A.R.M.

The Advanced ROM Manager Manual

NOTICE

Advanced Computer Products Ltd,
6 Ava House,
High Street,
CHOBHAM,
Surrey GU24 8LZ.

Tel. (0276) 76545

This product is not intended to operate in sideways RAM.

CONTENTS

COMMAND SUMMARY

*AUTOROM	- makes a program run from ROM
*GOROM	- execute a machine code subroutine in a ROM
*KILL	- turn off a ROM
*MAKEROM	- create ROM files for the ROM filing system
*OFFER	- send a command to a specific ROM
*RDUMP	- display the contents of a ROM
*PYREX	- examine/edit the contents of a ROM/RAM
*CARLOAD	- load a file into sideways RAM
*RMOVE	- copy a ROM into memory
*ROMS	- catalogue the sideways ROM sockets
*RSAVE	- save a ROM to a file
*RSUM	- display a ROM checksum and CRC bytes

1 FITTING ARM

For the BBC Computer

After turning off the power, take off the top cover of the computer by removing the four fixing screws located at the rear of the computer and on the underside near the front.

Now locate the ROM sockets. In the BBC B+ they are situated to the right of the power supply. The two sockets nearest the front of the computer are used by the speech synthesizer and should not be used. In the BBC B the ROM sockets are situated beneath the keyboard, so remove the four screws securing the keyboard to the computer and move carefully to one side just enough to expose the four ROM sockets located on the right side of the board.

Insert the **ARM** ROM into one of the empty ROM sockets, with the little notch on one end of the chip facing the back of the computer. Take care not to bend any of the legs on the chip. On the BBC B screw the keyboard back on to the computer. Replace the top cover.

Turn on the computer and type ***HELP ARM**. A list of commands should appear on the screen. If no list appears it is likely the ROM is not positioned correctly in its socket.

For the Electron

The ARM ROM must be fitted to the Plus 1 via a suitable ROM carrier board, such as the ROM adaptor boards supplied by ACP.

Fitting the ROM cartridge

Turn off the power and insert the ROM cartridge into one of the slots in the top of the Plus l. Ensure the label on the cartridge faces the keyboard. If in doubt refer to "Using cartridges" on page 16 of the Plus 1 User Guide.

Turn on the power to the computer and type ***HELP ARM**. A list of commands should appear on the screen. If no list appears it is likely the ROM cartridge is not correctly positioned in the Plus l.

2 ARM COMMANDS

Type ***HELP ARM**

This shows a list of all the commands. Their parameters are shown in brackets after the command. A parameter is a name, number, address or a letter which the command needs to operate correctly. Some parameters are optional, meaning the command will supply its own number or address. Optional parameters are shown in round brackets, eg.

(<start>), (<end>), (<dest>), etc.

If a command is typed in with a parameter missing a message will be displayed showing the correct syntax for the command, eg.

SYNTAX : REX <rom> (<start>) (<b>)

To avoid command names conflicting with other ROMs, **ARM** commands can be preceded with the letter A, eg. *AREX is the same as *REX and *ARSAVE is the same as *RSAVE.

Commands can be abbreviated by adding a full stop to the abbreviated name, so long as the name remains unique to ARM and other * commands, eg. *RL. is the same as *RLOAD and *RS. is the same as *RSAVE.

ARM commands will be accepted in both upper and lower case.

From now on, the BBC Microcomputer used with the second processor will be referred to as the I/O processor.

Parameter Definitions

In this manual & printed in front of a number means the number is in hexadecimal (hex), base 16, otherwise the number is in decimal, base 10.

<b> - bytes, 0-255.

Determines how data is printed on the screen, and how much on each a line. A byte can be printed as part of a hex number, an ascii character, or part of a disassembled line. The list below shows how a byte is per line for different values of <b>.

