FOR THE AMSTRAD
CPC464, CPC664, CPC6128

TOOLKIT
Basic Programmer's Aid

BEEBUG SOFT

BEEBUGSOFT

TOOLKIT
FOR THE AMSTRAD CPC 464, CPC 664 & CPC 6128

By D. Pilling

© BEEBUGSOFT 1985
Dolphin Place, Holywell Hill, St. Albans, Herts.

All rights reserved. No part of this product may be reproduced in whole or part by any means without written permission of the publisher. Unauthorised hiring, renting, loaning, public performance or broadcasting of this product or its constituent parts is prohibited. While every care is taken, the publisher cannot be held responsible for any errors in this product.

CONTENTS

INTRODUCTION

Beebugsoft's Toolkit is a sophisticated piece of software designed to assist programming on the Amstrad CPC464, CPC664, and CPC6128 range of computers. It is supplied on cassette, disc or rom, and provides the user with more than 30 new commands which not only speed up the process of programming, but assist in the task of debugging, and generally streamline the activity of computing.

For simplicity of use, all commands may be entered with a unique command name directly from the keyboard, or from within your own Basic program. A special option also allows nearly all commands to be selected from a main menu, providing extreme ease of use. A special Help call has been incorporated to give the user immediate information on the syntax of Toolkit commands, and as a further aid, parameters required by the various Toolkit routines are prompted for, if not supplied by the user. Extensive error checking routines are also incorporated, and the user is given a range of error messages if commands have been incorrectly entered etc.

Conventions used in this manual

In this manual specific key presses required by Toolkit (such as the "Enter" key) will be indicated thus: ENTER.

All parameters are shown in this manual enclosed in angled brackets. Single sets of brackets <> are used to indicate essential parameters, while double brackets <<>> indicate optional parameters.

Fitting or Loading Toolkit

Cassette: If Toolkit is supplied on cassette, you should simply insert your cassette into the player, type:

```
    run"toolkit ENTER
```

then press the "Play" button on the recorder as instructed. When loading is complete, the familiar "Ready" message will appear.

Disc: If you have purchased Toolkit on disc, simply insert the disc into your drive, and type:

```
    run"disc ENTER
```

Again, when the "Ready" message appears, Toolkit is ready for use.

Cassette and disc versions of Toolkit will remain in memory until you switch off your machine, unless they are overwritten in memory by other programs. Because Toolkit must reside in memory when run from cassette or disc, the amount of memory available to Basic is reduced. This will not normally cause problems, but with very long Basic programs, there may not be enough room in memory for Toolkit. To assist you when this occurrs, we have provided two further files on cassette and disc called toolkit1 and toolkit2. These each contain approximately half of the Toolkit routines, and will each function on their own occupying around half the memory of the full Toolkit.

To find in which half a given routine is located you should run the file Info. To do this type:

```
    run "info ENTER
```

Cassette users should note that the files toolkit1, toolkit2 and info are located on the reverse side of the cassette.

Rom: If Toolkit is supplied on rom, you will need an external rom socket attachment. These may be purchased from a number of suppliers, and you should consult the instructions which accompany your rom socket about its connection to your computer, and the insertion of Toolkit into it. Once fitted in this way, Toolkit will remain on call at all times without the inconvenience of having to load it from disc or cassette.

Once Toolkit is resident in your machine (whether on rom or in memory), you may use your computer as normal, except that where Toolkit has been loaded into memory from cassette or disc, less memory will be available to the user for his Basic programs.

For most purposes, you will probably be using Toolkit to work on a program in Basic. This may be loaded and/or run in the normal way from cassette or disc once Toolkit is resident in your machine.

Calling Toolkit

Whether Toolkit is on rom or has been loaded into memory from cassette or disc as described above, it is called in exactly the same way using "I" (bar) commands.

For example if you type the following:

```
    |help ENTER
```

you will see Toolkit's help screen appear.

Although each Toolkit command has a unique command word associated with it, you may find that some of these clash with the names of commands of other roms in your machine (if your machine is fitted with an external rom board). Toolkit has a special feature to avoid command name clashes. If any command name clashes, simply preface the command name with a "b" - for Beebugsoft – (eg type "|bhelp" instead of just "|help"). This will ensure that the command is intercepted by Toolkit rather than any other rom.

Another way to call a Toolkit command is to type:

```
    |tools ENTER
```

This calls a menu from which almost all of Toolkit's routines may be selected.

Parameters

All Toolkit commands must be entered in lower case; and if parameters are being entered with the command, the command should be followed by a comma, and each parameter separated by a comma. In fact most of Toolkit's routines require one or more parameters to be entered to specify how they are to be executed. For example the renumber routine needs to know how many lines to renumber, what line to start at, and so on. These may be entered immediately following the command name as one or more numbers separated by commas. Thus for example the command Ilmove which will move a block of Basic lines within a Basic program, might be called as follows:

```
    Ilmove,50,100,2000 ENTER
```

In this case the routine will move those program lines from 50 to 100, and relocate them starting at line 2000.

