Vic Pack

VP052

PROGRAMMING AIDS FOR THE VIC-20 plus 3K MEMORY EXPANSION

BUTII

INSTRUCTION MANUAL

AUTOMATIC ERROR CATCHING
AUTO - DELETE - DUMP - EDIT
FIND - HELP - KILL - OFF
RENUMBER - REPEAT - STEP - TRACE
UNNEW - HEX TO DEC - RECONFIGURE

audiogenic Ltd.

BUTI - INSTRUCTION MANUAL

CONTENTS

BUTI - DESCRIPTION

BUTI is a Basic programming utility cartridge which provides seventeen programming aid commands, and an extra 3K of user memory. BUTI gives you increased control of programs and variables, and eases programming through its Basic error catching and powerful editing features. Also with one instruction it formats the VIC to imitate the minimum memory or +3K configurations, or +8K when used with other expansions on a multi-cartridge board.

Programming commands available from BUTI are as follows.....

APPEND - Adds a program from tape storage to the end of a program in VIC memory.

AUTO - Automatically gives new line numbers while writing a Basic program.

DELETE - Deletes a selected block of lines from a Basic program.

DUMP - Displays all program variables and their values in alphabetical order, including dimensioned arrays.

EDIT - Searches for a given string of characters within the program, and replaces it with another given string.

FIND - Searches for a given string within the Basic program and displays each occurrence of it.

HELP - Displays a summary of BUTI commands, with syntax, for quick reference.

KILL - Disables BUTI without powering off your VIC.

OFF - Turns off STEP and TRACE commands.

RENUMBER - Renumbers all or selected parts of Basic programs. Automatically renumbers line references contained in Basic statements.

REPEAT - Toggles on or off the ability to automatically repeat each VIC key when held down.

STEP - Permits single step execution of a Basic program, and displays the number of the line being executed.

TRACE - Allows you to run your Basic program slowed down, with a display of the line numbers being executed.

UNNEW - Restores a program just eliminated with the NEW instruction.

VIC - Reconfigures the memory to minimum, +3K, +8K, or greater memory formats.

- Converts a decimal number to its hexadecimal and binary equivalents.

$ - Converts a hexadecimal number to its decimal and binary equivalents.

AUTOMATIC ERROR CATCHING - Whenever a program error stops the execution of a program, BUTI will list the line in which the error occurred, and display the offending characters in reverse field.

USING BUTI

BUTI commands operate in direct mode only (except STEP and TRACE, which can under certain circumstances be called from within a program). The command must be the only command on the screen line. Any command maybe abbreviated down to the first three letters. Press RETURN to execute the command. Most of the commands must be followed by certain parameters. These will be explained in detail below. Commands which require parameters can be used without the parameters being defined, in which case the parameters take on predefined default values.

IMPORTANT - When using abbreviated commands, a SPACE must be typed between the command and any parameters that follow.

EXAMPLES - Entering command in full...

```
RENUMBER1000,20,630-660
```

Note that the parameters are separated by commas. The dash indicates a range of lines, in this case from 630 to 660 inclusive. We can also use the abbreviated form, and leave out some of the parameters, as long as the commas and dash remain.....

```
REN 1000,,-660
```

The second parameter has not been stated, and so takes on its default value. You will see that the first number of the range has been left out too. BUTI will interpret the range as being from the start of the program to line 660 inclusive. Note the space between the abbreviation and the first parameter. Another example.....

```
REN ,20,630-
```

In this case, the first parameter has been left out, so that will take on its default value. Also the second number of the range has been left out. BUTI will interpret the range as being from line 630 to the end of the program inclusive. Another example.....

```
REN ,1
```

The range has been omitted, and BUTI will interpret the range as the whole program, from start to end. Also, all the parameters can be omitted.....

```
REN
```

All the parameters take on their default values.

The commands UNNEW, REPEAT, OFF and KILL do not require any parameters. REPEAT and OFF do not produce any visual output on the screen.

BUTI COMMANDS IN DETAIL

APPEND

SYNTAX	-	APPEND"FILENAME"
ABBREVIATION	-	APP
DEFAULT VALUES	-	If no filename is specified, APPEND defaults to first program found on tape.
EXAMPLE	-	APP "PROG2"

APPEND is used to attach a program or subroutine stored on tape to the end of a program residing in VIC RAM memory. APPEND works like a LOAD command, so that a typical screen display while APPENDing will look something like this....

```
APP "PROG2"

PRESS PLAY ON TAPE
OK

SEARCHING
FOUND PROG2

APPENDED PROG2
READY.
```

The programmer must make sure before using this command that the line numbers of the program to be APPENDed are higher than those of the program already in the VIC. BUTI will not interleave or merge program lines: it simply adds a program to the end of another.