<b> = 0	65C02 diaassembler
<b> = 1-99	ascii + hex in <b> byte increments
<b> = 100	ascii in 32 byte increments
<b> = 101-199	ascii in <0> - 100 byte increments
<b> = 200	hex in 11 byte increments
<b> = 201-255	hex in <0> - 200 byte increments

<dest> - Destination memory address.
<end> - End memory address. See <start>

Sometimes it is easier to express an end memory address as the number of hex bytes from the start memory address. This can be done by

preceding the number with +, eg. ~100, +500, etc.

<fsp> - File specification, eg :1.$.PROG

<rom> - Sideways ROM number, 0-15.

<start> - Start memory address. (I/O processor only)

A memory address is a hex number from 0 to &FFFFFFFF, so called 32 bit addressing.

If a second processor is not connected addresses 0 to &FFFF will refer to memory in the I/O processor. The top two bytes are ignored, eg. address &FFFF1234 is the same as &1234.

If a second processor is connected addresses 0 to &FFFEFFFF will refer to memory in the second processor and addresses &FFFF0000 to &FFFFFFFF will refer to memory in the I/O processor. To summarise:

I/O Processor Only

addresses 0-&FFFF - I/O processor memory

I/O Processor and Second Processor

addresses 0-&FFFEFFFF - second processor memory

addresses &FFFF0000-&FFFFFFFF - I/O processor memory

For more information on I/O and second processor memory addresses refer to Chapter 9, 'Distinguishing between memories' in the 6502 Second Processor User Guide.

*AUTOROM <fsp> <title> <fsp>

Purpose

To create a ROM file of a program which can be run from ROM or sideways RAM. This command is useful far 'making ROMs' of frequently used programs. Instead of loading the program into memory from TAPE or DISC, the program can be loaded very quickly from ROM.

The first parameter <fsp> is the output file which can be programmed into an EPROM or loaded into sideways RAM using the CARLOAD command. The second parameter <title> is the title given to the ROM as displayed by *__HELP__. The maximum length of a title is 20 characters. The last parameter <fsp> is the program to be put into the ROM file. The maximum size of this file is &3E2B bytes, and only the first 10 characters of the filename are used in the ROM file.

Both BASIC and machine code programs can be 'ROMed'. The program is run by typing * followed by the name of the program (the last parameter above). The program is copied from ROM into memory at the programs LOAD address, and then RUN. Machine code programs are run from the programs execution address. If a program is copied below OSHWM, this is normally the value assigned to PAGE, then the TAPE filing system is selected.

This command identifies a BASIC program from its execution address. BASIC program execution addresses should be &8023, &801F or &802B.

Example

***AUTOROM MYROM "MY FIRST ROM" PROG1**

Creates a ROM file called MYROM from the file PROG1. The ROM title is "MY FIRST ROM".

Note

This command uses user RAM, so save any program in memory before using this command.

*GOROM <rom< <start> (A/X/Y)

Purpose

To start execution of a machine code subroutine in a ROM. <rom> is the ROM socket with the subroutine. <start> is the memory address of the subroutine in the ROM. A/X/Y, by default all 0, are the values assigned to the 6502 registers A,X, and Y on entry to the subroutine.

Example

***GOROM 15 9A45**

Starts execution of a subroutine in ROM 15 at address &9A43.

Note

This command can be used to start execution of a subroutine at any address in the I/O processor. This can be useful if a second processor is connected.

*KILL (<rom>)

Purpose

To 'kill' sideways ROM <rom>. This has the effect of turning a ROM off, without having to physically remove it from its socket. <rom> can be the title of the ROM as an alternative to the ROM number.

If <rom> is omitted from the command, each ROM title is displayed followed by :. Press **Y** to 'kill' the ROM or any other key to leave the ROM intact. If a ROM which has already been 'killed', is displayed and the response to the prompt is **N**, the ROM will be 'turned' back on.

Examples

***KILL 14**

To unplug the ROM in socket 14.

***KILL DFS**

To turn off DFS. The value of PAGE will not be effected until **BREAK** or **CTRL BREAK** is pressed.