If you had selected "LMOVE" from the Tools menu, or had simply entered:

```
    Ilmove ENTER
```

Toolkit would have prompted you for parameters.

Addresses

The numeric parameters so far introduced have been program line numbers. With other commands, parameters may take the form of one or more addresses in memory. Where addresses are required they may either be entered in decimal or in hex. Hex addresses should be preceded by an ampersand character "&". Thus to move your entire Basic program to a new address in memory starting at 1000 hex, you could use either of the following:

```
    Ibmove,&1000 ENTER or
    Ibmove,4096 ENTER
```

since 4096 is the equivalent of 1000 hex.

Optional Parameters

Some parameters are optional, such as the number used with the Irom command. Optional parameters are indicated in this manual with a double angle bracket thus:

```
    Irom,<<rom number>> ENTER
```

If you wish to call the command without the parameter, simply type

```
    Irom ENTER
```

Strings

Four of the Toolkit routines also require text (or strings) to be entered. Partsave is one of these: the Partsave routine needs to know what filename to save a segment of your program under. The correct syntax for this is illustrated by the following example. To save lines 300 to 5000 of the Basic program in memory under the new filename "partone", you could type the following:

```
    |partsave,"partone",300,5000 ENTER
```

There are three important points to note here:

a. String parameters are always entered before numeric ones.
b. You must always enclose string parameters between quotation marks.
c. String parameters may not be entered in a bar command on the CPC464.

This last point needs some elaboration. It is feature of the CPC464 (though not the CPC664 or the CPC6128) that strings (ie text) may not be entered in a "|" (or bar) command. So if you have a CPC464, you should not try to enter any parameters for any of the four commands:

search
replace
list
partsave.

Simply enter the command name, and you will be prompted for the parameters. Users of the 664 and 6128 have the option of entering all text and numeric parameters directly following the command name.

Thus for example to Partsave lines 10 to 300 of your Basic program, under the new filename "progpart", you could enter the following on a CPC664 or CPC6128:

```
    |partsave,"progpart",10,300 ENTER
```

On a CPC464, you could only enter:

```
    |partsave ENTER
```

Toolkit would then prompt you with a request for the filename and the two line numbers.

By contrast of course, if a Toolkit routine has been called from the main Tools menu, there is no opportunity to directly enter any parameters (since you are simply selecting one item from a menu screen), and all parameters will be prompted for regardless of which computer you are using.

Escape

The ESCAPE key may be used at any time to exit a Toolkit routine, or to exit the main Tools menu. On pressing ESCAPE you will see the "Ready" prompt appear signifying that you have been returned to Basic. Toolkit will remain resident in your machine and may be called at any time.

During scrolled screen displays, such as those generated by the XREF command, the ESCAPE key has a different effect. Again it operates exactly as in Amstrad Basic. A single press of the ESCAPE key will halt a scrolling display, and any other key will reinstate it.

Screen Modes

Toolkit will work in all screen modes, though you will find its output clearest in mode 1. When a Toolkit routine is called individually using a bar command, no changes will be made to the currently selected screen mode or ink and pen colours. If on the other hand you call the Tools menu, mode 1 will be selected, and the colour palette will be specifically defined in order to obtain the clearest display.

THE COMMANDS

In the bulk of this manual, each command will be treated in alphabetical order. But there are three commands which deserve special attention out of strict order. These are |help, |tools and |toolsoff.

HELP

Syntax: |help
Function: General help page giving command list, and syntax.

The help command will give a list of all of Toolkit's keywords, and their syntax. As in this manual itself, essential parameters are enclosed in single angular brackets <>, while optional parameters have a double bracket<<>>.

The amount of information displayed by this command requires two screen pages, and on the first screen the user is prompted to press any key to display the second.

TOOLS

Syntax: |tools
Function: Displays a menu from which almost all of Toolkit's commands may be selected, and sets the function keys for use with Toolkit.

The Tools menu provides the easiest way of calling Toolkit commands. When |tools is entered, mode 1 is first selected, and the colour palette adjusted if necessary for a clear display. The following menu then appears:

```
        AMSTRAD TOOLKIT from BEEBUGSOFT
----------------------------------------

       A BMOVE               M PACK
       B EMEM                N PARTSAVE
       C FORMAT              O PMEM
       D FREE                P RENUM
       E HELP                Q REPLACE
       F KON                 R ROM
       G KOFF                S RSX
       H KEY                 T SEARCH
       I KEYDEF              U START
       J LIST                V TRON
       K LCOPY               W TROFF
       L LMOVE               X XREF

----------------------------------------

Select option:
```

The user selects a routine by letter (or returns to Basic by pressing ESCAPE). If the routine selected requires the entry of any parameters, these will be prompted for.