AUTO

SYNTAX	-	AUTO start, increment
ABBREVIATION	-	AUT
DEFAULT VALUES	-	Start = 10 or last line generated by AUTO. Increment = 10 or last increment used.
EXAMPLES	-	AUTO10,20 AUT ,5

AUTO is used to generate line numbers as one types in Basic program lines. The parameters are start, i.e. the line number that you want to start with, and increment, i.e. the interval between the successive line numbers. For example, the command....

AUT 10,5

will give you line number 10, and the new numbers will go up in steps of 5. A new number is given as soon as you press RETURN at the end of a line.
If the line number that AUTO generates already exists in the program, then AUTO will put a '@' character immediately after the line number. This is to warn you that if you type in a new line the old one will be deleted and the new line will replace it. You do not have to delete the '@' character, as this will not be entered into the line as you press RETURN.

To exit AUTO, simply type RETURN after the line number is displayed.

All parameters are optional, and, if omitted, will revert to default values. Once AUTO has been used in a programming session, it will remember what line it was on when AUTO was exited, and the increment being used. Before AUTO is used, the default values for start and increment are both 10. After using AUTO, the default values are updated to reflect the values that were last used.

The current line number can be changed dynamically. That is, if you change the line number that AUTO generated, it will accept your change and continue numbering from there.

DELETE

SYNTAX	-	DELETE low - high
ABBREVIATION	-	DEL
DEFAULT VALUES	-	Low = beginning of program.
		High = end of program.
EXAMPLES	-	DELETE10-100
		DEL 200-
		DEL -150

DELETE removes blocks of Basic program lines from your program. To prevent accidental erasure of the wrong section of code, you must specify both low and high range parameters, in that order. The following formats are incorrect, and will result in a syntax error....

DELETE
DELETE10

Here are some examples to illustrate the use of the DELETE command.....

DEL 30-210

This will delete lines 30 and 210 and every line in between.

DEL 360-

This will delete line 360 and all lines up to the end of the program.

DEL -120

This will delete all lines from the start of the program up to and including line 120.

DUMP

SYNTAX	-	DUMP type of variable
ABBREVIATION	-	DUM
DEFAULT VALUES	-	Displays all types of variables, i.e. floating, integer, string, and array.
EXAMPLES	-	DUMP DUMP% DUM . DUM $ DUM (

DUMP displays in alphabetical order every program variable, and its value, of the type specified in the parameter. The parameter must be one of the following.....

. Display only floating point variables.
% Display only integer variables.
$ Display only string variables.
(Display only array variables.

If no parameter is specified then every variable of every type will be displayed.

You can control the display while DUMPing by pressing the following keys.....

SHIFT	Stops display while held down.
STOP	Exits from DUMP command.
SHIFT/STOP	Also exits from DUMP command.

EDIT

SYNTAX	-	EDIT/s1/s2/low - high,Q
ABBREVIATION	-	EDI
DEFAULT VALUES	-	Low = start of program.
		High = end of program.
EXAMPLES	-	See below.

EDIT searches for a given string of up to ten characters within the specified line range, and then replaces that string with a second specified string. The optional query mode allows you to look at each occurence before deciding whether to replace or not. In the syntax....

EDIT/s1/s2/low-high,Q

/	is a non-space delimiter.
s1	is the string to be searched for and replaced.
s2	is the string that replaces s1.
low	is the starting line of the range in which the search and replace is to take place.
high	is the ending line of the range.
Q	switches on Query mode.

Default values and conditions.....

/	No default value. The delimiter must be entered.
s1	No default value. Must be entered.
s2	If omitted, merely deletes s1. See examples below.
low	Default value = start of program.
high	Default value = end of program.
Q	If omitted, Query mode is not switched on.

The delimiter need not be the '/' character. It can be any character except SPACE, as long as the character does not appear in either s1 or s2.
S1 and s2 can be up to 10 characters long.

Let us suppose that we have the following program in memory.....

```
10 REM AN EDIT EXAMPLE
20 PRINT:PRINT:GOTO500
30 END
```

If you wish to replace PRINT with PRINT#1 you type...

EDIT*PRINT*PRINT#1*

In this example we are using '*' as the delimiter. No range is specified, so the EDIT command will go through the whole program looking for the string 'PRINT'. Each time it finds an occurrence of the string it will replace it, and print out the new line on the screen. In this example, EDIT will find the string 'PRINT' twice, both times in line 20, and will display as follows...

