Brian Watson,
"Number Six",
Windmill Walk,
Sutton,
ELY
Cambs
CB6 2NH
ENGLAND

or

brian@spheroid.demon.co.uk

Manual scanned by Paul Collins.
OCRed by Kevin Thacker.
Many thanks to Brian Watson for his help with this manual.

UTOPIA

16k Utilities ROM

AMSTRAD CPC464
CPC664 & CPC6128

UTOPIA

16K UTILITIES ROM

AMSTRAD CPC464 CPC664 & CPC6128

Introduction

UTOPIA is a collection of useful commands relating to many aspects of Amstrad computing. Although sold as a single product. UTOPIA is really a 'library' of programs, all contained on a single ROM chip for convenience. Just like a library of books, UTOPIA's commands can be classified by subject:

1. Commands useful to BASIC programmers. These include find and replace, move lines, and various commands displaying useful information such as the currently defined variables.
2. Commands relating to files on tape or disc, such as TYPE, DUMP, VERIFY and COPY.
3. Disc users utilities. All the essential facilities for the disc user are contained in the ROM. With UTOPIA installed there is no longer any need to use your CP/M disc to format or copy a disc.
4. Commands relating to sideways ROMS and external commands. These allow all sideways ROMs and all external commands to be listed and also the switching off of selected ROMs.
5. Commands to echo all screen output to either the printer or a file.

The UTOPIA program was developed using the MAXAM assembler ROM.
This manual was printed from camera-ready copy written on the PROTEXT word processor and printed on a KAGA-TAXAN printer.
MAXAM and PROTEXT are both available in ROM from Arnor.

Arnor Ltd., The Studio. Ledbury Place. Croydon CR0 1ET. 01-688-6223

INSTALLATION

UTOPIA is supplied in a 16K EPROM which can be fitted into any available ROM expansion board. You may find that the legs of the EPROM will need straightening a little in order to fit it. Do this with extreme care by holding the EPROM in the centre (avoid touching the legs) and pressing the side of the legs against a flat surface. Keep the EPROM away from sources of electrostatic charge such as the monitor screen.

ROM numbers

Each ROM, when installed must have a unique ROM select number so that the firmware may access it. UTOPIA is a background ROM, which means it provides facilities that can be used by other ROMs, such as BASIC. The number associated with a background ROM must lie between 1 and 15 (with the 1.0 firmware as on the CPC464 it must be between 1 and 7). However UTOPIA must be installed with a select number between 1 and 6. This is because AMSDOS occupies ROM 7, and UTOPIA must have a lower number in order to intercept some of the AMSDOS commands. It you have the Arnor AD-1 adaptor with MAXAM or PROTEXT that will take ROM 5. Select one of the remaining numbers for UTOPIA - this will be done by choosing an appropriate socket and, possibly, fitting a link. Refer to the documentation accompanying your ROM board for details.
When your UTOPIA ROM is installed, switch on and type '|HELP'. This will list all ROMs. If UTOPIA is listed then all is well and all the commands are ready for use. If not, or if the message 'Unknown command' is displayed, check that you have not chosen a ROM select number already occupied by something else.

USING UTOPIA

Command entry

UTOPIA's facilities are all available from BASIC, MAXAM or PROTEXT as 'external commands'. They can also be used with other well-written commercial ROMs. The commands are entered by typing a vertical bar ‘|’ followed by the command name.
Most of the commands take one or more parameters (such as filename, string to find). These parameters can be entered in one of two ways:

(i) after the command name separated by commas,
(ii) by pressing ENTER (or RETURN) after the command name, in which case UTOPIA will prompt for all required parameters.

Method (ii) allows a command to be used if you've forgotten what parameters it takes. The other advantage is that it allows the entry or string parameters easily from BASIC. Method (i) Can be very quick and convenient, especially if used with MAXAM or PROTEXT which allow free format entry of external commands.

The HELP command can be used to list all UTOPIA's commands on the screen. '|HELP' lists all sideways ROMs, including UTOPIA, together with their 'ROM numbers'. Then '|HELP,n' where n is UTOPIA's ROM number will list all the external commands that are available. HELP also displays the UTOPIA version number, which should be quoted in any correspondence.

If you have ROMs produced by different companies you may find that there to a conflict of command names. Where two or more ROMs each have a command with the same name, the command in the lower numbered ROM will normally take priority. UTOPIA includes a feature to override this if another ROM to intercepting a command meant for UTOPIA. To do this type the command |U. You will than be asked for the name, after which the command will be executed as normal.

As far as possible UTOPIA follows BASIC in the use of the ESC key. So for virtually all commands, pressing ESC once causes the command to pause and a second press aborts the command (another key resumes it). However when commands are accessing the disc the computer does not scan the keyboard so it may be necessary to hold the ESC key down for about half a second.

Filenames

If a filename is entered without an extension, the AMSDOS convention is followed - if the file is not found the suffix .BAS is added. If this is not found the suffix BIN is tried. This applies to the following commands:

ACCESS COPY DELETE DUMP INFO LIST LOAD TYPE VERIFY VTEXT

It does not apply to ERA and REN, these work exactly as the AMSDOS command and are merely intercepted to allow easy parameter entry from BASIC.