Seven of Toolkit's commands do not figure on the menu. These are as follows:

|tools
|toolsoff
|pron
|proff
|dumpa
|dumpe
|reset

The reason for this is simply that in each case the command would serve little or no purpose if called from the menu. For example, calling the screen dump from the menu would just give you a graphics dump of the menu each time!

Function keys

Calling |tools will also set the function keys to the following functions for ease of use:

f0	Set mode 0
f1	Set mode 1
f2	Set mode 2
f3	|reset
f4	|help
f5	|free
f6	|rom
f7	|pron
f8	|proff
f9	|tools

Thus once |tools has been called, the main tools menu may be recalled at any time by pressing the 9 key on the keypad at the right hand side of the keyboard. The other defined keys work in a similar way. Press 1 and you will set up mode 1, and so on.

Also CTRL together with the small ENTER key on the numeric keypad will be defined:
run"disc
This will automatically 'boot' start programs where the first file is called 'disc'.

TOOLSOFF

Syntax: |toolsoff
Function: Clears Toolkit from memory.

This command is for use on cassette and disc versions only, and is used to reclaim space for Basic, should the need ever arise, by clearing Toolkit from memory. The operation leaves resident Basic programs intact. To use Toolkit routines after this command has been used, Toolkit will need to be reloaded into the machine.

As a safety precaution whenever this command is called, the user is asked to confirm his intention before Toolkit is cleared.

BMOVE

Syntax: |bmove,<address>
Function: moves a Basic program in memory to a different address.

This routine carries out a block move of the whole of the current Basic program to a new address in memory. It also resets the "start" address (used by Basic to find the user's program) to the new start address which you have chosen.

Toolkit allows you to alter this "start" address separately. See the command |start.

The most likely reason for wishing to move a Basic program in this way is to leave room for a second Basic program to be held in memory at the same time.

The routine requires an address to be entered. This is the new start address of the memory block that will hold the program. Like all address parameters required by Toolkit, it may be entered in decimal or in hex. If it is entered in hex, it should be preceded by a "&" character. For example:

```
    |bmove,&1FF0 ENTER
```

When you are using bmove, you should take great care in determining the relocation address, since an ill-chosen address could corrupt your Basic program; though Toolkit will make checks on the viability of your proposed new address, and prompt with the message:

```
    Address out of range
```

if problems are envisaged.

DUMPA / DUMPE

Syntax: |dumpa
or |dumpe
Function: 16 tone screen dump for Amstrad or Epson printers.

These commands will produce a screen dump to either the Amstrad DMP1 printer (|dumpa) or to an Epson type MX or FX printer (|dumpe). The dump will work in any mode, and will represent up to 16 different ink colours by using different types of shading.

As suggested earlier, you obviously need to call this routine when the screen which you wish to dump is actually on-screen. For this reason it cannot be called from the Tools menu: that would just give you a dump of the Tools menu each time! You also do not, in all probability, wish to type the command ldumpa or whatever, directly to screen for the same reason: your carefully prepared screen will have the word "ldumpa" written across it.

The way to avoid this is to call the dump routines from the same Basic program which has put onto the screen the contents which you wish to dump.

If for example the screen was produced by a file on disc, then the following program could be used to load the screen from cassette or disc, and then dump it to the printer – without corrupting it by overprinting command text across it.

```
10 load"filename
20 ldumpa
```

The feature which makes this possible is the ability to embed Toolkit's bar commands within a Basic program, as opposed to always executing them directly from the keyboard in the so-called "immediate mode". In practice any of Toolkit's commands may be called in this way.

EMEM

Syntax: lemem,<address>,<<rom number>>
Function: Display and allow editing of the contents of memory.

Calling lemem followed by an address in memory (preceded by "&" if in hex) will display the contents of a block of memory starting at the address given. For example:

```
lemem,&1F00 ENTER
```

The range of directly addressable memory on all three Amstrad computers covered in this manual is from 0 to 65536 decimal (&0000 to &FFFF hex).

The display gives the contents of memory in both hex and Ascii (with all codes outside the printable range represented by dots). To edit memory, simply overwrite at the current cursor position, either in hex or Ascii. A banner at the top of the screen indicates which mode you are in (ie hex or Ascii), and in any case this is also indicated by the position of the reversed editing cursor. An example is given below.

Once inside the editor the following keys are operative:

Tab: This key switches between editing the hex and Ascii parts of the display

Arrow keys: The cursor keys move the editing cursor around the block of memory on display. This will cause the display to scroll if you attempt to move the cursor beyond the top or bottom line of the display.

Shift/Arrow keys: Pressing SHIFT at the same time as the up or down arrow keys will move the display by a whole screenful.

Escape: This will exit the editor.

To edit memory, simply type in the new values that you require, either as a pair of hex digits if you areediting in the hex area, or with a character from the keyboard if you are editing in Ascii. The digit or character typed will immediately appear on-screen, and be placed into memory at the same time. The cursor will also move on to the next position.