(cont.)

```
20 PRINT#1:PRINT:GOTO500
20 PRINT#1:PRINT#1:GOTO500

READY.
```

You will see that line 20 is listed twice, once after each replacement. Lines 10 and 30 were left untouched since they did not contain 'PRINT' in them. Now if we type.....

```
EDIT.#1..
```

we can get the program back to how it was. We are telling EDIT to replace all occurrences of the string '#1' with nothing, so effectively we are deleting the string. We are now using '.' as the delimiter. This is what happens....

```
20 PRINT:PRINT#1:GOTO500
20 PRINT:PRINT:GOTO500

READY.
```

The program is now back to its original form. Here are some more examples.....

EDIT&GOTO&GOSUB&	Replaces all occurrences of the string 'GOTO' with the string 'GOSUB'.
EDI "GOTO"GOSUB"10-60	Replaces all occurences of 'GOTO' from line 10 to line 60 inclusive with 'GOSUB'.
EDI +GOTO+GOSUB+,0	Searches for 'GOTO' in Query mode, i.e. each time it finds an occurrence it displays the line and asks whether to replace or not, before going on to the next occurrence.
EDI %GOTO%GOSUB%50	Replaces all occurrences of 'GOTO' with 'GOSUB' in line 50 only.
EDI *GOTO*GOSUB*-90,0	Searches for occurences of 'GOTO' from start of program up to and including line 90. In Query mode, so each occurrence found is displayed and queried before proceeding to the next.

In Query mode EDIT will search for the first occurrrence of your specified string and display that line on the screen, and a question mark, indicating that it is waiting for your response. The response must be one of the following.....

```
Y  -  To replace  s1 with s2.
R  -  To NOT replace, and to skip to the next occurrence.
X  -  To exit from EDIT command.
```

Whilst in EDIT, you can also: 1 - Hold down the SHIFT key to freeze the screen output. Release to continue. 2 - Press STOP key to abort any EDIT operation.

FIND

SYNTAX	-	FIND/s1/low - high
ABBREVIATION	-	FIN
DEFAULT VALUES	-	Low = start of program.
		High = end of program.
EXAMPLES	-	FIND/"ACC/60-120
		FIN *A1=*200-
		FIN &GOTO&

FIND is used to search for and display all occurrences of a certain string within the given line range. In the syntax.....

FIND/s1/low - high

/	Is a non-space delimiter.
s1	Is the string that to be searched for.
low	Is the starting line of the range in which the search is to take place.
high	Is the ending line of the range.

As in EDIT, the delimiter can be any character except SPACE, as long as it is not one of the characters in the string to be searched for. The following examples illustrate the use of the FIND command.....

FIND.GOTO.	Finds and displays all occurrences of the string 'GOTO'.
FIN *GOTO*-99	Finds and displays all occurrences of 'GOTO' from the start of program up to and including line 99.
FIN /GOTO/100-	Finds and displays all occurrences of 'GOTO' from line 100 to the end of program.
FIN &GOTO&1000-2000	Finds and displays all occurrences of 'GOTO' from line 1000 to line 2000.

You can freeze the screen output by holding down the SHIFT key. Release to continue.
You can abort any FIND operation by pressing STOP.

HELP

SYNTAX - HELP
ABBREVIATION - HEL
DEFAULT VALUES - None
EXAMPLE - HEL

HELP is not a programming command. Rather, it is a quick reference guide to the commands available from BUTI, and the syntax that they require. When HELP is executed, the screen will clear and the commands will be displayed with the syntax and parameters required. Where a parameter is enclosed by these brackets, < >, that parameter is optional.

KILL

SYNTAX - KILL
ABBREVIATION - KIL
DEFAULT VALUES - None
EXAMPLE - KIL

KILL disables BUTI. While disabled you cannot make use of any of the BUTI commands until re-enabled. BUTI can be re-enabled by turning the VIC power off and on again, or by entering the following SYS call.....

SYS 40993

OFF

```
        SYNTAX   -   OFF
  ABBREVIATION   -   OFF
DEFAULT VALUES   -   None
       EXAMPLE   -   OFF
```

OFF is used solely for the purpose of switching off STEP and TRACE commands. Along with STEP and TRACE, it can be called from within a program by a SYS call. Thus you can turn on TRACE or STEP from the program just before it gets to the part you want to examine, and then switch it off afterwards. The SYS values are as follows.....

```
OFF          SYS 44434
TRACE        SYS 44436
TRACE W      SYS 44443
STEP         SYS 44449
STEP W       SYS 44455
```

All modes can be changed directly. It is not necessary to execute an OFF to go from STEP to TRACE, etc..

