SLOGGER

Slogger Ltd
215 Beacon Road
Chatham
KENT ME5 7BU

Elkman
(Sideways Rom Manager)
User Guide

Written by H.A.L. Harper-Wilkingson
and A.T.J. Hilbig

CONTENTS

1. INTRODUCTION

ELKMAN is a unique and versatile program which allows the user of an Electron computer to manage the 'sideways' ROMs on his system. *Any* ROM can be enabled or disabled, even on a <CTRL><BREAK>, thus removing the problems associated with the clashing of command names. Even ELKMAN itself may be disabled in this way.

ELKMAN offers full support of RAM devices (8K or 16K) which may be fitted in your ROM extension unit. Any ROM may be saved to tape or disc (if fitted) and later loaded back into sideways RAM memory...*both* by a single command. Thus allowing back-ups of all your ROM software.

ELKMAN will report the names of all ROMs present in your system, giving details of the size of each ROM (4K, 8K or 16K) and *even* indicates whether the device is ROM, RAM or empty. Data in ROM or memory can also be 'peeked', 'poked', transferred from one area to the other, or simply checked for consistency with the checksum facility. ELKMAN also provides facilities to display the contents and also the status of all sixteen function keys for editing purposes.

ELKMAN allows you to identify ROMs by using their name, which may be abbreviated for convenience, or by giving the 'sideways' page number of the ROM.

In all, the comprehensive ROM management facilities offered makes this ROM essential if you have several ROMs in your machine.

2. INSTALLATION

2.1. INSERTING THE CHIP

1. Before taking the chip out of its protective packaging, identify Pin 1 on the chip. It is marked with a dot on the top, in the corner of Pin 1, or the half moon notch at one end of the chip identifies the end of the chip nearest Pin 1. Pin 1 should be on the left if the notch is held up.

2. Hold the ends of the chip between finger and thumb, and line up the notch and all the pins over the notch and contacts of the destination socket.

3. Now apply firm and even pressure to the chip, but try not to force it! When the chip is in, it appears to be slightly raised. Check that all the pins do enter the socket, and that none are bent out or underneath.

With previously unused chips it may be difficult to insert them due to their legs being slightly splayed out. The legs can be aligned by carefully resting the rows of pins on a flat surface and applying slight pressure, first to one side then the other.

2.2. REMOVING CHIPS

Unless a special socket called a zero insertion force socket (Z.I.F.) has been fitted, removal of a chip is accomplished as follows:

1. Switch the power to the computer off, taking care that you will not lose important data in memory or any sideways RAMs which may be fitted.

2. Gently lever out the chip you wish to remove first from one end and then the other, taking care not to bend the legs of the chip.

3. Once removed, the chip should be placed somewhere safe to avoid it being unduly handled or damaged.

2.3. SIDEWAYS ROMs OPERATING PRINCIPLES

The Operating System allows 16 sideways pages, identified as pages 0 through to 15, and gives them an operating priority which decreases from sideways page 15.

In other words, when you enter a command the ROM in page 15 is offered the command first to see if it is recognised. If the ROM does not recognise the command then it is offered to the ROM in page 14 and so on.

When installing the ELKMAN ROM it is recommended that the chip be placed in the highest possible sideways page, normally page 15, in order to give it maximum priority over the other ROMs. This will

ensure ELKMAN's ability to 'kill' any ROM in your system; ELKMAN cannot 'kill' a ROM which is in a higher sideways page than itself.

2.4. TESTING INSTALLATION

Once you have fitted the ELKMAN ROM, cautiously turn on the power to the computer. If the computer does not come up with its normal sign on message then turn off immediately and recheck the installation of the chip.

Now type *HELP ELKMAN and press <RETURN>. The screen display should be the same as indicated in section 3.1 for correct installation.

If this is not the case there may be poor electrical contact between the pins of the chip and the socket and you should install the chip again.

