Specification for

CARETAKER

The BASIC utility ROM for the BBC micro

COMPUTER CONCEPTS

CARETAKER

the BASIC utility for the BBC Microcomputer

You've just spent all evening perfecting a large and impressive program. You type RUN and wait expectantly. Nothing happens. Muttering under your breath, you press BREAK and type OLD. "Bad program", it says ....

Now at this point, depending on your approach to life, you will either smile and reload the program which you had saved prior to running it (full marks!), give up and start the evening's work all over again, or set about the task of trying to recover what you can.

With CARETAKER in your machine, however, all is not lost, as it can recover a crashed or corrupted program with a single command. It can also search your program for a given string, say a word or variable name, which can include 'wildcard' characters, and either replace all occurrences automatically or let you decide in each case. CARETAKER allows you to enter BASIC keywords at a single key stroke, and to keep track of the amount of memory space left. You can renumber parts of a program without affecting the rest, save a section of it onto cassette or disc, merge two programs together, or insert one program into another at a chosen point. BASIC programs can be squashed to save space, expanded to make them easier to read and edit, or moved around in memory.

Everything is designed to make life easier for the programmer and to leave more time for writing and enjoying your programs. For example, the TABSTOPS facility is a most useful aid, particularly when using the BASIC Assembler, making it simple to produce legible programs that are easier to de-bug. Some commands can be used within a BASIC program, making them even more powerful. In this context, LVAR gives you the option to see which procedures or functions have been used, and to display variables with their current value while the program is actually running.

CARETAKER commands are described more fully on the following pages.

*CURSOR (ON/OFF)

Used as a direct command or within your BASIC program, it will remove the cursor from the screen or turn it on again.

*EXCHANGE <old> <new> G/S (<length>)

A search and replace facility for strings in BASIC programs. Using the BBC Micro's string escape character (¦) you can also find and replace BASIC keywords, while a special wildcard facility allows sophisticated string searches to be made.

*EXPAND (<start>) (<end>)

Lists a BASIC program between lines <start> and <end> with spaces before keywords, and with multi-statement lines separated. This makes it possible to read or edit a program which has been squashed to save memory, without having to increase its size again.

*INSERT <fsp> (< line no >)

Will insert a BASIC program on file into the BASIC program in memory without checking line numbers. This is an easy way to add standard routines to your BASIC program. If<line no>is omitted,<fsp>will be added to the end of the BASIC program and can be safely renumbered.

*KEYLOAD (<fsp>)

Re-loads the function key definitions which have previously been saved using KEYSAVE. If <fsp> is not given, the command will load a file called 'keez'.

*KEYSAVE (<fsp>)

Saves the function key definitions using the default filename 'keez' if<fsp>is not given.

*LVAR (F) (I) (S) (A) (P)

Lists all variables, functions and procedures that have been used in a BASIC program. The options allow you to choose any combination of types of variables: floating point, integer, or string, with or without arrays, together with procedures and functions.

*MERGE <fsp> (<fsp>)

Merges BASIC program (s) on cassette or disc with the program in memory. Any lines in memory with the same numbers as those on file will be overwritten.

*MOVE <address>

Will move a BASIC program in memory from its present position to the address specified, provided that there is sufficient room to accommodate it and the address given is legal. BASIC pointers are automatically reset.

*NORMALKEY

Turns off singlekey entry of BASIC keywords.

*NOTAB

Turns off the tabulation function *TABSTOPS.

*PARTSAVE <fsp> (<start>) (<end>)

Saves a part of BASIC program specified by <start> and <end> as a complete BASIC program called<fsp>. If either<start>or<end>are omitted then it takes defaults.

*RENUMBER (<1st> <inc> <start> <end>)

Allows you to renumber part of a BASIC program with complete safety. This command makes it easy to put the program in a different order and to move sections that have been inserted. All parameters are optional, and no BASIC program space is used.

*RETRIEVE (<bytes>)

Attempts to retrieve a BASIC program for<bytes>. If<bytes>is not specified then it will chop the BASIC program at the point where it finds an error.

*SINGLEKEY

Enables the user to input an entire BASIC keyword by pressing the TAB key and another. Labels are provided to stick onto the front of the keys. To use the TAB key as normal the SHIFT and TAB keys have to be pressed.