The command SAVE creates a file with a .BIN suffix if none is given.

Ambiguous filenames

Many commands allow the use of ambiguous filenames. An ambiguous filename contains one or more wildcard characters. The wildcard characters are:

? ... this will match any single character in a filename

* ... this will match any string of characters in a filename.

Examples of use of wildcards:

.	all files.
*.BAK	all backup files.
?AT	all 3 latter filenames ending in AT (e.g CAT, MAT).

Function keys

UTOPIA is present in the machine from the moment you switch on, and all the commands are instantly accessible. In fact, by the time BASIC displays 'Ready' for the first time UTOPIA has already done several things. These are:

1. It determines whether discs are being used. If so the expansion string produced by pressing CTRL and ENTER (this means the small ENTER key on the 464 and 664) is changed from RUN" to RUN"DISC. This allows the use of an auto-boot facility with disc. Save a BASIC program called 'DISC' which performs any required action (e.g. running a BASIC or machine code program, setting pen and paper colours), then in future you just need to insert the disc and press CTRL-ENTER.
2. Expansion tokens 150 to 159 are associated with the keys f0 to f9 (0 to 9 at the right of the keyboard) when CTRL is pressed. Expansion token 149 is associated with the full stop key at the right of the keyboard when CTRL is pressed.
3. The expansion tokens 149 to 159 have strings assigned to them. The affect of 2 and 3 is to define the following function keys:

CTRL-f0	RUN
CTRL-f1	LIST
CTRL-f2	MODE 2
CTRL-f3	|PRINTOFF
CTRL-f4	|HELP
CTRL-f5	|TOKENS
CTRL-f6	|PRINTON
CTRL-f7	|PROTEXT
CTRL-f8	|MAXAM,2
CTRL-f9	CALL &BB4E:CLS
CTRL-f.	|STATUS

CALL &BB4E calls the firmware routine 'TXT INITIALISE', which performs the following actions: selects stream 0, sets text paper to ink 0, sets text pen to ink 1, sets the text window to the entire screen, sets the character writing mode to opaque, enables VDU, turns off the graphic character write mode, moves the cursor to top left, cancels user defined character definitions. Thus CTRL-f9 can be used to reset many aspects of the text VDU - often useful after a program has gone wrong and left the display in an unusable state.

CTRL-f7 and CTRL-f8 are only of use to owner of the PROTEXT word processor and the MAXAM assembler respectively. Both of these acclaimed programs are available in ROM from Arnor.

THE COMMANDS IN DETAIL

The major part of this manual details each command, in alphabetical order. The following headings are used, though they are not all used in every case:

1. Syntax: The parameters required by the command are listed on one line - but don't forget you can just press ENTER (or RETURN) after the name to obtain self explanatory prompts for all parameters. The following conventions and abbreviations are used in the syntax descriptions:

<>: an item enclosed in angle brackets is a description of the parameter. Other letters etc, should be typed literally as shown.

(): an item enclosed in parentheses is optional.

2. Description: The use and effect of the command explained.
3. Examples: Examples illustrating the use of the command and the output produced by the command.
4. Technical notes: These notes are included for those who are interested. They are concerned with how the command works rather than how to use the command. They may therefore assume certain knowledge about the workings of the Amstrad computer. Don't worry though - they can be safely ignored.
5. Related commands: A list of commands that are either used in association with the command just described, or whose description throws some light on its use.

Access - Set access attributes (disc only)

Syntax: ACCESS <ambiguous filename> (p)

Every file stored on disc has an ‘access attribute’. This simply says whether the file is protected against alteration or deletion. When a file is created its access attribute is 'Read/Write'; that is it is not protected. The file can be protected by using the ACCESS command with parameter ‘P’. This sets the file to 'Read-Only'. In order to save a new file or delete the file, first use ACCESS without the parameter 'P'. Protected files are marked in the catalogue with an asterisk.

Examples:

```
   1. |ACCESS,"myprog",”P”
   Sets file 'myprog' to Read-Only
   2. |ACCESS, "*.BAS", "P"
   Set all BASIC program files to Read-Only.
   3. |ACCESS,"*.*"
   Set all files to Read/Write.
```

Technical note:
|ACCESS <file> P’ is equivalent to CP/M's 'STAT <file> $R/O'.
|Access <file>’ is equivalent to 'STAT <file> $R/W'.

Related commands: CAT, COPY, DELETE

Arrays - List currently defined arrays

Syntax: ARRAYS

A complete list of all defined arrays is produced. The dimensions of each array is given. The arrays are listed in the following order: all real arrays in the order they were defined, all integer arrays in the order they were defined, all string arrays in the order they were defined. The type of each array is indicated by the appropriate type marker: ! for real, % for integer, $ for string.