3. ELKMAN OVERVIEW

3.1. HELP MENU

If you wish to list all available ELKMAN commands and their corresponding syntax, type *HELP ELKMAN and press <RETURN>. ELKMAN will then respond as follows:

```
ElkMan 1.0 (C) Slogger 1985
  PROMS   ((id))
  PSIZE   (id)
  RAMCLR  (id)
  OFFROM  (id)
  ONROM   (id)
  KILLROM (id)
  RSAVE   (id) (filename)
  RLOAD   (filename) (id)
  CRCSUM  (st addr) (end addr) ((id))
  ROMMEM  (id) (dest addr)
  MEMROM  (st addr) (id)
  PEEK    (st addr) (end addr) ((id))
  POKE    (st addr) ((id)) (bytes)
  ILIST   (st addr) (end addr) ((id))
  PKEY    ((n))
  KLIST

OS 1.20
```

The syntax parameters are defined as follows:

(id)	Identifies the ROM by name or sideways page
((id))	As above, but optional
(st addr)	Start of block address
(end addr)	End of block address
(dest addr)	Destination address
(filename)	Filename for load or save to Filing system
(bytes)	Data to POKE to memory
((n))	A number in the range 0 to 15

3.2. ELKMAN COMMANDS

ELKMAN is a service ROM. This means that all its commands can be accessed whilst operating from another ROM by the normal 'star' means. i.e. *PROMS or *RAMCLR etc. and you can even use them in your BASIC programs.

3.2.1. MINIMUM ABBREVIATIONS

All ELKMAN's commands may be abbreviated for convenience in the normal way. For example, typing *CRC. and *KL. is the same as typing *CRCSUM and *KLIST respectively.

The minimum abbreviation required for a command depends on the number of characters required to make the command string explicit and unique to ELKMAN. Normally, a command may be abbreviated to the first two characters,

Example:	*CR.	*KL.	*PR.
for	*CRCSUM	*KLIST	*PROMS

but this depends on clashes with the commands of other ROMs present.

As an example, suppose another ROM present in your computer had *PRINT and *KLENE as commands and also had a higher priority than ELKMAN (Section 2.3). Then if you type *PR. or *KL. for *PROMS and *KLIST, as before, you would find they are intercepted by this ROM. in fact, in this case the minimum abbreviation for these ELKMAN commands is *PRO. and *KLI.

3.2.2. DELIMITERS

Any of the following characters may be used to separate command line parameters:

space ! " £ $ % & ' ( ) * + , - . /

This is a most useful facility, particularly when you need to include the usual 'space' delimiter in an ASCII string, as in declaring the name of a ROM or POKEing to memory.

Example:

```
*RSIZE"DISK MAN"   ....if DISK MAN was not enclosed in quotes then
                       ELKMAN would interpret the command as *RSIZE
                       DISK

*CRCSUM/2000/5FFF/
*CRC./2000/5FFF/   ....same as above
*CRC./2000/5FFF    ....same as above
*CRC.2000 5FFF     ....same as above

*POKE/2000/@HARRY'S PROGRAM/
```

3.2.3. IDENTIFYING A ROM/SIDEWAYS PAGE

(id) identifies the ROM. ROMs may be identified by name or by sideways page number. Names may be abbreviated, ie BASIC may be abbreviated to 'BAS.'. If referenced by sideways page number then the format is Rx, where x is a number in the range 0 to 15.

Example:

If 'BASIC' and 'DISK MAN' ROMs were in sideways pages 11 and 12 respectively, then you may reference them in any of the following ways:

```
 *PROMS BASIC ..... using full name
 *PROMS BAS. ..... using abbreviated name
 *PROMS R11 ..... using sideways page

 *PROMS"DISK MAN" ..... using full name
 *PROMS DISK. ..... using abbreviated name
 *PROMS R12 ..... using sideways page
```

3.2.4. DECLARING ADDRESSES

All addresses declared must be hexadecimal and in the range 0 to FFFF.

Example:

```
 *CRCSUM E00 1900
 *CRCSUM C000 R15
 *PEEK E 1F
```

3.2.5. FILENAMES

ELKMAN allows you to enter up to twelve characters for filenames as used in *RLOAD and *RSAVE. For the Tape Filing System any more than ten characters will produce an error.

If a Disk System is fitted then the twelve character filename provides maximum flexibility by allowing the destination drive and directory to be specified as part of the whole filename.

Example:

```
 *RSAVE R9 :2.$.STARWRD ..... drive, directory and filename
 *RLOAD W.STARMON R15 ..... directory and filename
 *RLOAD ADDCOMM R15 ..... filename only
```

3.3 CLASHES OF COMMAND NAMES

Sometimes two or more ROMs may share the same command name for differing functions. For example, *REPORT in one ROM may display its copyright message, whilst in another ROM it could have a completely different meaning!

In such cases you will never have access to both functions because of the ROM operating priorities (section 2.3). The highest priority ROM will always intercept the command ahead of the intended ROM.

To overcome this limitation, ELKMAN allows you to issue a '*' command to a particular ROM by preceding the command with the identification of the intended ROM (Section 3.2.3).