*SQUASH (S) (R) (M)

Will reduce the size of a BASIC program to save memory. The options are for spaces, REM statements, and multiple statements lines.

*STATUS

Lists PAGE, TOP, LOMEM, HIMEM and 'vartop', together with the current program length and the amount of memory available between 'vartop' and HIMEM.

*TABSTOPS (<columns ...>)

Allows you to use 8 tabulation columns on the VDU, and has defaults of 9, 18, 27, 36, 45, 54, 63 and 72. If different tab settings are defined, they are stored in the resident integer variables T% and U%. *NOTAB disables this function.

Order as CARETAKER ROM   Price : £33.35 including P&P and VAT.
Available from your local BBC Micro software dealer, or direct from:

Computer Concepts, 16 Wayside, Chipperfield, HERTS WD4 9JJ (09277) 69727

As from August 1984, our Address will be: Gaddesden Place, Hemel Hempstead, HERTS HP2 6EX (0442) 63933

COMPUTER CONCEPTS

USER MANUAL

Caretaker

The BASIC utility ROM for the BBC micro

COMPUTER CONCEPTS

Additions to CARETAKER's manual

We have noted that some games, 'hang' when CARETAKER's SINGLEKEY entry is on. This is because these games overwrite page D (&D00-&DFF), which is used for extended vectors. To overcome this problem type: *NORMALKEY or *NOR. before running the game, or include the above command within a !BOOT file.

Two additional error messages:

152 Not in BASIC - the BASIC ROM is not selected for related commands

153 Bad Hex - Hexadecimal parameter contains an illegal character

The TABSTOP function is disabled as default and not enabled as stated in the manual.

In EXCHANGE (page 7) the last paragraph should read as:

.... e.g. (¦241¦) will find "PRINT" ...

When fitting the special stickers for SINGLE-KEY entry make sure the key fronts are clean and free from dust. Then, with the aid of a pair of tweezers, remove the stickers and fix them to the appropriate keys. The 'Keywords' sticker should be fixed to the 'TAB' key and the others should be fixed as shown on page 21 of the manual.

CONTENTS

IT IS VITAL THAT THE REGISTRATION CARD SUPPLIED WITH CARETAKER IS RETURNED TO US, COMPLETED WITH YOUR NAME AND ADDRESS. THE CARD IS POSTAGE PAID FOR THE U.K. ONLY REGISTERED OWNERS WILL BE ABLE TO OBTAIN UPDATED VERSIONS, TECHNICAL HELP ETC. THE SERIAL NUMBER OF THIS PACKAGE SHOULD BE PRINTED INSIDE THIS MANUAL, AND THIS MUST BE QUOTED IN ALL CORRESPONDENCE WITH REGARD TO CARETAKER. IF FOR ANY REASON A REGISTRATION CARD IS NOT SUPPLIED, PLEASE CONTACT COMPUTER CONCEPTS.

CARETAKER
the BASIC utility

INTRODUCTION

CARETAKER is a collection of utilities designed to be used with BASIC programs. Because it is in Read Only Memory, it uses no BASIC program space and yet its extremely powerful routines are always available. They include a search and replace facility; insert and merge routines; single key entry of keywords; squash and expanded listing; memory status display; part renumbering and moving; part saving; together with a routine that will attempt to recover a corrupted program when all seems lost!

CARETAKER commands are used in a similar way to Operating System (OS) commands by entering an asterisk '*' followed by the command name:

```
*STATUS <RETURN>
```

Most of them are designed to help you to work on your BASIC program. Some can also be used within a BASIC program itself, and like all OS commands should be the last or only statement on a line. The command name can be entered in any combination of upper and lower case letters, so that *status, *STATUS and *sTAtus will all have the same effect.

A command may usually be abbreviated to two or three letters followed by a full stop (e.g. *ST. for *STATUS), providing that this minimum abbreviation is sufficient to distinguish it from any other commands with a similar name. It is always safer to use the whole word if you are in doubt.

Most commands are qualified by one or more parameters which may either be optional or compulsory.