RENUMBER

SYNTAX - RENUMBER start, increment, low - high
ABBREVIATION - REN
DEFAULT VALUES - Start = 10
Increment = 10
Low = start of program.
High = end of program.
EXAMPLES - REN
REN ,1
REN ,,10-99
REN 1000,,50-

RENUMBER changes the numbers of the lines within the given range to new numbers starting at the number specified in the 'start' parameter, going up in steps of whatever number was specified in the 'increment' parameter. Also, all the commands in the program which specify line numbers are updated to reflect the newly renumbered lines. Such commands are GOTO, GOSUB, ON xxxx GOTO, ON xxxx GOSUB, THEN, RUN, LIST. These are renumbered whether they fall within the line range of the RENUMBER command or not.
If a line reference is made to a line that does not exist, then that line reference will be changed to 63999 to reflect an error condition.

Any or all parameters may be omitted and left to their default values. To skip over a parameter, just leave it out....

RENUMBER,,10-99

This would renumber lines 10 to 99, starting at 10 and going up in increments of 10. You could also renumber the entire program starting at 10 and incrementing by 10, by just entering.....

REN

Or to renumber the entire program with an increment of 1, enter....

REN ,1

Or you could renumber part of the program up to line 1000 using the default start and increment by entering.....

REN ,,-1000

REPEAT

SYNTAX - REPEAT
ABBREVIATION - REP
DEFAULT VALUES - None
EXAMPLE - REP

REPEAT acts as a toggle switch for the automatic repeat of any key on the VIC keyboard for as long as the key is depressed. To switch on REPEAT, enter REPEAT, or abbreviation, and to switch off, enter it again.

STEP and TRACE

STEP

```
        SYNTAX   -   STEPW
  ABBREVIATION   -   STE
DEFAULT VALUES   -   Window option off.
      EXAMPLES   -   STE
                     STE W
```

The STEP command switches on STEP mode. When in STEP mode, whenever you RUN your program it will run one line or statement at a time. Each time you press the SHIFT key, the program will advance to the next line or statement. If you use the window option by typing.....

STEPW

then, when you RUN the program, a window will appear in the top right hand corner of the screen, with line numbers displayed in it. As you STEP through the program, the line numbers in the window will show you which lines have just been executed, and which one is just about to be executed. The line number at the bottom of the window is the line about to be executed, the one above it is the one which was executed last, and those above that are the lines which were executed before that one.

TRACE

```
        SYNTAX   -   TRACEW
  ABBREVIATION   -   TRA
DEFAULT VALUES   -   Window option off.
      EXAMPLES   -   TRA
                     TRA W
```

TRACE mode is used to slow down the execution of your program. As in STEP, the window option allows you to see which line numbers are being executed at the time. The program can be slowed still further by holding down the SHIFT key.

The OFF command is used to switch off STEP and TRACE.

All STEP and TRACE commands, as well as the OFF command, can be executed with a SYS call from within your program. This allows the user to switch on TRACE or STEP just for the part of the program that is giving trouble, and OFF afterwards. The SYS values are as follows.....

```
OFF          SYS 44434
TRACE        SYS 44436
TRACE W      SYS 44443
STEP         SYS 44449
STEP W       SYS 44455
```

(cont.)

All modes can be changed directly, it is not necessary to execute an OFF to go from TRACE to STEP, etc..

UNNEW

SYNTAX - UNNEW
ABBREVIATION - UNN
DEFAULT VALUES - None
EXAMPLE - UNN

UNNEW will restore a program that has been NEWed. It will not restore a program that has been deleted with the DELETE command. In that case you will succeed in restoring only the last line of the program. If you have typed NEW and then done any kind of line editing (adding, deleting, changing), or created new variables, then UNNEW will not be able to retrieve your program. This is because NEW does not actually erase your program, it merely changes some pointers. So new data may have been written over your program. Hence it will not be possible to UNNEW your program.
If you try to UNNEW a program that has been written over, you will probably get some 'garbage' in your program area. If this occurs, simply type NEW again to get rid of the garbage.

Try following this example.....Enter....

NEW

Then enter this program.....

```
10 REM A SHORT PROGRAM
20 REM TO TEST UNNEW
30 END
```

Now enter.....

NEW

and LIST

You will see that no list appears. Your program has been NEWed. Now enter.....

UNNEW

and LIST

Your program has been successfully restored. Now enter.....

NEW

and LIST

We have NEWed the program again. Let us see what happens if we change a variable value and then try to restore the program. Enter.....

A=3

Then.... (cont.)

UNNEW

and LIST

What we get is garbage.....