Example:

```
   |arrays
   VECTOR! (9)
   A$ (13,6)
```

Related commands: FNS, VARS

C - Calculate value of expression

Syntax: C

This command always prompts for an expression. The expression must contain the following: integer constants expressed in decimal, hexidecimal (prefixed by '&'), binary (prefixed by '%'), and the operators +, -, *, /. The result is evaluated modulo 65536, that is the two least significant bytes of the result are displayed. The result is displayed in two ways - in hexadecimal and signed decimal.

Example:
The BASIC expression evaluator is unable to cope with certain hexadecimal calculations because it always treats integers as signed numbers. The C command avoids this problem.

```
|C
Expression: &9a15-&7858
Value is: &21BD = 8637
```

CALL - call machine code routine

Syntax: CALL <address> (<a>) (<bc>) (<de>) (<hl>)

The routine at the specified address is called, using any parameters to set the values of the registers A, BC, DE, HL (in that order). Some or all of the register values may be omitted, in which case those registers are set to zero. On return from the routine the register values are displayed.

Examples:
This command is useful for investigating the effect of machine code routines (especially the firmware routines) without the need to assemble any code. This is not usually possible from BASIC because the BASIC 'CALL' command does not allow values to be returned.

6. |CALL,&BBA5,241
   Returns with HL containing the address of the character matrix for the ASCII code 241.
7. |CALL,&BD2A,68
   Returns with A containing the character or token that the TAB key is translated to. (68 is the key number of the TAB key).
8. |CALL,&BB87,0,0,0,&908
   Checks whether the position (row 8, column 9) is in the current window, and alters HL if it is not.

CAT – Catalogue files

Syntax: CAT (<drive>)

CAT is almost exactly the same as the BASIC command CAT. The main advantage for disc users is that the drive to be catalogued may be specified, allowing the cataloguing of drive B without altering the default drive.

Note: CAT will use the AMSDOS commands A and B as necessary to select the required drive, and afterwards restore the previous setting. Thus if the currently selected drive has no disc in it an error will occur, in this case press C to cancel and note that the default drive will have changed.

Examples:

```
1. |CAT         Catalogue current drive
2. |CAT,"B"     Catalogue drive B
```

Technical note:

The BASIC command CAT calls CAS IN ABANDON and CAS OUT ABANDON, thus abandoning any open tiles. This is not necessary though, so |CAT only calls CAS IN CLOSE for tape, and does not affect either stream for disc.

COPY - copy file

Syntax: COPY (<new filename>) <old filename>

This will copy a file from the current input filing system to the current output filing system. If <new filename> is emitted, the new copy will be given the same name as the old file. If copying from tape both names may be omitted - the first file will then be copied. It copying from tape to disc the name will he truncated if it is too long. Any type of file may be copied.
Note: COPY closes any open files.

Examples:

```
1. Copying from tape to tape or disc using the same name.
|TAPE.IN
|COPY
2. Copying a named file from disc or tape to tape.
|TAPE.OUT
|COPY,”progfile”
3. Copying a file and renaming it
|COPY,”newname”,”oldname”
4. Copying file from drive A to drive B
|COPY,”b:maxam”,”a:maxam”
```

Technical notes:

COPY copies the file byte by byte using CAS OUT CHAR. It is requires 2K of free memory.
If copying onto tape the load address of the file will be lost. This is because the firmware does not permit the setting of the load address field in the file header. This does not APPLY if copying onto disc because AMSDOS allow the load address field to be set.

Related commands: DELETE, LOAD, SAVE

DEDIT - Disc editor

Syntax: DEDIT <drive> (<track>) (<sector>)

DEDIT allows the examination and direct alteration of the contents of a disc. If a sector number is specified, DEDIT will attempt to read that sector. Otherwise it will determine the format of the disc and read the first sector if the required track. If no track number is given, DEDIT assumes track 0. The top line of the screen will show the drive (A or B), the track number and the sector number. Sector numbers depend on the format of the disc but are always determined automatically. For reference though they are as follows:

System or Vendor format:	&41 to &49 (9 sectors per track)
Data only format:	&Cl to &C9 (9 sectors per track)
IBM format:	1 to 9 (8 sectors per track)

Each sector is 512 bytes long and the display shows half of a sector at a time in 80 column mode. Pressing SHIFT with the cursor up and down key, swaps between the two halves of the sector. The left part of the screen shows the hexadecimal representation and the right part shows ASCII. Each can be edited, simply by overtyping the displayed values.

Use CTRL with the cursor keys to move between sectors as shown below.

When you have finished editing a sector you will need to write the sector to the disc. This is not done automatically so if the alterations are displayed on the screen and you move to a new sector, the disc will not be changed. To save the changes press CTRL-COPY. The current sector as it is displayed on the screen is copied to the disc.

Corrupted discs etc.

If a read error occurs the message "Unable to read sector move to new sector or press ESC' is displayed. This usually means that DEDIT was trying to read a non-existent sector - this will happen it you enter a sector number corresponding to a different disc format, or it you are trying to access a protected disc. If neither of these are the case then you Probably have a corrupted disc. It may be that there is just one bad sector though, in which case you can use the commands below to move to another sector or track. Before consigning the disc to the bin, reset the machine completely and try again: also try other discs in the case the drive is faulty.