THE HELP MENU

A listing of all the commands that CARETAKER understands, together with their syntax, can be displayed by typing

```
*HELP CARETAKER <RETURN>
```

or any abbreviation down to

```
*h.c. <RETURN> :
```

```
CARETAKER 1.00
  CURSOR (ON/OFF)
  EXCHANGE <old> <new> G/S (<length>)
  EXPAND (<start>) (<end>)
  INSERT <fsp> (<lineno>)
  KEYLOAD (<fsp>)
  KEYSAVE (<fsp>)
  LVAR (F)(I)(S)(A)(P)
  MERGE <fsp> (<fsp>)...
  MOVE <address>
  NORMALKEY
  NOTAB
  PARTSAVE <fsp> (<start>) (<end>)
  RENUMBER (<1st> <inc> <start> <end>)
  RETRIEVE (<bytes>)
  SINGLEKEY
  SQUASH (S)(R)(M)
  STATUS
  TABSTOPS (<columns...>)
```

Parameters are shown within angle brackets - <>

<fsp> means that a file specification or filename is expected

CARETAKER
the BASIC utility

INTRODUCTION

CARETAKER is a collection of utilities designed to be used with BASIC programs. Because it is in Read Only Memory, it uses no BASIC program space and yet its extremely powerful routines are always available. They include a search and replace facility; insert and merge routines; single key entry of keywords; squash and expanded listing; memory status display; part renumbering and moving; part saving; together with a routine that will attempt to recover a corrupted program when all seems lost!

CARETAKER commands are used in a similar way to Operating System (OS) commands by entering an asterisk '*' followed by the command name:

*STATUS <RETURN>

Most of them are designed to help you to work on your BASIC program. Some can also be used within a BASIC program itself, and like all OS commands should be the last or only statement on a line. The command name can be entered in any combination of upper and lower case letters, so that *status, *STATUS and *sTAtus will all have the same effect.

A command may usually be abbreviated to two or three letters followed by a full stop (e.g. *ST. for *STATUS), providing that this minimum abbreviation is sufficient to distinguish it from any other commands with a similar name. It is always safer to use the whole word if you are in doubt.

Most commands are qualified by one or more parameters which may either be optional or compulsory.

THE HELP MENU

A listing of all the commands that CARETAKER understands, together with their syntax, can be displayed by typing

```
*HELP CARETAKER <RETURN>
```

or any abbreviation down to

```
*h.c. <RETURN> :
```

```
CARETAKER 1.00
  CURSOR (ON/OFF)
  EXCHANGE <old> <new> G/S (<length>)
  EXPAND (<start>) (<end>)
  INSERT <fsp> (<lineno>)
  KEYLOAD (<fsp>)
  KEYSAVE (<fsp>)
  LVAR (F)(I)(S)(A)(P)
  MERGE <fsp> (<fsp>)...
  MOVE <address>
  NORMALKEY
  NOTAB
  PARTSAVE <fsp> (<start>) (<end>)
  RENUMBER (<1st> <inc> <start> <end>)
  RETRIEVE (<bytes>)
  SINGLEKEY
  SQUASH (S)(R)(M)
  STATUS
  TABSTOPS (<columns...>)
```

Parameters are shown within angle brackets - <>

<fsp> means that a file specification or filename is expected

If a command is entered without the required parameters or with parameters that CARETAKER does not understand, then the correct syntax will be shown:

```
*MERGE <RETURN>

Syntax: *MERGE <fsp> (<fsp>)...
```

showing that at least one filename needs to be given. The round brackets show that the use of the parameter, letter or word they enclose is optional. In this case, for example, further filenames can be given.

Where more than one parameter is required, they must be separated by either a space or a comma.

Most parameters will be self-explanatory and are dealt with under the particular command name. Although full details of all parameters that CARETAKER uses are given in an appendix at the end, it may be helpful at this stage to note that the <start> and <end> parameters take default values in the same way as BASIC's LIST statement. Thus, where <start> and <end> are called for, they are both optional and may be used as follows:

no parameter	the command operates on the whole program
<start>	means the line 'start' only
<start>,	means from the line 'start' to the end of the program
,<end>	means from the beginning of the program to the line 'end'.

When a atring parameter is called for, you must use double quotes if a space or non-alphanumeric character is to be included within the string.