As an example of using the editor, enter the following:

```
    |emem,&1000 ENTER
```

The display produced by this command is clearest in mode 1, where the screen is split into three blocks: A column of hex addresses at the far left of the screen incrementing in units of eight; a block consisting of 23 lines each containing 8 pairs of digits – probably all zeros; and a block of 8 by 23 characters, probably all dots.

The top left hand zero of the hex block will be reversed out, indicating that the cursor is currently at that position. If you press "5" on the keyboard, the 5 will appear in the cursor position; and together with the 0 to its left makes up 50 hex. This is equivalent to the letter "P" on the so-called Ascii code; and for this reason you will see a "P" appear in the first position in the Ascii block to the right.

If you now enter a "1" in the position adjacent to the 5, you will produce a 51 hex, and the "P" will change to a "Q". And so on.

So far we have dealt only with cases in which emem is called with only a single parameter. A second parameter may be used in any call to emem to indicate that you wish to look at a particular rom in your machine. The default is rom 0, as you will see from the upper banner whenever you call emem with only a single parameter.

The rom number may be anything from 0 to 255 decimal; but of course, you cannot alter roms by writing to them with an editor, so emem will only allow you to look at them; but this can still be extremely useful.

If you do wish to examine the roms in your machine, you will need to look at memory in the range &C000 to &FFFF.

FORMAT

Syntax: |format
Function: Format a disc

This command is used to format a disc ready for use with the CPC464, 664 or 6128 computers.

New discs must be formatted before use, but you should note that formatting a disc will irretrievably erase its entire contents, so be careful!

When the format command is called the user will be prompted as follows:

```
     Drive A/B:
```

You should press **A** if you have a single drive. You are then asked:

```
     Type S-System D-Data I-IBM:
```

These are the three formats available on the Amstrad disc system, and are detailed in the user manual (or in the disc drive manual if you have purchased the disc drive separately). For most purposes you will probably be using the System format; and will therefore reply with an **S**.
You are then asked to confirm your intention

```
     Are you sure Y/N:
```

Pressing **N** at this point will abort with no damage done.

Once formatting begins, you will see an updated report of the track currently being formatted. When the process is complete, you will be asked if you wish to format another disc; and a reply of **Y** will cause a repeat of the routine.

FREE

Syntax: |free
Function: Gives a set of status information.

On calling |free, the user is presented with a screen similar to the following:

```
     Program Start     :    367    (Ø16F)
     Program End       :    414    (Ø19E)
     Program Size      :     47    (ØØ2F)
     Himem             :  22499    (57E3)
     Last Location     :  41971    (A3F3)
     Free Memory       :  22Ø85    (5645)
```

Each parameter, as you can see, is given in both decimal and hex for convenience. All but Himem are fairly self-explanatory terms, each of which has its particular significance to the Amstrad programmer. The most obviously important to the newcomer is Free Memory. This is simply the amount of memory remaining for a Basic program. As the program size increases, so this will decrease. "Last Location" is the address of the last free location at the top of memory.

The term Himem is a pseudo variable of Amstrad Basic, and is explained in the User Manual.

KON / KOFF

Syntax: |kon
or |koff
Function: Turns on or off Toolkit's abbreviated keyword entry feature.

This feature can save the programmer a great deal of time when entering Basic from the keyboard. Once |kon has been input, Basic keywords can be entered in a highly abbreviated form. The abbreviations are generally speaking obvious ones, and therefore easily remembered; and once you list your program, Basic prints out all keywords in full, so that you do not have to try to decipher your program listings when you have used this feature.

As a simple example, if you wish to edit line 150, you need only enter the following:

```
    e.150 ENTER
```

Note that you do not even need to leave a space after the "e", since Toolkit handles all this for you.

The command |koff simply turns off the feature should it not be required for any reason.

For convenience, a table of abbreviations used is given below:

a.	auto
b.	border
c.	chr$
cl.	clear
d.	data
dr.	draw
del.	delete
e.	edit
f.	for
g.	goto
gr.	graphics (CPC 664 & CPC 6128 only)
gos.	gosub
h.	hex$
hi.	himem
i.	input
in.	inkey$
k.	key
l.	list
lo.	load
loc.	locate
m.	mode
mo.	move
n.	next
p.	plot
pa.	paper
r.	return
re.	renum

s.	step
sa.	save
so.	sound
t.	then
w.	window

KEY

Syntax: |key,<<first key>>,<<last key>>
Function: Lists the function key defintions in a form in which they can be edited.

This command lists the function keys and their contents. You may if you wish supply the number of the key at which listing starts and ends. If no numbers are supplied the standard 13 keys are listed.

The list is produced in such a way that you may use the cursor and Copy keys to edit a particular key definition.

If no parameter is given with the command, Toolkit will just print out all those keys that are set.