Note

This command should be used with caution. Some ROMs can respond unpredictably to this command, and not produce the intended effect. Pressing **BREAK** or **CTRL BREAK** after killing a ROM will not turn it back on. To recover any 'lost' ROMs turn the machine off and then on, or use ***FX200,2** and then press **CTRL BREAK**. Type ***HELP** to see which ROMs are active in your computer.

*MAKEROM <fsp> <title> <fsp>

Purpose

To create ROM files for the ROM filing system. The first parameter <fsp> is the file which this command creates to store the ROM files. The second parameter <title> is the title given to the ROM, as displayed by a ***HELP**. The last parameter <fsp> is the file this command uses to make a ROM file. This parameter may be repeated to put more than one file into ROM. The file size of the first parameter <fsp> may not exceed 16K, so there is a limit to the size and number of ROM files created by this command.

The file <fsp> which this command creates could be programmed into an EPROM, and the file(s) could then be accessed from the ROM filing system. Alternatively, if' sideways RAM is available the file could be loaded into memory using the ***RLOAD** command.

The ROM filing system is very similar to the TAPE filing system, except files cannot be saved to ROM. To enter the ROM filing system type, ***ROM**. Filing system commands include:

*CAT
*LOAD file
LOAD "file"
*RUN file
CHAIN "file"
*EXEC file
*OPT1, n

Examples

***MAKEROM DEMOROM DEMO PROG**

Creates a file DEMOROM, with a title DEMO from the file PROG

***MAKEROM DEMOROM DEMO PROG1 PROG2 PROG3**

Creates a file DEMOROM, with a title DEMO from the files PROG1, PROG2 and PROG3.

Notes

This command uses user memory, so save any program in memory before using this command. A "No room" error message is displayed if the destination file size would exceed 16K, or if the user memory is not large enough to take the ROM files. If the user memory is not large enough select mode 7 (mode 6 on Electron) and try the command again. This command cannot be used in the TAPE filing system.

*OFFER <rom> <command>

Purpose

To send a *command to a specific ROM. <rom> is the ROM socket which the command is sent to. <command> is the *command.

This command is useful when two ROMs have the same command name for two different commands. Normally the command would come from the ROM which had the higher priority. Using ***OFFER**, the command can be sent to either ROM, regardless of priority.

Example

***OFFER 15 ROMS**

Sends the command *ROMS to ROM 15.

*RDUMP <rom> (<start>) (<end>) (<b>)

Purpose

To display the contents of a ROM. <rom> is the ROM socket from which memory is displayed. <start> is the first ROM address from which to display memory, by default &8000. <end> is the last ROM address from which to display memory, by default &8000 plus the size of the ROM. <b> determines the format of the display, by default 8, which displays memory as hex bytes and ascii characters in increments of 8 bytes (see page 2).

Examples

***RDUMP 15**

Displays memory from ROM 15 as hex and ascii.

***RDUMP 15 A000**

Displays memory from ROM 15 starting from address &A000.

***RDUMP 15 8000+4000 0**

Displays memory frcm ROM 15 in disassembler.

*REX <rom> (<start>) (<b>)

Purpose

To examine/edit the contents of a ROM/sideways RAM. <rom> is the ROM socket number from which memory is displayed. <start> is the start address from which memory is displayed. <b> determines the format of the display, by default 8, which displays memory as hex bytes and ascii characters in increments of 8 bytes (see page 2).

The cursor points to the current memory byte which is displayed as a hex byte surrounded by round brackets and an ascii character above a flashing cursor. Moving the cursor is done using the four cursor keys.

Moving forward or back a screen is done using the **SHIFT** cursor keys. To move the current memory byte to the top left of the window press **CTRL ^** (CTRL . on an Electron). To view memory from the next ROM socket press **CTRL R**.

If memory is displayed in disassembler it is possible to follow the address of a **JSR**, **JMP** or branch instruction by pressing **RETURN** whilst the memory pointer is over the instruction opcode. To return from a **JSR** press **CTRL X** or press **RETURN** whilst the memory pointer is over an **RTS** instruction.

Press **ESCAPE** to exit this command.