A space or non-alphabetic character is always required between the command word and the first parameter.

If there are several ROMs in your BBC computer, there may be a command name in CARETAKER which is the same as a command name in another ROM. If this is the case, then the letter "C" may be added as a prefix to the command name to uniquely indicate that the CARETAKER utility is the one required:

*CKEYSAVE is the same command as *KEYSAVE

CARETAKER commands are arranged in alphabetical order as in the index and help menu. The following pages deal with each command, after which there are appendices with details of error messages, parameters and a list of BASIC keywords and their tokens.

CURSOR (ON/OFF)

Enables or disables the cursor

Example

```
*CURSOR OFF <RETURN>
```

will switch off the cursor.

```
*CURSOR <RETURN>
```

without a parameter, or with the parameter 'ON', will switch on the cursor.

Note

This command can either be typed directly on the keyboard or embedded within your BASIC program.

EXCHANGE <old> <new> G/S (<length>)

Provides a search and replace facility when editing BASIC programs that operates either globally or selectively (G/S), together with a special wildcard feature.

Example

```
*EXCHANGE big large G <RETURN>
```

will replace each occurence of the letters b-i-g with l-a-r-g-e, listing the line numbers in which the exchange has been made. This means that wherever the letters b-i-g appear, they will be replaced. Thus 'bigger' will become 'largeger' and 'Abigail' will become 'Alargeail'. If there is any danger of this sort of mistake, it is better to choose the selective option:

```
*EXCHANGE big large S <RETURN>
```

will show on the screen each line in which the letters b-i-g occur and offer the user the option to replace the letters with l-a-r-g-e.

In this case, pressing 'N' will move on to the next occurence without an exchange; pressing any other key will make the exchange.

If you only wish to locate a particular string, entering

```
*EXCHANGE big big S <RETURN>
```

will allow you to look at all the occurences of b-i-g without affecting them whatever action you take.

Keywords may be found by enclosing the appropriate decimal token number with the string escape character. e.g. (241) will find "PRINT". The numbers are listed in two appendices at the end of the manual.

Alphanumeric search

The final optional parameter, <length>, allows you to use two wildcards, '#' and '*', in the search string <old>. This enables searches to be made for words where the exact spelling is uncertain or where there are several similar forms such as 'part1', 'part2', 'part3'.

If you are using wildcards, the search is limited to alphanumeric strings and keyword tokens should not be included. (An alphanumeric string is one that only contains numbers, letters or the underline and pound characters).

<length> is used to specify the number of characters in the string being searched for. For example

```
*EXCHANGE big large S 3 <RETURN>
```

will only replace b-i-g when it is on its own. Any occurence in a string which is not three characters long will be ignored.

The hash symbol '#' means "any character in this position":

```
*EXCHANGE part# event S 5 <RETURN>
```

will find 'part1', 'part2' or any other 5-character string containing p-a-r-t followed by any single character, and offer the option to replace it with 'event'.

More than one hash wildcard may be used in the search string.

The asterisk '*' means "all the rest of the name up to the specified length", and can be placed at the beginning and/or the end of the search string:

```
*EXCHANGE te* section S 5 <RETURN>
```

will offer the option to replace with 'section' any 5-character string starting with 'te'.

```
*EXCHANGE *te section S 5 <RETURN>
```

will do the same for any word ending in 'te', and

```
*EXCHANGE *te* section S 5 <RETURN>
```

will offer any 5-character string for replacement that contains the letters 'te'. This is different from

```
*EXCHANGE te section S 5 <RETURN>
```

which replaces only the letters t-e with 'section' whereas the previous example will replace the whole 5-character string.

Notes

<length> should be given as a decimal number, and cannot be less than the number of characters/wildcards in the search string.

<old> and <new> are string parameters with a maximum length of 127 bytes.

EXPAND (< start >) (< end >)

Displays a program in a form that makes it easier to read and also enables it to be edited, without taking up more memory. The two optional parameters are for line numbers.