DEDIT offers the following commands:

TAB	switch between hex and ASCII editing.
SHIFT <cursor-up>	move to top half of sector
SHIFT <cursor-down>	move to bottom half of sector.
SHIFT <cursor-left>	move to start of line
SHIFT <cursor-right>	move to end of line
CTRL <cursor-up>	move back one track
CTRL <cursor-down>	move on one track.
CTRL <cursor-left>	move back one sector
CTRL <cursor-right>	move on one sector.
CTRL-COPY	write sector to disc.
COPY	COPY sector to memory. A memory address will be requested which should be the start of a 512 byte (&200 byte) area of memory.
ESC	finish.

Examples:

1. It a file is accidentally deleted it may (with the appropriate knowledge of the CP/M filing system) be restored using DEDIT.
2. It a disc becomes corrupted it may be possible to restore some or all of a file by copying the sectors into memory, and then using SAVE to save the recovered file onto another disc. For example, if a file consists of 3 sectors and the sectors can all he read by DEDIT, copy the sectors to &1000, &1200, &1400 and then enter the command '|SAVEA,”file",&1000,&600’. If you have MAXAM or PROTEXT you can then edit the recovered file.

Related commands: FORMAT, MEDIT

DELETE - Delete file or files (disc only)

Syntax: DELETE <ambiguous filename>

DELETE is similar to the AMSDOS command ERA, the difference being that instead of just deleting each file matching the ambiguous filename, the filename is displayed and the question 'Delete (y/n)?' is displayed. If Y is pressed the file will be deleted, otherwise it will not. If you realise you have made a mistake before DELETE has finished, press ESC and no changes will be made. Otherwise all the files you marked for deletion will be deleted together after all filenames have been displayed. Files set to Read-Only, may not be deleted.

Related command: ERA

DISCCOPY - Copy disc

Syntax: DISCCOPY <source drive> <destination drive>

DISCCOPY will copy a disc using one or two drives. The question 'Are you sure (y/n)?” will ask you to confirm that the copying is to go ahead. If any key other than Y is pressed the copying will not proceed. Any format of disc may be copied but the two discs must be of the same format.
The use of this command will overwrite user memory and any RSX that has been loaded from disc will be lost.

Examples:

1. Copying with a single drive
```
|DISCCOPY, "A”, "A"
```
2. Copying with a dual drive system
```
|DISCCOPY, "A”,”B”
```

It is recommended that you always copy from drive A to drive B and NEVER the other way round, and also that whilst copying the source disc is write protected. If these precautions are not followed it is almost inevitable that you will at some time copy the backup disc onto the working disc!

Related commands: COPY, DISCTEST, FORMAT

DISCTEST - Test disc for read errors

Syntax: DISCTEST <drive>

DISCTEST simply reads every sector of the disc in the specified drive. If the disc is badly formatted or corrupted a 'Read Fail' will occur at some point. If this occurs try re-formatting the disc. If DISCTEST succeeds the message 'Disc formatted correctly is displayed.
The operation of DISCTEST is carried out automatically by DISCCOPY and FORMAT.

Related command: FORMAT

DUMP - Dump file contents

Syntax: DUMP <filename>

DUMP will read a file from tape or disc and display the contents of the file on the screen. Both hexadecimal and ASCII representations are shown.
Note: DUMP closes the input file if there is one.

Technical note:

When used on disc DUMP treats soft end of file (&1A) as a character in the file and continues to the hard end of file that is the end of a 128 byte block.

Related commands: LIST, TYPE

ERA - Erase file or files (DISC only)

Syntax: ERA <ambiguous filename>

This is the AMSDOS command and so is identical in affect and use to that described in the Amstrad manual. The command is, however, intercepted and will prompt for parameters in the usual way, thus allowing easy deletion of files from BASIC.
ERA will also warn you if you attempt to delete all files, as CP/M (but not AMSDOS) does.

Related commands: DELETE

FIND - Find tokenised string in BASIC program

Syntax: FIND <string>

The FIND command will search for any string of BASIC text in the current BASIC program. In order to use FIND to the best affect it is necessary to understand a little about how BASIC stores programs. It would be very simple to search for a string if programs were stored exactly as you type them in, but they are not. For reasons of speed and compactness BASIC takes the lines of program as you enter them and converts them into a special code. This operation is called 'tokenisation'. So for example if you enter the command 'PRINT' in a program, BASIC does not store the 5 letters of 'PRINT', but instead the single number (or 'token') &BF. Similarly every BASIC keyword has a token. BASIC also stores numbers and variables that occur in programs in a special way.
What this all means is that when you enter a string to be found, it must be tokenised in exactly the same way before beginning the search.
The entire program is searched for the string and the line number on which it occurs is listed.
FIND is case sensitive; that is letters typed on lower case will only match lower case letters in the program, and capitals will only match capitals.

Notes:
FIND cannot be used to search for string constants because these are not tokenised. For this, use the command FINDA.
If the string being searched for is very short (1 or 2 characters) it may sometimes appear to find the string wrongly. In fact it is finding the string in the BASIC text where it is part of a token, such as a variable.