Note that the first and last key parameters used with this command must be the so-called "expansion character" numbers, and lie in the range 128 to 159. You are referred to the machine user manual in the section dealing with the Basic keyword "KEY" for further details.

KEYDEF

Syntax: |keydef,<<first key>>,<<last key>>
Function: Lists what codes any key or group of keys have been set to.

This command is used to read the code value of any of the Amstrad's keys. The key number following the command may specify a single key, or a range of keys. The number used is the so-called key number of the key to be investigated.

The printed result for each key will contain five numbers as follows:
- Key number
- Auto repeat (1=set, 0=unset)
- Ascii code of key
- Ascii code of key when used with SHIFT
- Ascii value of key when used with CTRL

If for example you enter:

```
|keydef,46ENTER
```

the following will be printed:

```
KEYDEF 46,1,110,78,14
```

This shows that the 'N' key (key number 46) has auto repeat set (1=set, 0=unset), and prints 'n' (Ascii code 110) when pressed. When pressed together with SHIFT it prints 'N' (Ascii code 78) and with CTRL it prints Ascii character 14.

If an asterisk is displayed before the word KEYDEF e.g.

```
    * KEYDEF 46,0,110,78,14
```

then it indicates that the definition has been altered from its default setting.

As with the |key command above, |keydef prints the key codes in a manner suitable for Copy editing. To edit a key definition printed by Toolkit, use the cursor and copy keys as you would when copy-editing a line of Basic.

LIST

Syntax: |**list,<filename>,<<start line no>>,<<end line no>>**
CPC464 users should type the command name only; the parameters will be prompted for.
Function: Lists a program directly from cassette or disc without affecting any program in memory.

This routine is particularly useful if you are working on one program, and wish to look at the listing of another. The parameters are used as follows:

Filename: This is simply the filename of the program on cassette or disc which you wish to list.

Start line number: This optional parameter is the start line for the list.

End line number: This is the line number at which listing terminates.

If only one numeric parameter is given, Toolkit will take this to be the start line number.

To pause the listing, press ESCAPE as normal with the ordinary List command, and resume with a press of the SPACE BAR. Alternatively, a second press of ESCAPE will terminate the routine.

LCOPY

Syntax: |**lcopy,<start line>,<end line>,<new start line>**
Function: Copy a set of Basic lines to a new position.

You may sometimes wish to duplicate a set of lines within a Basic program. This routine streamlines the operation. Though it should be said here that in many cases where groups of lines might be duplicated, it will probably prove more efficient to put them into a subroutine which may be called as many times as you wish without the need for duplication. Having said that, there are some occasions where duplication is the most efficient course of action.

To copy a section of Basic lines, simply specify the three addresses with the command as follows:

Start line: This is the number of the line at the start of the block to be copied.

End line number: This is the line number at the end of the block to be copied.

New start line: This is the new line at which the copied block will start.

Once the copy is made, the program is automatically renumbered, to keep the number sequence intact; and if necessary, Toolkit will even create space in which to insert the new lines.

LMOVE

Syntax: **|lmove,<start line>,<end line>,<new start line>**
Function: Move a set of Basic lines to a new position.

This routine is similar to the |lcopy routine above, except that the source lines are deleted from their old position, thus effecting a **move** rather than a **copy**. lmove will thus enable you to pick up a block of Basic lines and physically move them to some other position in your program. The parameters which the command requires are the same as those for lcopy:

Start line: This is the number of the line at the start of the block to be copied.

End line number: This is the line number at the end of the block to be copied.

New start line: This is the new line at which the copied block should start.

As with the previous command, Toolkit creates any space necessary for the moved lines, and renumbers the program after the move has been made to keep the sequence intact.

PACK

Syntax: **|pack**
Function: **Compacts a Basic program.**

When this routine is called the user is prompted for the program compacting options which he requires. Toolkit prints the following:

```
Enter options:
  1. Rems
  2. Spaces
  3. Variables
  4. Concatenate
  5. Print Output
Press ENTER to pack
```

Options 1 and 2 simply remove all rem lines, and unused spaces. This is performed in an intelligent fashion, avoiding for example removing rems which are the subject of a GOTO command elsewhere in the program, and of course avoiding the removal of spaces when they appear within strings of text.

Option 3 scans the program, replacing all variable names with ones of the shortest possible length.

Option 4 takes groups of Basic lines, and rewrites them as single multi-statement lines.

Once you have been prompted for your selection of options, pressing ENTER causes the pack routine to run all four packing options. Alternatively, any combination of the four may be specified by simply entering the numbers required, followed by ENTER. Toolkit itself provides the separating commas.

Thus to compact rems and variables only, you would type:

    13ENTER

When packing begins, Toolkit prints out the original length of the program with a continuously updated display of bytes of memory saved. When packing is complete, the new program length is also displayed.

Before using the pack routines you are advised to make a backup copy of your full program, because compacted versions are not easy to read or to modify. It is a good policy to only compact a program once it is in the form in which you wish to use it.