Example

```
*EXPAND 10 20 <RETURN>
```

The effect of this command on the lines between 10 and 20 inclusive depends on what has already been done to the program using CARETAKER's SQUASH routine or any other such utility. Assuming that the maximum contraction possible has been effected, the routine will insert a space before each BASIC keyword and break multi-statement lines down into separate statements, placing them below the first one in the line. Thus the BASIC line:

```
10Z=0:REPEAT:PRINTTAB(Z*S)NAME$(C);:Z=Z+1:UNTILZ=30ORC+Z=X:IFZ=30PRINT
```

will become:

```
10 Z=0
 : REPEAT
 : PRINT TAB(Z*S)NAME$(C);
 : Z=Z+1
 : UNTILZ=30 ORC+Z=X
 : IF Z=30 PRINT
```

Note

If you have a lot of editing to do it is usually better to go back to the original program and re-squash it afterwards. See SQUASH for more details.

INSERT <fsp> (<lineno>)

Enables BASIC routines on disc or tape to be added to the program in memory.

Example

```
*INSERT prog2 1860 <RETURN>
```

will look for a BASIC program called "prog2" in the currently selected filing system and insert it after line 1860 in the BASIC program in memory.

```
*INSERT prog2 <RETURN>
```

without the second parameter will cause "prog2" to be added to the end of the existing program.

The file <fsp> will be loaded into memory at the location given without taking its line numbers into account.

Note

Care must be taken to ensure that the inserted section does not contain any line numbers that are referenced by GOTO, GOSUB or THEN statements in the original program. Regardless of where the insertion is made it is always safe if the line numbers being inserted are completely different from any already in the program. Any general purpose routines that you might wish to insert are best numbered unusually, say 32001,2,3, which will ensure that any referenced lines do not clash when renumbering.

KEYLOAD (< fsp >)

Loads function key definitions directly into memory.

Example

```
*KEYLOAD MYKEYS <RETURN>
```

will load a file called "MYKEYS" from the currently selected filing system.

```
*KEYLOAD <RETURN>
```

without a parameter, will attempt to load using the default filename "keez".

Description

This command loads a data file, which has been created by *KEYSAVE, straight into the function key definition area at &B00.

The minimum abbreviation is *KEYL.

KEYSAVE (<fsp>)

Saves function key definitions.

Example

```
*KEYSAVE MYKEYS <RETURN>
```

will save a file called MYKEYS to the currently selected filing system.

```
*KEYSAVE <RETURN>
```

without a parameter, will save using the default filename "keez".

Description

This command will save the function key definition area using either the default filename "keez" or a specified filename. This can then be reloaded at a later date, using *KEYLOAD.

Notes

If you use this command without a parameter make sure that you do not unintentionally overwrite an existing file already created with the filename "keez".

The minimum abbreviation is *KEYS.

LVAR (F) (I) (S) (A) (P)

Description

```
*LVAR <RETURN>
```

without parameters lists all variables, functions and procedures that have been used in the current BASIC program in memory.

The parameters may be used in any combination to select the following options:

F - floating point variables

I - integer variables

S - string variables

A - include arrays among variables selected above

P - Procedures and Functions

Any other letter will cause a Syntax error.

MERGE <fsp> (<fsp>)...

Merges a BASIC program on file with the program currently in memory.

Example

```
*MERGE PRNTOUT <RETURN>
```

will attempt to merge the program "PRNTOUT".

```
*MERGE PRNTOUT GPI <RETURN>
```

will attempt first to merge "PRNTOUT" and then "GPI".

Description

This command will attempt to load each file in turn and merge it with the BASIC file in memory. This means that any lines in memory which are the same as those on file will be overwritten.

Notes

If any program on file causes an error, *MERGE will terminate immediately giving the appropriate error message. *MERGE will, however, merge as many lines as it can before the error occurs.

MOVE < address >

Allows the current BASIC program to be moved to another location in memory.

Example

```
*MOVE 3000 <RETURN>
```

will attempt to move the current program at PAGE to location &3000 (hex). If sucessful, it will automatically alter PAGE, LOMEM and TOP to the appropriate values, making it unecessary to type 'END' after using *MOVE.

An error message will be issued if:

either there is not enough memory between <address> and HIMEM to accomodate the program

or the address specified does not lie between &E00 and HIMEM.

In either case no move will be made.

If you are moving a program down to &E00, don't forget to enter *TAPE to avoid the DFS corrupting your program.