Related commands: FINDA, REPLACE

FINDA - Find ASCII string in BASIC program

Syntax: FINDA <string>

FINDA is the same as FIND in every respect but one: the string is not tokenised before searching. So this can be used to search for items in REM or DATA lines, or for string constants. Wildcards are allowed in the string. A wildcard is a character which will match any character in the BASIC program text. A wildcard is specified in the string by typing a question mark ('?'). Any number of wildcards may be used.
FINDA is case sensitive: that is letters typed in lower case will only match lower case letters in the program, and capitals will only match capitals.

Example:
```
|FINDA, “f??e”
```
Finds all 4 character ASCII strings staring with 'f' and and with ending with ‘e’.

Related commands: FIND, REPLACEA

FNS - List currently defined functions

Syntax: FNS

The names of all functions and the line numbers in which they are defined are listed. The type of the function is also indicated by the type marker after the name: ! for real, % for integer, $ for string.
Note that only defined functions are listed so FNS should be used after running a BASIC program.

Example:
```
|fns
 START! Line 1000
 ENTER$ Line 1250
```

Related commands: ARRAYS, VARS

FORMAT - format a disc

Syntax: FORMAT <drive> (<format type>)

FORMAT does the same as the CP/M utility ‘FORMAT'. The format type is a single letter identifying one of two formats:
V: Vendor Format. This is the default, and is the same as the System Format except that the CP/M system tracks are left blank. This format gives 169K per disc.
D: Data Format. This has no system tracks and gives 178K per disc.
The default if neither V nor D is specified is Vendor Format.

Technical note:
Discs are formatted in the recommended way with 2 to 1 sector interleave.

Related commands: DISCCOPY, DISCTEST

HELP - list sideways ROMs or external commands

Syntax: HELP (<rom number>)

HELP with no parameter will list sideways ROMs. All foreground and extension ROMs will be listed. Those background ROMs that have been initialised are listed (this generally means all unless the ROMOFF command has been used).
The information given is ROM select number, ROM name, version number, ROM type, and for a background ROM the address of it’s upper workspace area.
This Information can than be used to list the commands provided by any background ROM. Enter the selected number of the required ROM as the parameter for HELP and all the commands will be listed.

Technical note:
The ROM name is taken from the name of the initialisation command, that is the first name in the name table. The version number is taken from bytes 1 to 3 of the ROM. Commands such as the AMSDOS commands CTRL-A to CTRL-I are not listed.

Example:
```
|HELP
|HELP,n lists commands for ROM n
|HELPR lists RSX commands
ROM 0. BASIC         1.10 foreground
ROM 1: UTOPIA        1.00 back &A2F4
ROM 2: PROTEXT       1.00 back &A3F8
ROM 5. MAXAM         1.10 back &A5FC
ROM 7: CPM ROM       0.50 back &A700

|help,5
ROM 5: MAXAM         1.10 back &A5FC
 MAXAM        ASSEMBLE
 ASSEM        CAT
 CLEAR        FIND
 HELP         M
 MAXAM        MODE
 MSH          MSL
 RAMON        RAMOFF
 ROMOFF       SPEED
 MCLEAR       MFIND
 MHELP        MROMOFF
```

Related commands: HELPR, ROMOFF

HELPR - list RSX command.

Syntax: HELPR

This lists all external commands provided by RSXs that have been loaded from tape or disc, in the same way that HELP,n lists external commands provided by background ROMs.

Related command: HELP

INFO - display information about file or files

<u>Syntax:</u>

```
        INFO (tape only)
        INFO <ambiguous filename> (disc only)
```

The information displayed is taken from the file header. When used on tape all files are listed, on disc all files matching the ambiguous filename (so, in Particular, INFO *.* will list all files on the disc). Note that ASCII files have no header and zeros will be shown.

The information listed 1isted is as follows:

1. File type. The same characters shown by CAT on tape:
   * is ASCII, & binary, $ unprotected BASIC, % Protected BASIC.
2. Load address. The address in memory to which the file will be loaded by default.
3. Logical length. The length of the file as stored in the file header.
4. Entry address. For machine code programs, the address at which execution is to begin after the file has been loaded.
5. (Disc only) Size. The actual size of the file on the disc. This is listed for all files, whether they have a header or not. This will be different to the logical length because the size includes the size of the header itself. The size is a multiple of 128 bytes, the unit CP/M handles.

All these numbers are shown in hexadecimal.
Note: INFO closes the input file if there is one.

<u>Example:</u>

```
|info,"*.*"
                      LOAD LOGL ENTRY SIZE
A:DISC    .BAS    $ 0170 006A 0000 0100
A:DISC    .BIN    & 9000 2182 9156 2280
A.REPORT  .       * 0000 0000 0000 5E80
```

<u>Related command:</u> CAT

LIST - list ASCII file

Syntax: LIST

LIST will read a file from tape or disc and list the contents on the screen, numbering the lines. Tab characters are acted upon, taking tab positions at every eighth column.
Note: LIST closest the input file if there is one.