PARTSAVE

Syntax: |partsave,<filename>,<<start line>>,<<end line>>
CPC464 users should type the command name only; the parameters will be prompted for.
Function: Saves a specified part of a program to cassette or disc.

As its name suggests, this routine allows you to save to cassette or disc any part of a program resident in memory.

The filename is essential, and may be any legal name. The start and end line numbers are both optional. But obviously if neither are specified, Toolkit will save the whole program. If only one of the two line number parameters are entered, Toolkit will assume it to be the start line number, and will save all lines starting at that number.

PMEM

Syntax: |pmem,<start address>,<end address>,<<rom number>>
Function: Dumps memory block to printer.

The function of |pmem is similar in many respects to that of |emem. The same parameters are required, and the display takes a similar format, but |pmem does not allow editing of memory, and simply dumps the whole block specified to the printer. In this way you may obtain the complete listing of a machine code program for example.

PRON / PROFF

Syntax: |pron
or |proff
Function: Turns printer on or off.

Although the Amstrad allows the simple listing of Basic programs to a printer, it is not so easy to specify that normal screen output during the running of a particular program should be sent to the printer. To do this you must replace all PRINT statements within a program with PRINT[#].119. This can be quite tiresome, since it does not allow you to specify at the time of printing whether you require hardcopy or not.

Toolkit incorporates two simple commands which control screen output. If |pron is active, all output goes to both screen and printer. Printer output is turned off with |proff.

Like other Toolkit commands these may be incorporated within your own programs. For example you may wish to use a routine similar to the following:

```
100 print"Do you require hardcopy ?"
110 a$=get$
120 if a$="Y" or a$="y" then |pron
```

Of course you may also use |pron to print the output from Toolkit.

RENUM

Syntax: |renum,<<new start line>>,<<old start line>>,<<new increment>>,<<old end line>>
Function: Wholly or partially renumber a Basic program.

The |renum command in Amstrad Basic allows for three parameters to be used when renumbering a program. These are the same as the first three used in this Toolkit command; and in fact supplying less than the four parameters required causes a default to the renumber options provided by that command.

The extra facility provided by Toolkit's renumber is the ability to specify a last line number to be renumbered; thus uniquely defining any block that the user requires. The original three-parameter Amstrad command does not allow you to specify the last line number to be renumbered, and the routine always takes this to be the last line number of the entire program.

Toolkit's four-parameter renumber has built-in error checking, and will inform you if you have asked it to renumber in a way which would overwrite existing lines.

REPLACE

Syntax: |replace,<search string>,<<replace string>>,<<start line>>,<<end line>>
CPC464 users should type the command name only; the parameters will be prompted for.
Function: Search for occurrences of a given string and replace it with another.

This routine will search for all the occurrences of the specified search string and replace it with the specified replace string. If the optional line numbers are supplied, it will operate only on the specified range of Basic lines.

Once the command and parameters have been supplied, Toolkit will prompt as follows:

```
        Press K for keywords:
        Global or selective:
```

Enter **K** if you wish to search for Basic keywords, otherwise press **ENTER**. Then reply with **G** or **S** to denote a global or selective search. Pressing **ENTER** at this point defaults to a global search.

In a global replace, Toolkit will replace all the occurrences of the search string with the replace string within the range of program lines specified (or the whole program if no line numbers are given).

In a selective search, as each find is made the routine prints the line to the screen, and prompts whether the particular string should be replaced or not. Only if the user enters "Y", will the swap be made. In either case, the routine continues its search until it reaches the end of the program, or the last line number if specified. Any further finds will again give rise to the prompt, and the user will be able to specify whether that particular occurrence should be replaced.

All searches are case specific. This means that if you specify a search string of "File", the routine will ignore any occurrences of "FILE", or "file" etc.

Two types of wildcard are however allowed:
"?" may be used to match any single character
"*" may be used to match any group of characters.

Thus if you search for b?g$, then the routine will find big$, bug$ and so on. Alternatively, using the other wildcard, a search for b*g$ would throw up all the finds above, plus many more, including beebug$, etc. Used intelligently this system of wildcards makes the replace (and the search option, in which they also feature) a very powerful one.

If you wish to use |replace to delete the occurrences of a certain string, you should enter a null string for the replace string; ie just press ENTER when the replace parameter is requested.

Please note that because of the way in which Amstrad Basic (quite correctly) tokenises all Basic keywords, you may not use the Toolkit |replace routine to work on strings within a Basic keyword.

For example, if you request all occurrences of INT to be replaced with say OGRAM, you will not find any PRINT statements altered; though if the word PRINT appeared as text, then it would indeed be altered to PROGRAM.

Having said this, it is possible to search for Basic keywords in their entirety by selecting K when prompted after the command has been called.

RESET

Syntax: |reset
Function: Resets the chief machine parameters to their value at start-up.