65 NEXT@

We have got garbage back, since the original program has been 'destroyed' by the A=3 statement. At this point your program is lost. (You might try to restore it with POKE statements).

Be sure to type NEW to clear out any garbage when you get it.

NOTE - Variables are created when you mistype a command in direct mode. Try the following experiment.....

```
Enter.....    CLR
Response..    READY.
Enter.....    ABCDEFGH
Response..    ?SYNTAX
               ERROR
              READY.
Enter.....    DUMP
Response..    AB=0
              READY.
```

The variable AB was created from the string ABCDEFGH that we typed. Care must be taken not to mistype entries after a NEW command, otherwise an UNNEW command will not restore the program.

VIC

SYNTAX - VIC type
ABBREVIATION - VIC
DEFAULT VALUES - Configure as standard VIC.
EXAMPLES - VIC
VIC 3
VIC 8

The VIC command is used to reconfigure the VIC to imitate a different memory size and back again. If this version of BUTI comes included in a 3K expansion cartridge, then the VIC command can be used to reconfigure the VIC to imitate the unexpanded configuration, and back again to +3K. Or if BUTI is being used with other expansions, or in a multi-cartridge board, then the VIC command can be used to reconfigure to and from the +8k format.

The commands are.....

VIC Configure as a standard VIC.
VIC 3 Configure as a VIC with a 3K expansion board.
VIC 8 Configure as a VIC with 8K or larger expansion.

Executing one of the VIC commands will cause the system to be re-initialised just as if the power had been turned on, except for the following differences.....
1 - The screen memory will be moved.
2 - The colour RAM will be moved.
3 - The start of free RAM will be moved.

Where these are moved to depends on which VIC command was executed, and what memory is available in the system.

Memory will be configured for each of the VIC commands as in the following table.....

COMMAND	DESCRIPTION	LOCATION
VIC	Screen RAM	$1E00-$1FFF
	Colour RAM	$9600-$97FF
	User RAM	$1000-$1DFF
VIC 3	Screen RAM	$1E00-$1FFF
	Colour RAM	$9600-$97FF
	User RAM	$0400-$1DFF
VIC 8	Screen RAM	$1000-$11FF
	Colour RAM	$9400-$95FF
	User RAM	$1200-top of RAM

A VIC 8 command will look for the highest RAM in the system and set the user memory to that value. Thus the end of user RAM is dependent on the size of memory installed.

NOTE - The use of the VIC command is dependent upon the amount of memory actually installed in your system.

and $

```
        SYNTAX    -    # number
  ABBREVIATION    -    #
DEFAULT VALUES    -    None
       EXAMPLE    -    #255

        SYNTAX    -    $ number
  ABBREVIATION    -    $
DEFAULT VALUES    -    None
       EXAMPLE    -    $F000
```

Entering # followed by a decimal number will give you the hexadecimal and binary equivalent of that number.
Likewise, entering $ followed by a hexadecimal number will give you the decimal and binary equivalent of that number.

Typical conversions would look like this.....

```
#255
=$00FF
=%00000000 11111111

READY.
$F000
=61440
=%11110000 00000000

READY.
```

NOTE - Numbers to be converted must be within the range of zero to 65535 ($FFFF). The shift key is used to type the # and $ characters.

BUTI DEDICATED ERROR MESSAGES

BUFFER OVERFLOW - As BUTI makes changes within a line, it uses a buffer as a work space. If the physical length of a line being changed exceeds the buffer length, this error message will be displayed. The line in which it occured will also be displayed. This is not a fatal error, and the command being executed will continue at the next line. This message can occur in RENUMBER and EDIT.

To avoid this problem, keep the length of program lines less than three screen lines long.

INSUFFICIENT MEMORY - If a VIC 3 command is attempted, and there is no 3K memory expansion installed, this message will be displayed. No changes will be made to the VIC's memory configuration.

To successfully execute a VIC 3 command, it is necessary to have a 3K RAM expansion board installed. Of course, if your version of BUTI comes included in a 3K expansion cartridge, then this situation will not arise.

OUT OF RANGE - When the line number being generated exceeds the VIC's maximum legal line number, 63999, this message will be displayed. This error causes the current routine to abort. Also, when RENUMBERing a section of program, and the line number of the renumbered section exceeds or equals the number of the first line immmediately after that section, then this message will be displayed and the RENUMBER routine will abort. This error can occur in AUTO and RENUMBER, since they are the two commands that deal with line numbers.

To avoid this problem, try entering a different set of parameters for the command.

MANUFACTURED & DISTRIBUTED IN THE U.K. BY AUDIOGENIC, PO BOX 88, READING, BERKS.