Examples:

```
 *TELETEXT REPORT .......... issue *REPORT to TELETEXT ROM
 *TELET. REPORT ............ as above
 *TELET. REP. .............. as above
 *R12 REPORT ............... as above, but using sideways page number
 *R12 REP. ................. as above

 *R4 EDWORD 40 ............. issue *EDWORD 40 to ROM 4
 *R10 PEEK RAMDISK ......... issue *PEEK RAMDISK to ROM 10
```

3.4. MEMORY USAGE

ELKMAN's memory usage is restricted to Operating System allocated workspace and use of the bottom of the stack on a temporary basis during command execution. Three locations in this workspace are reserved as private memory in order to provide the *KILLROM facility.

The Operating System hi-water mark (OSWHWM) and BASIC's PAGE are not altered.

Sometimes there may be a contention between ELKMAN and another ROM for this private workspace. When this occurs, ELKMAN sacrifices its use of these locations in favour of the other ROM with the resulting loss of the *KILLROM facility.

ELKMAN will make you aware of this loss by showing 'O' instead of 'K' for disabled ROMs in the *PROMS display.

To restore the *KILLROM facility, enter: *ELKMAN

3. 5. ERROR MESSAGES

If you make a mistake, ELKMAN will tell you what is wrong as best it can by printing a message on the screen.

If ELKMAN is being accessed from a program, these messages may be suppressed to just return an error number which indicates the type of error. The program can then pick up this number and use it as an indicator of what guidance is needed to overcome the mistake.

Messages are listed below in alphabetical order together with their error numbers in hexadecimal (base 16).

FC Bad address
An invalid hexadecimal address was given on the command line. This can indicate a bad hexadecimal digit or an address greater than &FFFF.

02 Bad byte

An invalid byte was given on the command line. Byte values must be in the range 0 to &FF.

1E Current language!
An attempt was made to disable the currently selected language. This could cause the computer to crash if ELKMAN was to execute this command.

1A No such ROM
The ROM name given could not be found.

10 Syntax error
The parameters declared on the command line are not what was expected. Check command syntax by typing *HELP ELKMAN and press RETURN.

4. ELKMAN COMMANDS

4.1. DISPLAY SIDEWAYS PAGE STATUS

*PROMS ((id))

This command will display status information for a ROM position as defined by the (id). If an (id) is not entered, then the status will be given for each of the sixteen ROM pages on the Electron.

The format of the PROMS display is as follows:

```
a:bbbbb  (c)  dk  e
- -----   -   -   -
```

```
Where:    a          ROM page number
          bbbbb      ROM title
          c          ROM status: 0 = disabled
                     1 = enabled
                     K = killed
          d          size of ROM or RAM (4, 8 or 16K)
          e          "Ram" displayed if detected
```

```
Field bbbbb....BASIC...Basic is in this ROM
               Empty...No ROM present in this slot
               ?????...ELKMAN cannot recognise this ROM
```

The status indicates whether the ROM's presence is acknowledged by the Operating System. The status takes the following meanings:

(1) .... Performance of the ROM is not impaired

(0) .... Operating System calls to the ROM, such as responding to 'star' commands, are inhibited; by the ROM may still be active in the system for the purpose of claiming vectors and private workspace memory on a BREAK reset.

If private workspace memory is claimed then the Operating System's hi-water (OSWHWM) and BASIC's PAGE will be altered, resulting in less memory for your programs.

(K) .... A ROM present in this sideways page will not function and in so far as the computer is concerned is equivalent to removing the chip entirely.

Example:

```
 *PROMS
 15:?????          (K)  8k Ram
 14:ElkMan         (1)  8k
 13:STARWORD       (1) 16k
 12:DISK MAN       (1) 16k
 11:BASIC          (1) 16k
 10:BASIC          (K) 16k
  9:?????          (K) 16k
  8:?????          (K) 16k
  7:ADDCOMM        (K)  8k
  6:TOOLKIT        (1)  8k
```

```
  5:STARMON        (1)  8k
  4:GRAPHICS ROM   (0)  8k
  3:Empty          (K) 16k
  2:Empty          (K) 16k
  1:Empty          (K) 16k
  0:Empty          (K) 16k

 *PROMS STARWORD
 13:STARWORD       (1) 16k
```

4.2. DISPLAY ROM SIZE

*RSIZE (id)

The size of the ROM specified by the (id) is displayed by this command.

