particular buffer in use.

[expand window]

Enlarges the current window by one line, if there is enough room. The other window is reduced by one line. Cursor positions in both windows remain on screen.

[load file]

Loads a text file into the current buffer. If modified text will be erased in the process, TWIN asks you to confirm this.

[insert file]

A text file will be inserted at the current cursor.

[close window]

Closes current window (the text is not harmed) and displays the other alternate window, which may or may not have been on the screen.

[*path load]

This loads a text file like [f2] but uses a file path such as *m2path*, *cpath*, or *twinpath* provided by the filing system. The *path file must not be longer than 1024 bytes.

[save file]

There are a number of variations of SAVE:

- Save all text
- Save text from a mark (m1) to the cursor
- Save using the current filename
- Save using a name supplied in the text
- TWIN date-stamps the file, beeps to remind you that save has been pressed. [ESCAPE] cancels the save if it isn't wanted.

[new margins]

Controls the scrolling action of a window.

- B to set the bottom scroll margin to cursor line
- R to remove the top and bottom scroll margins
- T to set the top scroll margin to cursor line

A mode change or window command resets the margins to their default value.

[next line]

TWIN can look for occurrences of line or LINE in the error messages of a compiler's output and use the information to go to that line in the source file's buffer.

[find and replace]

Begin a find or a selective find and replace.

[exit to]

After checking if there is any altered text to be saved, TWIN will exit to a named language (or to the operating system if just [RETURN] is pressed).

[global replace]

Performs an automatic count of occurrences (with no replace) or a fully-fledged, non-selective find and replace. The whole text or just a block of text may be specified.

[set mode]

A number of screen modes are available:

- 0, 3: select mode 0 or 3 (32 lines or 25 lines)
- D: Descriptive: shows all keys and details commands
- K: Key legends: shows the function keys names
- T: Time: shows the time on the top line.

[enter character]

Inserts the character from the keyboard directly, or allows the character's decimal code to be typed.

[pop]

A pushed buffer will be recovered from its stack. See [push].

[mark place]

Put a marker into the text at the cursor position. The status line shows how many (0,1,2) marks are being used in a buffer. Each buffer and each pushed buffer may have up to two marks.

[clear marks]

All place marks in the current buffer are cleared.

[format]

Format lines to a width of 78 characters or less. The text from the line the cursor is on to the next new line followed by a character ASCII 0-31 or 128-255 is formatted to remove word splits.

[push]

Push the current contents of the buffer into stack memory. Only buffers currently selected may be pushed, so two (represented by each screen window) may be pushed at any one time. A new file may be loaded into the buffer vacated by the push. The newly loaded file may also be pushed. In this way files can be loaded and viewed without having to open a new buffer to receive them. TWIN can push any number of files, subject only to limitations of memory.

[copy block]

Text between two marks is copied to the cursor. If there are no marks in the current buffer, text can be copied from the alternate buffer (provided it has two marks set in it).

[move block]

Text between two marked places is moved to the current cursor position which should be outside the marked area. The marks are then cleared.

[send mail]

On a network fileserver system, TWIN attempts to append the text to a file using $.EcoMail.Out.*name* If that fails, it tries $.*name*.MailBox.

[print text]

The whole, or marked, text is printed out.

[delete block]

The text between the cursor and the marked place is deleted. The mark is then cleared.

[toggle LF <-> CR]

All carriage return |M and line feed |J symbols currently in the file are exchanged.

[auto bottom]

Toggle between task-linked and task-unlinked states. With task-linked, whenever the task window is redrawn, the cursor is moved to the end of the buffer. With task-unlinked, no change is made to the cursor positon.

[bytes free]

The number of bytes free in TWIN's section of RAM. It is possible to adjust TWIN's LOAD ADDRESS to obtain more space.

[clear text]

All text in the buffer is deleted.

[task status]

This key is used both to start and to finish a single background task. It will also reveal to you whether a task is currently running. The output of the task is appended to the end of the current buffer. While it is running editing can be done on the appended part of the current file, or in the whole part of a file in a different buffer. Keyboard characters are sent to the task.

[task bottom]

Moves the cursor position to the end of the task window.

[SHIFT]-[TAB]

Toggle the tab mode.

[CTRL]-[TAB]

Expand tabs in the text.

[SHIFT]-[COPY]

Toggle edit mode: toggles between insert and overtype mode and copy editing. Copy editing allows the use of user-defined soft keys and allows all characters except the escape character to be put into the text.

[CTRL]-[COPY]

Deletes from start of current screen line to next new line.