Technical note:
If the file in of type ASCII, LIST stops at a soft end of file (&1A).

Related commands: DUMP, TYPE

LOAD - load a file into memory

Syntax: LOAD <filename> (<load address>)

This LOAD command has much greater use than the equivalent BASIC command, which only operates with binary files, and only allows files to be loaded at an address greater than HIMEM. Here there is no restriction on either file type or address, and the command is of particular use in MAXAM for loading machine code or other files.
The file is loaded at the specified address, or, if the address is omitted, to the load address from the file header. In the case of ASCII files there is no header so an error message will be given if no load address is specified.
The load address to not checked in any way, so ill-chosen addresses may cause a system crash.
Note: LOAD closes the input file if there is one.

Examples:
1. |LOAD,”binary”
Load file called “binary” at the address in the header.
2. |LOAD,”ascii”,&3000
Load file called “ascii” at &3000.

Related commands: SAVE, VERIFY

MDUMP – memory dump

Syntax: MDUMP <start address> (<end address>)

MDUMP displays the contents of memory in both hexadecimal and ASCII. If no end address is specified it will continue to the end of memory, &FFFF. Usually MEDIT is more useful for examining memory contents, but MDUMP is needed in conjunction with PRINTON to send a memory dump to the printer.

Example:
```
|MDUMP,&170
```
Lists the memory area containing the BASIC Program.

Related commands: MEDIT

MEDIT - memory editor

Syntax: MEDIT <address>

Examination and direct alteration of memory contents are possible with MEDIT. When the command is entered, the whole screen is used to display an area of memory with the selected address in the centre. Hexadecimal and ASCII representations are shown, and the contents may be altered simply by overtyping what is shown on the screen. Pressing TAB will switch between hex and ASCII editing modes.

MEDIT offers the following commands:

TAB	switch between hex/ASCII editing modes.
SHIFT <cursor-up>	move to top of screen.
SHIFT <cursor-down>	move to bottom of screen.
SHIFT <cursor-left>	move to start of line.
SHIFT <cursor-right>	move to end of line.
CTRL <cursor-up>	move back one screenful.
CTRL <cursor-down>	move on one screenful.
CTRL <cursor-left>	move to top left.
CTRL <cursor-right>	move to bottom right.
ESC	finish

Related commands: DEDIT, MDUMP

MOVE – move BASIC lines

Syntax: MOVE <first line> <last line> <destination line>

It is often useful to be able to move a section of a BASIC program when reorganising or tidying up your program. MOVE takes three parameters, all of which are BASIC line numbers. The first two line numbers define the block which is to be moved. The line specified need not exist: if the first number does not exist the block is taken to start at the first line after that specified: if the second number does not exist the block is taken to end at the last line before that specified.

The destination line determines where the block is to be moved to. The block is inserted after the line. If the line does not exist the block is inserted at the point where the line would have been.

After the block has been moved the line numbering will be wrong because the line numbers are not altered. So after using MOVE it is essential to renumber the program.

Example:

```
MOVE 200,340,1200
RENUM
```

PRINTOFF - turn off printer echo

Syntax: PRINTOFF

After issuing the PRINTOFF command screen output is no longer copied to the printer.

Related command: PRINTON

PRINTON - turn on printer echo

Syntax: PRINTON

After issuing the PRINTON command all subsequent screen output is copied to the printer. In particular it can be used to print the output from commands such as TYPE, LIST, DUMP, MDUMP, HELP, TOKENS, FIND, VERIFY, VARS, FNS and ARRAYS. Also, the output from BASIC programs can very easily be sent to the printer, which can be very useful for debugging.

Technical note:
PRINTON works by intercepting TXT WRITE CHAR and TXT OUT ACTION. Control codes sent to TXT OUT ACTION are correctly intercepted and not sent to the printer.

Related commands: PRINTOFF, SPOOL

REN - rename file

Syntax: REN <new filename> <old filename>

This is the AMSDOS command and so is identical in effect and use to that described in the Amstrad manual. The command is, however, intercepted and will prompt for parameters in the usual way, thus allowing easy renaming of files from BASIC.

REPLACE - find and replace string in BASIC program

Syntax: REPLACE <old string> <new string>

REPLACE works exactly as FIND to find the old string, and then replaces it by the new string. Both strings are tokenised (see FIND). The new string can be shorter, the same length, or longer than the old string. BASIC variables remain defined, even if the strings are of different lengths.
Care must be taken when replacing single character strings (see note under FIND).
The old string may contain wildcards (see FIND).

Example:
```
|REPLACE,”f%”,”flag%”
Renable variable f% to flag%
```

Related commands: FIND, REPLACEA

REPLACEA - replace ASCII string in BASIC program

Syntax: REPLACEA <old string> <now string>

REPLACEA is the ASCII equivalent of REPLACE, replacing one string by another.

Related commands: FINDA, REPLACE

ROMOFF - turn off all or selected ROM.