The format of the RSIZE display is as follows:

```
aaaaa   bbbb cccc dddd
-----   ---- ---- ----
```

Where	aaaaa	ROM title
	bbbb	start address of ROM
	cccc	end address of ROM
	dddd	used bytes length of ROM (Not displayed for RAM or Empty sockets)

Example:

```
 *RSIZE R5
 STARMON         8000 9FFF 2000

 *RSIZE STARWORD
 STARWORD        8000 BFFF 3168

 *RSIZE R15
 ?????           8000 9FFF
```

4.3. CLEAR SIDEWAYS RAM MEMORY

*RAMCLR (id)

This command is used to clear sideways RAM memory to data FF's. When cleared the Operating System no longer recognises a ROM program which may have occupied that Sideways RAM.

If a ROM program was currently occupying the sideways page before the clear and was not disabled (Sections 4.4 and 4.6), then you must reset the machine with a BREAK after the clear to prevent the system 'crashing'.

No error is returned if RAM is not present in the selected sideways page.

Example:

```
 *RAMCLR R15
 *RAMCLR STARWORD
```

4.4. DISABLE ROM FROM 'STAR' COMMANDS

*OFFROM (id)

Used to disable a ROM. This command is very useful when conflict occurs due to the same command name being used by another ROM. Conflicts can be avoided by disabling the offending ROM.

When disabled, the ROM will remain disabled, even on a CTRL-BREAK reset, until a *ONROM is issued.

Examples:

```
 *OFFROM STARMON
 *OFFROM R5
```

4.5. ENABLE PREVIOUSLY SAVED ROM

*ONROM (id)

Used to enable a ROM that has previously been disabled by *OFFROM or *KILLROM. You may need to reset the computer with a BREAK in order to allow the ROM to start up correctly.

Example:

```
 *ONROM STARMON
 *ONROM R5
```

4.6. DISABLE ROM FROM SYSTEM

*KILLROM (id)

This is a more severe way to disable a ROM than using *OFFROM. When a ROM is 'killed' it is equivalent to removing the chip from the computer.

Enter the command and press BREAK.

Example:

```
 *KILLROM STARMON
 *KILLROM R5
```

4.7. BACKUP ROM DATA TO TAPE OR DISK

*RSAVE (id) (filename)

This command is used to save the contents of any of the ROMs directly to a file on tape or disk depending on which filing system is active.

If the Disk Filing System is active then the memory about the Operating System hi=water mark (OSWHWM) is used as an intermediate area during the transfer to the file (similarly for *RLOAD) and so any programs in this area will be corrupted.

Example:

```
 *RSAVE STARMON STARMON
 *RSAVE STARMON :2.$.STARMON

 *RSAVE R5 STARMON
 *RSAVE R5 :2.$.STARMON
```

4.8. LOAD ROM DATA TO SIDEWAYS PAGE

*RLOAD (filename) (id)

Similar to *RSAVE, this command is used to load sideways RAM memory with ROM software direct from tape or disk.

Using *RSAVE and *RLOAD together allow you to hold backups of all your ROMs and load them into your computer when you require to use them during the session; hence avoiding tying up valuable ROM sockets.

Example:

```
 *RLOAD STARMON R15
 *RLOAD :2.$.STARMON R15
```

4.9. GENERATE CHECKSUM ON MEMORY

*CRCSUM (st addr) (end addr) ((id))

This command will produce a 16 bit checksum on memory; the area of the memory being specified by the address range inclusive.

If the range of memory declared encompasses the sideways page area (&8000-&BFFF) then a ROM (id) must be specified. If the (end addr) is not given then the end of ROM address is assumed. If no address is given a checksum on the entire ROM is performed.

Checksums provide a means of validating the consistency of data in memory and makes them ideal when searching for memory faults.

Checksums may also be used to identify programs which alter themselves when run. This is achieved by comparing the checksum on

the program before and after it is run. Those that alter themselves will produce different checksums.

Example:

```
 *CRCSUM 1900 3FFF
 EA12

 *CRCSUM 2000 8100 R15
 1F1A

 *CRCSUM 8000 8A00 STARWORD
 0648

 *CRCSUM R7
 D293
```

4.10. MOVE ROM DATA TO MEMORY

*ROMMEM (id) (dest addr)

This command is used to transfer the contents of a ROM into the memory of the computer. You cannot transfer directly from a ROM socket to sideways RAM memory, but this is readily achieved by using the combination *ROMMEM then *MEMROM.