[CTRL]-[SHIFT] [CURSOR DOWN]

One page forward on both windows.

[CTRL]-[SHIFT] [CURSOR UP]

One page backward on both windows.

10. Appendix E

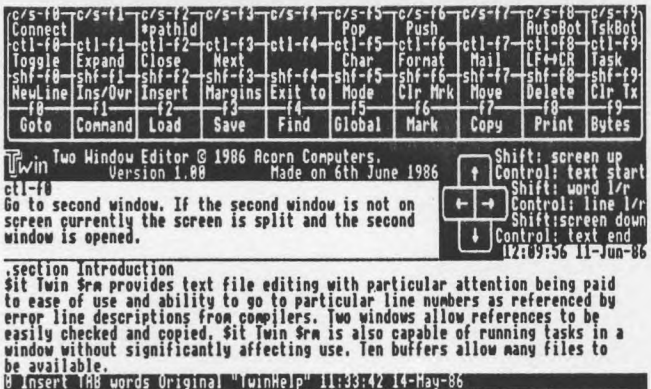
10.1 The help screen

The screen print-out shows the five areas of the screen. The top portion, in reverse video, is the D mode help panel. The function-key legends are slightly more concise than their equivalents in the K mode help panel, but the D type panel has key-prompted help messages. In the screen shown, [toggle window] had just been pressed, so the help explanation of [toggle window] is on the screen.

Beneath the help panel is a window showing a date-stamped file. Its name is "TwinHelp". This window is connected to buffer 0, and the whole status line is in reverse video, indicating that this window is the current window.

Beneath window 0, and occupying the last 8 lines of the screen, is the second TWIN window, connected to buffer 1. It shows an assembler source file "Editor". Both "TwinHelp" and "Editor" are marked as original files, so their memory image is the same as their filing system image in both cases. The date stamp for each file reveals the time and date when they were last altered.

As shown in the help panel, the screen dump was made at 12.09 am on 11th June.



11. Appendix F

11.1 Screen modes

Change mode. Only BBC screen modes 0 and 3 may be used (80*32 and 80*25 respectively): 0 and 3 select either of these operating modes.

When TWIN starts up it will select either 32-line mode or 25-line mode. It does this with a BYTE call (see *Executive ROM manual*) with R0=161, R1=8, R2=0: if the bottom two bits of R2 are zero then mode 0 is used, otherwise mode 3 is used. This means that a standard BBC machine or B+ will start in mode 0, a Master will start in 0 or 3 (the call returns state from CMOS RAM: change it by using Edit on the IO processor: TWIN starts up in Edit mode) and the TWIN on an IBM PC starts in mode 3 since that operating system specifically returns 3.

12. Appendix G

12.1 Tasks and events

The swapping between the task and the main TWIN functionality is performed by events from the BBC I/O system: consequently the task is denied the use of events. TWIN swaps the environments created by Control and SetEnv as swapping between itself and the task occurs: thus escape codes will be sent to the task as will errors.

When a task is running the CSD or filing system should not be changed because this could confuse the task.

13. Appendix H

13.1 Moving TWIN

The release version of TWIN runs at &1D0000 in 2 or 4-Mbyte ARM, &D0000 in 1-Mbyte ARMs, or &10000 in 1/4-Mbyte ARMs, the Executive ROM performing the translation automatically. Note that the Executive cannot deal with the &3D0000 address in a 2-Mbyte machine. This gives either 3/4 or 1/4 of the machine to background tasks. The load and execute addresses of TWIN can be altered by using a program on the release disc called MOVEFILE. For the ARM* command line the instruction:

```
movefile twin address
```

should be given. The address given should be the new address for TWIN, for example, &2D0000. The address range accepted by the movefile program is &1000 – &3FFFFFF but a high-memory address should be chosen with care since movefile doesn't check whether there is sufficient physical RAM to hold the TWIN program.

14. Appendix I

14.1 TWIN called from ARM BASIC

If the language AB is running on the ARM, the command TWIN or EDIT will cause the following actions to take place:

TWIN will be loaded into memory from the filing system.
The AB program will be copied into memory, the copy being an ASCII file version of the program.
TWIN will be informed of the location of the ASCII file, and will start up with it displayed in buffer 0.

Return to AB is by the normal exit to.

If the start and end address of the ASCII file is known, it can in fact be loaded directly into TWIN by a command from the ARM* prompt. The syntax to do this is:

```
TWIN @ssssssss,ffffffff; filename
```

where s is the hexadecimal start address and f is the hexadecimal end address. The filename is the name the ASCII file will have as an AB program.

Acorn

The choice of experience.