Syntax: ROMOFF (<list of rom numbers>)

ROMOFF causes the machine to be reset, initialising only selected background ROMs. If no parameters are given no background ROMs will be initialised, otherwise all ROMs will be initialised except those specified. Use the HELP command to determine the ROM numbers.

Warning: ROMOFF will reset the machine completely and destroy memory contents.

Example:

```
    |ROMOFF,5,7
    Turn off ROM 5 and the disc ROM.
```

Related Command: HELP

SAVE - save block of memory as binary file

Syntax: SAVE <filename> <start address> <length> (<entry addr>) (<load addr>)

Any block of memory can be saved, and is specified as in the BASIC command SAVE by start address and length. Again as in the BASIC command the fourth parameter is optional and specifies the entry address for machine code programs. The fifth (optional) parameter is an addition to BASIC end allows the setting of a load address different to the address from which the file was saved. This latter option is only available to disc users because the cassette firmware does not allow the load address to be set.

Note: SAVE closes the output file if there is one.

Related commands: LOAD, SAVEA, VERIFY

SAVEA – save block of memory as ASCII file

Syntax: SAVEA <filename> <start address> <length>

SAVEA is the same as SAVE except that the block of memory is saved as an ASCII file. Only three parameters apply since the ASCII file has no header in which to store the other information. This command is provided for use with where only ASCII files may be used, such as CP/M applications.

Note: SAVEA closes the output file, if there is one.

Related command: SAVE

SPOOL - turn on spooling to file

Syntax: SPOOL <filename>

When the SPOOL command is issued the specified file is opened. All subsequent screen output is then copied to that file until a SPOOLOFF command is issued.
Warning: the use of some commands cause the output file to be closed and spooling to be turned off. These include SAVE, SAVE and the BASIC command CAT. If this occurs the command SPOOLOFF must still be issued before SPOOL can be used again.
Note: SPOOL closes the output file, if there is one.

Related commands: PRINTON, SPOOLOPP

SPOOLOFF - turn off spooling to file

Syntax: SPOOLOFF

The spool output file is closed and screen output is no longer copied to the file. Note that this command is essential because the last block of the file is not written until this command is issued.

Related command: SPOOL

STATUS - display status information

Syntax: STATUS

The status information listed is shown by the example below. The memory between 'Start of program' and 'End of Program' is occupied by the current BASIC program, followed by any PROTEXT or MAXAM text. The area between 'End of program' and 'First free location' is occupied by BASIC variables. The area between 'Last free location' and HIMEM is occupied by BASIC strings.

Example:
```
|status
Start of program = &0170
End of program          = &223C
First free location     = &223D
Last free location      = &A36F
HIMEM                   = &A36F
WIDTH setting=          13
SYMBOL AFTER            240
Program size     = 8397 bytes
Free memory= 31027 bytes
```

TOKENS – display expansion strings

<u>Syntax:</u> TOKENS

The strings associated with all the expansion tokens are listed. The tokens 149 to 159 are set on reset. To set further definitions (using the BASIC command KEY) it may be necessary to clear space in the expansion string buffer. To do this set some of the tokens to the null string. e.g. KEY 158,””

<u>Example:</u>

```
   |tokens
   128 0            129 1
   130 2            131 3
   132 4            133 5
   134 6            135 7
   136 8            137 9
   138 .            139
   140 RUN”DISC     141
   142              143
   144              145
   146              147
   148              149 |STATUS
   150 RUN          151 LIST
   152 MODE 2       153 |PRINTOFF
   154 |HELP        155 |TOKENS
   156 |PRINTON     157 |PROTEXT
   158 |MAXAM,2     159 CALL &BB4E:CLS
```

TYPE - type ASCII file

<u>Syntax:</u> TYPE <filename>

TYPE will read a file from tape or disc and list the contents on the screen. Tab characters are acted upon, taking tab positions at every eighth column.
Note: TYPE closes the input file if there is one.

<u>Technical note:</u>
If the file is of type ASCII, TYPE stops at a soft end of file (&1A).

<u>Related commands:</u> DUMP, LIST

U - execute UTOPIA command

<u>Syntax:</u> U <command name> (<command parameters>)

This command is provided for use when another ROM is intercepting a command meant for the UTOPIA ROM. It simply causes the UTOPIA command to be executed as if the other ROM were not present.

<u>Example:</u>

```
|U,“ROMOFF”, 7
```

<u>Related command:</u> XROM

VARS - display currently defined variables

<u>Syntax:</u> VARS

A complete list of all currently defined variables (excluding arrays) is produced. The variables are listed in the order: integer variables, real variables, string variables. Integer and string variables are listed with their current value, Integers being shown in decimal and hexadecimal. The type of each variable is indicated by the appropriate type marker: % (integer), ! (real), or $ (string).

<u>Example:</u>

```
|vars
COUNT%     15     &000F
SCORE%     6003   &3E83
TOTAL!
REPLY$     “yes”
```

<u>Related commands:</u> ARRAYS, FNS

VERIFY - verify file content.