This command produces a soft reset, resetting machine parameters such as mode, pen and ink colours to their value at start-up, and clears the screen. But it does not clear Toolkit or any Basic program resident in the machine.

ROM

Syntax: |rom,<<rom number>>
Function: Gives details of roms present in the machine.

If used without a following parameter, the utility displays on the screen a table of all roms present in the machine. This takes the following form, giving the socket number, the name of the rom, its version number, the rom type (whether foreground or background), and its size:

```
    Rom Ø       BASIC          1.Ø      (F)      16k
    Rom 7       CPM ROM        Ø.5Ø     (B)      16k
```

If a rom number is supplied as a parameter, the routine will display all commands listed within that particular rom.

You should note that not all roms have a command listing that is readable by this routine.

RSX

Syntax: |rsx
Function: List resident system extensions.

This command lists the so-called resident system extensions. These are the bar commands which are resident in your machine's ram memory. Thus if you have loaded Toolkit from cassette or disc, this command will list Toolkit's commands. If Toolkit is in rom, and you have other extension routines in memory, then |rsx will list them.

SEARCH

Syntax: |search,<search string>,<<start line>>,<<end line>>
CPC464 users should enter the command name only; parameters will be prompted for.
Function: Display all occurrences of a given string.

This option works in exactly the same manner as the |replace option described above. The parameters are the same, and have the same effect, except that no replace string is required, since matching strings are displayed, but left untouched.

The two wildcards described under |replace, above may also be used, and have exactly the same effect.

START

Syntax: |start,<address>
Function: Alters the start address of a Basic program in memory.

Amstrad Basic normally assumes that any resident Basic program will be located at &170. By altering this address using the |start command, it is possible to persuade Basic to use a location of your choice.

Of course you cannot have a program resident in your machine, and then alter |start and hope for the best. Basic would simply lose track of your program, and may even corrupt it in the process.

There are two ways to use this command. One is to alter Start before loading in your program; then when you load it in, Basic will load it to the new address.

Secondly, you could move a resident Basic program with the |bmove command described above, which automatically adjusts the value of Start to point to the relocated program.

In either case, the most interesting use of the command is probably to put more than one Basic program into your machine at the same time (by successively loading and then moving) and then to switch between them by altering Start.

TRON / TROFF

Syntax: |**tron,<<start line>>,<<end line,<<x coordinate>>,<<y coordinate>>**
or |troff
Function: Switch on or off enhanced Trace facility.

The Trace facility implemented in Amstrad Basic, is of limited use in program debugging. Toolkit's Trace functions are a little easier to use.

If you set trace on (by typing |tron ENTER), and then run a program you will see a continuously changing number appear in the top left hand corner of the screen. This, as you would expect gives the line number currently being executed.

The four optional parameters with |tron allow you to specify, if you wish, a start and end program line between which the trace will function. If none are given the default is taken to be the whole program.

You may also specify the coordinates of the position at which the trace information will be printed on the screen. In this way you can avoid overprinting an area of particular interest to you. The graphics coordinates used in this command are the same as used in Amstrad Basic for printing text. Thus the range in mode 1 is 1-40 horizontal, and 1-25 vertical: the origin is the top left hand corner, and corresponds to coordinates 1,1. You should note however, that if you wish to specify coordinates, you must also put in start and end line numbers; otherwise Toolkit will take your coordinates to be line numbers.

If no trace coordinate parameters are entered Toolkit will place the trace display at the top left of the screen.

Slow motion and pause

When trace is enabled the user is given two extra controls over his program to assist with debugging. Holding down the SHIFT key will slow down the rate at which the program executes, and pressing the SPACE BAR will temporarily halt the program.

XREF

Syntax: |xref
Function: Provides extensive information about Basic variables.

When xref is called, the user is prompted as follows:

```
Enter Options:
  1. Numeric Variables
  2. String Variables
  3. Arrays
  4. Functions
  5. GOSUBs
  6. Send output to printer
Press ENTER to start:
```

You may choose any combination of the six menu items, pressing ENTER at the end. Toolkit will supply commas to separate each of the numbers entered. If you type "12" (choosing to display data on both numeric and string variables) and press ENTER, you will see a display of the following kind:

```
op = 13
  130      150        1860        2000
D1$ = "Title"
  30        40         50
box = Undefined
  100      150             and so on
```

In this display, op, D1$ and box are all variables. The first has a current value of 13, the second is a string variable currently set to "Title", while the third is as yet undefined.

The variable op is referred to on program lines 130, 150, 1860, 2000 and so on, and this cross referencing is what gives this powerful utility its name.

If you rerun lxref, selecting options 3, 4 or 5 you will get a similar display; but this time for either arrays (giving the values of all elements!), functions or GOSUBs, depending on the option chosen.

By adding "6" to your choice of options you will produce a printout of the cross referencer's output.

Note that to prevent the cross referencer displays from scrolling, as with all other Toolkit displays, you should press ESCAPE. Pressing any other key will continue the scrolling display.