Editing keys

CURSOR keys	- cursor movement
SHIFT left cursor	- move cursor to left margin
SHIFT right cursor	- move cursor to right margin
SHIFT down cursor	- move one page down
SHIFT up cursor	- move one page up
TAB (CTRL I on Electron)	- cycle display format
CTRL ^ (CTRL . on Electron)	- home current memory byte
CTRL B	- change background colour
CTRL F	- change foreground colour
CTRL P	- print screen
CTRL R	- display next ROM socket
ESCAPE	- exit command

Disassembler Only

RETURN	- follow JMP, JSR or Branch or return from 3SR.
CTRL X	- return from JSR

Examples

***REX 15**

To examine ROM 15.

***REX 15 9000 0**

To examine ROM 15 in disassembler starting at address &9000.

*RLOAD <fsp> <rom>

Purpose

To load file <fsp> into sideways RAM socket <rom>. This command can only be used if sideways RAM is fitted in the computer.

Example

***RLOAD PROG 0**

Loads file PROG into sideways RAM socket 0.

Notes

This command uses user memory, so save any program in memory before using this command.

*RMOVE <rom> (<dest>) (<start>) (<end>)

Purpose

To copy all or part at a ROM to user RAM. <rom> is the ROM socket number to be copied. <dest> is the destination memory address, by default BASIC's value of PAGE. <start> is the start ROM address, by default &8000. <end> is the last ROM address to move, by default the end of the ROM.

Examples

***RMOVE 15**

Moves the contents of a ROM 15 to PAGE.

***RMOVE 15 2000**

Moves the contents of ROM 15 to address &2000.

***RMOVE 15 2000 8000+100**

Moves the first 256 bytes of ROM 15 to address &2000.

*ROMS (<rom>)

Purpose

To catalogue the sideways ROM socket <rom>, by default ROMs 0 to 15. <rom> can be the title of the ROM as an alternative to the ROM number. The catalogue of a ROM shows its ROM socket number, the size of the ROM (8K/16K), the type of program in the ROM, the title of the ROM and its version number, if it has one. A ROM can be a language shown as (L), a service ROM (like **ARM**) shown as (S), or both a language and a service ROM shown as (SL). ROMs which have had the **KILL** command used on them will appear as (**).

Examples

***ROMS**

Catalogues ROMs 0 to 15.

***ROMS 15**

Catalogues ROM socket 15.

***ROMS BASIC**

Catalogues the ROM socket containing BASIC.

***ROMS 12 13 14 15**

Catalogues ROM sockets 12, 13, 14 and 15.

*RSAVE <fsp> <rom>

Purpose

To save the contents of a ROM to the file <fsp>. <rom> is the ROM socket number from which memory is read.

Example

***RSAVE PROG 15**

Saves the contents of ROM socket 15 in a file PROG.

Notes

This command uses user memory, so save any program in memory before using this command.

*RSUM (<rom>)

Purpose

To print a ROMs checksum and CRC bytes. <rom> is the ROM socket number from which memory is read, by default sockets 0-15.

Examples

***RSUM**

Prints a checksum and CRC for ROMs 0-15.

***RSUM 15**

Prints a checksum and CRC for the ROM in socket 15.

***RSUM 0 1 2 5**

Prints a checksum and CRC for the ROMs in sockets 0, 1, 2 and 5.

Note

Calculating a checksum and CRC takes approximately 5 seconds for a 16K ROM on a BBC computer and a little longer on a Electron. The checksum and CRC are both two byte values. The checksum is simply the addition of all the bytes in the ROM, while the CRC (Cyclic Redundancy Check) is a little more complicated involving arithmetic shifts and EOR operations.

CRC algorithm:

```
H = C EOR H
FOR X=1 TO 8
   T=0
   IF (bit 7 of H = 1) THEN HL=HL EOR &0810: T=I
   HL=(HL*2+T) AND &FFFF
NEXT X
```

where H and L represent the high and low bytes of the CRC and C represents the character.

The above algorithm is not a BASIC program.