NORMALKEY

This command turns off a normal CARETAKER facility which enables you to enter BASIC keywords with a single key stroke.

Description

When you switch on your computer the keyboard functions in the usual way, except that CARETAKER automatically enables you to enter BASIC keywords by pressing a key while TAB is pressed. In this mode, you can still use the TAB key for its original function by pressing SHIFT-TAB.

*NORMALKEY <RETURN> will disable single key entry, and the command can also be used within a BASIC program.

The following keys produce keywords while TAB is pressed; the same keys are used on the Acorn ELECTRON computer:

A	AUTO	B	RENUMBER	C	COLOUR	D	DRAW	E	ELSE
F	FOR	G	GOTO	H	DEG	I	INPUT	J	RAD
K	CHAIN	L	LIST	M	MODE	N	NEXT	O	OLD <RETURN>
P	PLOT	Q	LOCAL	R	RUN <RETURN>	S	STEP	T	THEN
U	UNTIL	V	VDU	W	RESTORE	X	PROC	Y	REPEAT
Z	END	,	LOAD	.	SAVE	/	PRINT		

In addition you can use:

1	*CAT <RETURN>	2	*HELP CARETAKER <RETURN>

Stickers are supplied to fix to each key showing its associated keyword.

NOTAB

Switches off the TABSTOPS function

Example

```
*NOTAB <RETURN>
```

See **TABSTOPS** for further details.

PARTSAVE <fsp> (<start>) (<end>)

Saves part of a BASIC program as a separate program called <fsp>.

Example

```
*PARTSAVE section 250 1000 <RETURN>
```

will save a file called "section", consisting of lines 250 to 1000 of the current program in memory.

Note

The parameters <start> and <end> may be used in the same way as BASIC's LIST statement as explained in the Introduction. Thus

```
*PARTSAVE section ,500 <RETURN>
```

will save a program under the name "section" comprising all the lines from the beginning of the program up to and including line 500.

RENUMBER (<1st> <inc> <start> <end>)

Renumbers parts of BASIC programs and moves the renumbered section as necessary.

Example

```
*RENUMBER 600 10 372 434 <RETURN>
```

will renumber all lines from 372 to 434 inclusive, making the first one 600, the next 610 and continuing in increments of 10. If the renumbered lines would overlap another part of the program, an error message 'Renumber overlaps' will be issued and the part will not be moved or renumbered.

The parameters <start> and <end> may be used in the same way as BASIC's LIST statement as explained in the Introduction.

```
*RENUMBER <RETURN>
```

without parameters, will renumber the entire program in increments of 10, starting from line 10, as does BASIC's RENUMBER. A major difference, however, is that CARETAKER'S routine uses no memory after the BASIC program to achieve this. This is a considerable advantage if working on large programs where BASIC is unable to renumber, but does result in a slightly slower routine.

RETRIEVE (<bytes>)

Restores a program that has become corrupted to a form whereby it can be listed and edited.

Example

```
*RETRIEVE 12000 <RETURN>
```

will attempt to reconstruct the 'damaged' program for at least 12000 bytes from PAGE.

```
*RETRIEVE <RETURN>
```

without a parameter, will retrieve as much as possible of the program until an error is found and then terminate it at that point.

Note

If you are not sure how long your program was before it became corrupted, then specify a large number of bytes, say 32000, as the parameter. This will piece together the entire contents of memory from PAGE to HIMEM. An earlier program in the computer's memory may have been longer that the one you wish to recover, in which case the routine may append lines from the earlier program to the current one. Even if these duplicate lines in the program you are recovering, you can renumber them using CARETAKER's routine and then delete unwanted lines.

If you find that you haven't recovered enough, you can try again with a larger number of bytes.

SINGLEKEY

Enables BASIC keywords to be entered by pressing a single key.