COMMAND SUMMARY

HELP
Syntax: |help
Function: General help page giving command list, and syntax.

TOOLS
Syntax: |tools
Function: Displays a menu from which almost all of Toolkit's commands may be selected, and sets the function keys for use with Toolkit.

TOOLSOFF
Syntax: |toolsoff
Function: Clears Toolkit from memory.

BMOVE
Syntax: |bmove,<address>
Function: moves a Basic program in memory to a different address.

DUMPA / DUMPE
Syntax: |dumpa
or |dumpe
Function: 16 tone screen dump for Amstrad or Epson printers.

EMEM
Syntax: |emem,<address>,<<rom number>>
Function: Display and allow editing of the contents of memory.

FORMAT
Syntax: |format
Function: Format a disc

FREE
Syntax: |free
Function: Gives a set of status information.

KON / KOFF
Syntax: |kon
or |koff
Function: Turns on or off Toolkit's abbreviated keyword entry feature.

KEY
Syntax: |key,<<first key>>,<<last key>>
Function: Lists the function key definitions in a form in which they can be edited.

KEYDEF
Syntax: |keydef,<<first key>>,<<last key>>
Function: Lists what codes any key or group of keys have been set to.

LIST
Syntax: |list,<filename>,<<start line no>>,<<end line no>>
CPC464 users should type the command name only; the parameters will be prompted for.
Function: Lists a program directly from cassette or disc without affecting any program in memory.

LCOPY
Syntax: |lcopy,<start line>,<end line>,<new start line>
Function: Copy a set of Basic lines to a new position.

LMOVE
Syntax: |lmove,<start line>,<end line>,<new start line>
Function: Move a set of Basic lines to a new position.

PACK
Syntax: |pack
Function: Compacts a Basic program.

PARTSAVE
Syntax: |partsave,<filename>,<<start line>>,<<end line>>
CPC464 users should type the command name only; the parameters will be prompted for.
Function: Saves a specified part of a program to cassette or disc.

PMEM
Syntax: |pmem,<start address>,<end address>,<<rom number>>
Function: Dumps memory block to printer.

PRON / PROFF
Syntax: |pron
      or |proff
Function: Turns printer on or off.

RENUM

Syntax: |renum,<<new start line>>,<<old start line>>,<<new increment>>,<<old end line>>
Function: Wholly or partially renumber Basic program.

REPLACE

Syntax: |replace,<search string>,<<replace string>>,<<start line>>,<<end line>>
CPC464 users should type the command name only; the parameters will be prompted for.
Function: Search for occurrences of a given string and replace it with another.

RESET

Syntax: |reset
Function: Resets the chief machine parameters to their value at start-up.

ROM

Syntax: |rom,<<rom number>>
Function: Gives details of roms present in the machine.

RSX

Syntax: |rsx
Function: List resident system extensions.

SEARCH

Syntax: |search,<search string>,<<start line>>,<<end line>>
CPC464 users should enter the command name only; parameters will be prompted for.
Function: Display all occurrences of a given string.

START

Syntax: |start,<address>
Function: Alters the start address of a Basic program in memory.

TRON / TROFF

Syntax: |tron,<<start line>>,<<end line,<<x coordinate>>,<<y coordinate>>
or |troff
Function: Switch on or off enhanced Trace facility.

XREF

Syntax: |xref
Function: Provides extensive information about Basic variables.

BEEBUGSOFT FOR THE AMSTRAD

Beebugsoft has earned the reputation of producing quality 'Serious Software' for the BBC Micro. With this background we have now launched a range of utilities and tools for the Amstrad computers.

Our Amstrad range includes:

- **TOOLKIT**

  A whole host of programming tools for the Basic programmer, written in machine code and supplied on tape, disc or rom. Toolkit will save hours of time in programming and will put at your fingertips over 30 new commands, including:

  A program compactor, printer dumps, search & replace, keyword abbreviation (L. for list etc), partsave, Basic status, memory editor and much much more.

- **ULTRABASE**

  A general purpose file management package, supplied on disc or tape, allowing large amounts of information to be stored and processed. It is extremely powerful yet flexible and easy to use.

  Once set up, the information may be retrieved, sorted on any field, displayed, updated, printed etc as required.

- **REMBRANDT**

  An exciting new 16 colour painting and design package, controlled by icons. It allows you to create amazing screens on the Amstrad computer using some very advanced features. All options are selected from an on-screen icon menu and are extremely easy to use. This must be the ultimate drawing package for the Amstrad. Supplied on disc or tape.

- **BEEBUGSOFT**

  For further information or a technical specification on any of our products, please write to:

The Software Manager, Beebugsoft, Dolphin Place, Holywell Hill, St. Albans, Herts. AL1 1EX.

Alternatively, you may call our telephone hotline service on St. Albans (0727) 40303

BEEBUG SOFT

FOR THE AMSTRAD