Example:

```
 *ROMMEM STARWORD 2000
 *ROMMEM R7 1900
```

4.11. MOVE MEMORY TO SIDEWAYS PAGE

*MEMROM (st addr) (id)

This command is used to transfer data from the computer memory to sideways RAM memory. No error is returned if an attempt is made to write to ROM.

Example:

```
 *MEMROM 2000 R15
 *MEMROM 1900 STARWORD
```

4.12. DUMP MEMORY IN HEX AND ASCII FORMAT

*PEEK (st addr) (end addr) ((id))

This command allows you to look at the content of any sideways page of memory. If the range of memory declared encompasses the sideways page area (&8000-&BFFF) then a ROM (id) must be specified.

Eight bytes are displayed on each line in both hexadecimal and ASCII format. If the (end addr) is not given then the end of ROM address is used. If no address is given then the entire ROM contents are scrolled to the screen.

Page mode may be enabled before the execution of the command with a CTRL-N and disabled afterwards by a CTRL-O. Alternatively, the SHIFT and CTRL keys may be pressed simultaneously during the output to the screen to achieve a similar effect.

Example:

```
 *PEEK 8000 8010 BASIC
 8000 C9 01 F0 1F 60 EA 60 0E I.p.'j'.
 8008 01 42 41 53 49 43 00 28 .BASIC.(
 8010 43 ** ** ** ** ** ** ** C.......

 *PEEK 1900 190A
 1900 02 0E 02 0E 00 0C 00 03 ........
 1908 FD 00 FF ** ** ** ** ** ........

 *PEEK BASIC
```

4.13. SET MEMORY LOCATIONS

*POKE (st addr) ((id)) (bytes)

This command allows you to set locations in memory or sideways RAM from a hexadecimal or ASCII string. ASCII strings must be preceded by a '@'.

If the range of memory to be written encompasses the sideways page area (&8000-&BFFF) then a ROM (id) must be specified.

Example:

```
 *POKE 2000 @HARRY ..................... write ASCII string
 *POKE/2000/@HARRY'S PROGRAM/
 *POKE/2000/@HARRY'S PROGRAM ........... write hexadecimal string

 where:    2000 is set to 48
           2001 is set to 41
           2002 is set to 52
           2003 is set to 52
           2004 is set to 59

 *POKE 8000 R14 A9 5 AE 2 AC FF 20 F4 FF

 where:    8000 is set to A9
           8001 is set to 05
           8002 is set to AE
           8003 is set to 02
           8004 is set to AC
           8005 is set to FF
           8006 is set to 20
           8007 is set to F4
           8008 is set to FF
```

4.14. DUMP MEMORY IN 6502 MNEMONICS

*ILIST (st addr) (end addr) ((id))

This command allows ROMs and areas of memory to be disassembled into 6502 instruction mnemonics. Scrolling the screen may be controlled as for *PEEK.

If the range of memory to be disassembled encompasses the sideways page area (&8000-&BFFF) then a ROM (id) must be specified.

Example:

```
 *ILIST BASIC
 8000 C9 01     CMP #01      I.
 8002 F0 1F     BEQ &8023    p.
 8004 60        RTS
 8005 EA        NOP          j
 8006 60        RTS
 8007 0E 01 42  ASL &4201    ..B
 800A 41 53     EOR (&53,X)  AS
 800C 49 43     EOR #&43     IC
 800E 00        BRK          .
 800F 28        PLP          (
 8010 43        ???          C

 *ILIST 2072 207A
 2072 48        PHA          H
 2073 08        PHP          .
 2074 78        SEI          x
 2075 85 EF     STA &EF      .o
 2077 86 F0     STX &F0      .p
 2079 84 F1     STY &F1      .q
```

4.15. PRINT FUNCTION KEY STRING

*PKEY ((n))

Print the contents of the function keys. When function keys are programmed it is often so difficult to remember what was actually written into them. With the help of this command your troubles are over.

The number 'n' must be in the range 0 to 15. If omitted then the contents of all active function keys are displayed.

This facility is very useful for editing the function keys. To modify function key 0 for instance, simply enter *PKEY 0 and use the cursor editing keys to do the appropriate modification.

Example:

```
 *PKEY 10
 OLD|MRUN|M

 *PKEY
 Key  0 PRINT A,B,C|M
```

```
Key 10 OLD|MRUN|M
Key 11 *SAVE FRED 2000 +1FF|M
```

4.16. DISPLAY FUNCTION KEYS STATUS

*KLIST

This command will indicate the status of each of the 16 function keys.