<u>Syntax:</u> VERIFY <filename> (<address>) (<length>)

The function of VERIFY is to compare the contents of a part of memory with the contents of a file, and to list all differences between the two. It can be used in three ways:

(i) to verify the current BASIC program. If just a filename is specified that file will be compared with the current BASIC program.

(ii) to verify a complete file. Enter a filename and a start address. The file will be compared with the contents of memory starting at that address and continuing to the end of the file.

(iii) to verity a block of memory. Enter a filename, start address and length of memory in bytes. The complete block of memory will be compared with the file. If the end of the file is reached before the end of the block of memory a message to that affect is displayed. This option allows the possibility of verifying just part of a file.

If verification is successful the message “Verification successful” is displayed. It not, the memory addresses where memory and file differ are listed. MEDIT can he used to examine those memory locations.

<u>Examples:</u>

```
1. |VERIFY,”prog”
compare BASIC program with file “prog”.
2. |VERIFY,”mem”,&3000,560
Compare 560 bytes of memory starting at &3000 with the file “mem”.
```

<u>Related commands:</u> LOAD, SAVE, SAVEA, VTEXT

VERIFY - verify text

Syntax: VTEXT <filename>

VTEXT is a special verify command for users of PROTEXT or MAXAM. The current text in memory is compared with the file. VTEXT does not list all differences but stops at the first. This is because it is likely that any difference was caused by text being inserted or deleted, and in that case most of the subsequent bytes would give an error.

Example:

```
    |VTEXT,”letter”
```

Related command: VERIFY

XROM - execute command in specified ROM

Syntax: XROM <rom number> <command> (<parameters>)

XROM is provided for use when more than one background ROM has a command of the same name. When this occurs the command in the lower numbered ROM is normally executed. XROM allows the command in the higher numbered ROM to be called. Use HELP to determine the ROM number you require.
XROM may be abbreviated to X.

Example:

```
    |XROM, 5,“CLEAR”
```

Related command: U

APPENDIX

Memory map – RAM

Address	
0	firmware workspace
40	Background ROMs lower workspace area
	(not used by Arnor ROMs)
(110)	BASIC input buffer
(170)	BASIC program area
	BASIC variable storage
	PROTEXT/MAXAM text area
	** free memory **
HIMEM-1	** free memory if himem altered **
(A270)	user defined characters
(A2F0)	Background ROMs upper workspace
	(including UTOPIA workspace – 256 bytes)
AC00	BASIC workspace
B100	firmware workspace
C000	screen memory

Addresses given in brackets are variable - the numbers shown are for a machine with AMSDOS, UTOPIA, MAXAM and PROTEXT ROMs all initialised.

UTOPIA COMMAND SUMMARY

ACCESS <afn> (P)	Set access attributes.
ARRAYS	List currently defined arrays
C	Calculate value of expression.
CALL <addr> (<a>) (<bc>) (<de>) (<hl>)	Call code routine.
CAT (<drv>)	Catalogue files.
COPY (<fn>) <fn>	COPY file.
DEDIT <drv> <track>	Disc editor.
DELETE <afn>	Selective file deletion.
DISCCOPY <drv> <drv>	COPY disc.
DISCTEST <drv>	Test disc for read errors.
RUMP <fn>	Dump file contents.
ERA <afn>	Delete file or files.
FIND <str>	Find tokenised string in BASIC program.
FINDA <str>	Find ASCII string In BASIC
FNS	List currently defined functions.
FORMAT <drv> (<form>)	Format disc.
HELP (<rom>)	List sideways ROMs or external commands.
HELPR	List RSX commands.
INFO <afn>	Display information about file or files.
LIST <fn>	List ASCII file.
LOAD <fn> (<addr>)	Load file.
MDUMP <addr>(<addr>)	Memory dump.
MEDIT <addr>	Memory editor.
MOVE <line> <line> <line>	Move BASIC lines.
PRINTOFF	Turn off printer echo.
PRINTON	Turn on printer echo.
REN <fn> <fn>	Rename file.
REPLACE <str> <str>	Find and replace tokenised string in BASIC program.
REPLACEA <str> <str>	Find and replace ASCII string in BASIC Program.
ROMOFF (<roms>)	Turn off all or selected ROMs.
SAVE <fn> <addr> <len> (<entry>) (<reload>)	Save block of memory as binary file.
SAVEA <fn> <addr> <len>	Save block of memory as ASCII file.
SPOOL <fn>	Turn on spooling to file.
SPOOLOFF	Turn off spooling to file.
STATUS	Display status information.
TOKENS	Display expansion strings.
TYPE <fn>	Type ASCII file.
U <command>	Execute UTOPIA command.
VARS	List currently defined variables.
VERIFY <fn> (<addr>) (<len>)	Verify file contents.
VTEXT <fn>	Verify text.
X <rom> <command>	Abbreviation for XROM.
XROM <rom> <command>	Execute command in specified ROM.

Abbreviations:

```
 <fn>  ... filename
 <afn> ... ambiguous filename (i.e. allows wildcards * and ? )
 <str> ... sring
 <addr>... memory address
 <drv> ... drive (A or B)
```