Example

```
*SINGLEKEY <RETURN>
```

will restore the normal CARETAKER facility to enter BASIC keywords by pressing a single key if this has been disabled using the *NORMALKEY command. The command can also be used within a BASIC program:

```
10 *SINGLEKEY <RETURN>
```

For reference, the keys that produce keywords while TAB is pressed are as follows:

A	AUTO	B	RENUMBER	C	COLOUR	D	DRAW	E	ELSE
F	FOR	G	GOTO	H	DEG	I	INPUT	J	RAD
K	CHAIN	L	LIST	M	MODE	N	NEXT	O	OLD <RETURN>
P	PLOT	Q	LOCAL	R	RUN <RETURN>	S	STEP	T	THEN
U	UNTIL	V	VDU	W	RESTORE	X	PROC	Y	REPEAT
Z	END	,	LOAD	.	SAVE	/	PRINT		
1	*CAT <RETURN>			2	*HELP CARETAKER <RETURN>				

Special stickers are supplied to fix to each key showing its associated keyword.

SQUASH (S) (R) (M)

Reduces the size of a BASIC program in order to save memory space.

Example

```
*SQUASH SRM <RETURN>
```

The three optional parameters are as follows:

S - all unecessary spaces will be removed.

R - everything following the keyword REM in a line will be deleted including the keyword.

M - multiple statement lines will be created up to the maximum allowable line length and blank lines will be deleted unless they are referenced elsewhere in the program.

The letters can be given in any order, but any other letter will cause a Syntax error message showing the correct usage.

```
*SQUASH <RETURN>
```

without parameters is the same as typing *SQUASH SRM <RETURN>.

A SQUASHed program can be listed in a readable form using EXPAND (q.v.).

Notes

The *SQUASH routine only detects Operating System (OS) commands when they are at the beginning of a line, after a colon in a multiple statement line, or following the keyword THEN. Hence care should be taken when OS commands are used in unconventional situations since *SQUASH may remove spaces or add multiple statement lines to them.

STATUS

Displays available memory and memory variables.

Example

```
*STATUS <RETURN>
```

will list the values of PAGE, TOP, LOMEM, HIMEM, and 'vartop', together with the current program length and Bytes free, the amount of memory left between 'vartop' and HIMEM.

Description

The function TOP and the variables PAGE, LOMEM and HIMEM are described in the User Guide. The values shown for these variables and 'vartop' are hexadecimal numbers. 'vartop' is the last memory location used for variable storage, and thus HIMEM-vartop gives a more accurate picture of the amount of memory left for program development than HIMEM-TOP. If, however, you wish to know how much memory is left between HIMEM and TOP, then you can clear the variable storage space by typing CLEAR <RETURN> before using *STATUS.

TABSTOPS (<columns...>)

This facility is made available by CARETAKER when the machine is switched on, and provides a tab function on the VDU allowing the TAB key to be used as on a typewriter. At each press of the TAB key, spaces will be printed on the screen up to the next designated column. Please note that unless you have used *NORMALKEY to disable single key entry, you will have to press SHIFT-TAB instead of TAB alone. The tab stops provided can be altered as follows:

```
*TABSTOPS 4,12,35,56 <RETURN>
```

will allow the tab stops to be redefined to the figures given. Up to eight tab settings may be defined and there must be a comma or space between each. The maximum allowable difference between adjacent tab stops is 32.

```
*TABSTOPS <RETURN>
```

without any parameter, will use the following default tab settings:

```
    9   18   27   36   45   54   63   72
```

Default settings may be incorporated into your own by leaving a space, thus

```
*TABSTOPS 10,20, ,30.....etc. <RETURN>
```

will define the stops at 10, 20, 27, 30 .....etc.

Note

Defined tab stops are stored in the resident integer variables T% and U%, so avoid using these variables for other purposes. The default tabs can be used with complete safety.

The facility may be disabled by typing *NOTAB <RETURN>.

APPENDICES

ERROR MESSAGES

CODE	ERROR MESSAGE AND POSSIBLE REASON
131	Bad BASIC program Program in memory is either corrupted or not a BASIC program
132	Too big Numeric parameter exceeds 32767
133	Bad number Numeric parameter contains an illegal character
134	Bad string String given wrongly, e.g. with opening quotes but no closing quotes
135	Syntax: Syntax error made in a command
136	No such file File to be loaded does not exist on the current filing system
137	BASIC file cut short BASIC file loaded is incomplete
138	Bad BASIC on file File being loaded is corrupted or not a BASIC program
139	No room Insufficient room in memory to carry out INSERT, MERGE or EXCHANGE
140	<end> less than <start> can occur in EXPAND, PARTSAVE, and RENUMBER