The first of the returned information will give (on) if the cursor keys are active, (off) if the keys have been disabled by a *FX225,0 command, or (decimal value) if the keys have been reprogrammed to a different base (Section 5.4), the decimal value being the new base.

Unused space is the space remaining for further function key strings.

Example:

```
*KLIST
(on) Unused space 210
 0:in use
 1:in use
 2:
 3:
 4:
 5:
 6:
 7:
 8:
 9:
10:in use
11:in use
12:
13:
14:in use
15:
```

5. APPLICABLE OPERATING SYSTEM *FX CALLS

5.1. PROGRAMMING THE CURSOR EDITING KEYS

The cursor editing keys may be disabled and programmed in the same way as the user-defined function keys so that they may contain strings by typing *FX 4,2. When programmed in this way these keys have the following *KEY numbers:

```
COPY  11
LEFT  12
RIGHT 13
DOWN  14
UP    15
```

Example:

```
*FX4,2
*KEY 11"THIS STRING IS PRINTED BY THE COPY KEY"
*KEY 12"THIS STRING IS PRINTED BY THE CURSOR LEFT KEY"
```

Now press the COPY or the cursor LEFT keys to print the above strings.

5.2. RESTORE CURSOR EDITING KEYS

*FX 4,0 resets the system so that the cursor editing keys perform their normal cursor editing function. This call should be made to restore the keys after a *FX4,2.

5.3. PROGRAMMING THE FUNCTION KEYS

The user-defined function keys may readily be programmed to contain strings by using the Operating System's *KEY facility. When programmed the keys will print the string entered when pressed.

Example:

```
*KEY 10"OLD|MRUN|M" ..................... programs the BREAK key.
*KEY 11"*SAVE FRED 2000 +1FF|M" ......... programs the COPY key.
*KEY 0"PRINT A,B,C|M" ................... programs f0
```

5.4. DISABLE FUNCTION KEYS

The user-defined function keys may be disabled from producing a string of characters and either made to produce a single ASCII code or have no effect. The statement

```
*FX225,value
```

would cause key 'f0' to produce ASCII code 'value', 'f1' to produce 'value+1' and so on to 'f9' which would produce 'value+9'.

Example:

```
 *FX225,128 ....... makes the keys produce Teletext effects
 *FX225,0 ......... makes the keys have no effect
```

5.5. RESTORE FUNCTION KEYS

*FX 225,1 restores the function keys to their normal function of generating strings.

6. COMMAND SUMMARY

Review (Electron User)

ELKMAN is a sideways ROM designed for use with an external ROM expansion board such as Slogger's own ROMBOX (Not the Plus 1), and is identical to the ROMs that BBC owners have been using for years.

ELKMAN is a ROM manager so needs to have priority over all other ROMs present to operate properly. This means that it is best placed so that it appears as ROM 15 to the operating system.

Placing it in the rightmost socket on Slogger's ROMBOX achieves this. You'll have to check the manual on other systems. ELKMAN is a service ROM, which means that all its commands are available while another ROM is in use, using a * command. These commands can even be used within a Basic program.

While writing this review using View I can test each function without leaving the Word Processor.

*HELP ELKMAN reveals the ROMs 16 commands and their syntax. One of the simplest is *PROMS which lists all the ROMs present, their state and size. ROMs can be in one of three states. They are either on, off or killed. *OFFROM and *ONROM can be used to enable or disable a ROM.

If it has been disabled it will not respond to any commands and cannot be used. This is useful if two ROMs have the same name for different commands. The offending ROM taking the command can be switched off. Even though a ROM may be off, it can still reserve memory. *KILLROM is equivalent to physically removing a ROM. I found it useful for disabling the Plus 3 when playing games on tape.

*PEEK is a memory lister which can be used to display any section of memory, even sideways ROMs. The output is in hexadecimal and ASCII. *POKE will place a series of bytes or a string anywhere in memory.

ELKMAN contains a complete 6502 disassembler, which again is capable of operating on sideways ROMs. The hex address, object code, mnemonics and ASCII codes are listed.

There are several commands which operate on sideways RAM. These can clear the RAM if fitted, load it with data from memory, tape or disc, and save it to memory tape or disc.

ELKMAN is well written and simple to use. The documentation is excellent. It comes with a very smart 21-page manual which explains fitting and use in a clear and easy-to-read manner.

Even if it's the only ROM you have, you'll still find most of the utilities useful. I can recommend ELKMAN to all serious Electron users.

Roland Waddilove, ELECTRON USER 2.10