141 Can't open file
File cannot be accessed on the current filing system

142 Not found
Unable to find string in EXCHANGE

143 Renumber overlaps
Line numbers overlap other parts of the BASIC program

144 Renumber exceeds line 32767
32767 is the maximum BASIC line number

145 Nothing to renumber
No lines between <start> and <end> or no program in memory

146 Tabstop out of range
Column exceeding 79 in TABSTOPS definition

147 String too large
String exceeds 127 bytes in EXCHANGE

148 <length> too large
<length> more than 127 in EXCHANGE

149 Non-alphanumeric characters
incorrect string in EXCHANGE with <length> parameter

150 No room in line
If EXCHANGE would result in line just listed exceeding 255 bytes

151 Can't move program
MOVE unsuccessful

PARAMETERS

<1st>	The line number at which renumbering is to commence
<bytes>	The number of bytes in decimal that RETRIEVE is to recover
<columns...>	up to 8 decimal numbers defining the tabs in TABSTOPS
<end>	The final line number of a sequence - defaults to end of prograı
FISAP	Options in LVAR (q.v.)
<fsp>	File specification: the name of a file on the current filing system
G/S	Global (operates on all cases) or Selective (gives choice to user)
<inc>	Increment - the step from one line to the next in RENUMBER
<length>	used in EXCHANGE
<lineno>	BASIC line number
<new>	String to be substituted for <old> in search and replace routin
ON/OFF	Used in CURSOR to switch cursor on or off - default ON
<old>	The string to be replaced by <new> in EXCHANGE
SRM	options in SQUASH to omit spaces, REMs, or for multi-statement lines
<start>	the first line number of a sequence - defaults to start of program

BASIC KEYWORDS

ABS	148	EOR	130	LOMEM (R)	146	REPEAT	245
ACS	149	ERL	158	LOMEM (L)	210	REPORT	246
ADVAL	150	ERR	159	MID$(	193	RESTORE	247
AND	128	ERROR	133	MOD	131	RETURN	248
ASC	151	EVAL	160	MODE	235	RIGHT$(	194
ASN	152	EXP	161	MOVE	236	RND	179
ATN	153	EXT	162	NEW	202	RUN	249
AUTO	198	FALSE	163	NEXT	237	SAVE	205
BGET	154	FN	164	NOT	172	SGN	180
BPUT	213	FOR	227	OFF	135	SIN	181
CALL	214	GCOL	230	OLD	203	SOUND	212
CHAIN	215	GET	165	ON	238	SPC	137
CHR$	189	GET$	190	OPENIN (2)	142	SQR	182
CLEAR	216	GOSUB	228	OPENOUT	174	STEP	136
CLG	218	GOTO	229	OPENUP (1)	173	STOP	250
CLOSE	217	HIMEM (R)	147	OPT	–	STR$	195
CLS	219	HIMEM (L)	211	OR	132	STRING$(	196
COLOUR	251	IF	231	OSCLI (2)	255	TAB(	138
COS	155	INKEY	166	PAGE (R)	144	TAN	183
COUNT	156	INKEY$	191	PAGE (L)	208	THEN	140
DATA	220	INPUT	232	PI	175	TIME (R)	145
DEF	221	INSTR(	167	PLOT	240	TIME (L)	209
DEG	157	INT	168	POINT(	176	TO	184
DELETE	199	LEFT$(	192	POS	177	TRACE	252
DIM	222	LEN	169	PRINT	241	TRUE	185
DIV	129	LET	233	PROC	242	UNTIL	253
DRAW	223	LINE	134	PTR (R)	143	USR	186
ELSE	139	LIST	201	PTR (L)	207	VAL	187
END	224	LN	170	RAD	178	VDU	239
ENDPROC	225	LOAD	200	READ	243	VPOS	188
ENVELOPE	226	LOCAL	234	REM	244	WIDTH	254
EOF	197	LOG	171	RENUMBER	204		

Notes: (1) in Level 1 BASIC this keyword is called OPENIN
(2) only used in Level 2 BASIC
(L) used when keyword is on the left of an expression, e.g. PAGE=
(R) used when keyword is on the right of an expression, e.g. PRINT PAGE