MCX BASIC

For the TRS-80 Micro Color Computer MC-10

Darren Atkinson
January 25, 2011

OVERVIEW

MCX Basic is an extension of the MC-10's MicroColor Basic language. It is provided in a 16K EPROM which must be installed in the MCX128 expansion module. This document describes only those changes and additions provided by MCX Basic. It is assumed that the reader is already familiar with MicroColor Basic.

The two primary goals for developing MCX Basic were to provide an easier way to get programs and data in and out of the computer and to give the MC-10 an additional level of compatibility with its big brother; the TRS-80 Color Computer (CoCo).

The first goal has been achieved by the addition of LOAD and SAVE commands that can interact with a file server via the built-in serial port. This greatly improves the speed, ease and reliability of file storage.

The second goal required several additions to the language, bringing the MC-10's dialect closer to that of the CoCo's Extended Color Basic. All statements and functions found in ECB have been implemented with the exception of the following:

```
AUDIO           DEFFN       DLOAD       GET
JOYSTK          MOTOR       PCLEAR      PUT
PRINT USING
```

Several features not found in ECB have also been added, including:

```
AUTO            CHAIN       ERRL        ERRN
MERGE           MAXVAL      MINVAL      ON BREAK
ON ERROR        RUNM        STRMEM      SWAP
WAIT
```

INSTALLATION and STARTUP

Install the MCX Basic EPROM in the ROM socket (U4) of the MCX128 expansion board. Connect the MCX128 to the back of the MC-10 and turn on the computer. At startup, a Boot Menu is presented which allows you to choose the variation of Basic you wish to use. In addition to stock MicroColor Basic, two options for MCX Basic are available (see Standard vs. Large below).

After pressing a key to select the desired option, the screen will go dark momentarily while the system performs a memory test and copies the selected ROM contents into RAM.

NOTE:
To return to the Boot Menu without switching the computer on and off, hold down the BREAK key while pressing and releasing the Reset button.

STANDARD vs. LARGE

At startup, the Boot Menu gives you the choice of Standard or Large configurations for initializing MCX Basic.

* The Standard configuration provides roughly 28K of RAM for programs and variables along with five pages for graphics.

* The Large configuration provides roughly 46K of RAM for programs and variables along with eight pages for graphics.

The Large configuration should not be used when you intend to load and run machine language programs that are unaware of the MCX128 hardware. This is because the Large configuration works by placing programs and variables in a bank of RAM separate from that used by the video display. In this environment, programs that need to access the display must know how to select the proper RAM bank.

Basic programs that use PEEK and POKE to access video RAM must also be modified in order to work correctly under the Large configuration. PEEKing or POKEing addresses between 16384 and 48895 will normally access the bank containing the Basic program workspace. To instead access the bank containing the video display and graphics pages, you must suffix the PEEK or POKE keywords with '@'. For example:

```
10 REM BLANK THE SCREEN
20 FOR A=16384 TO 16895
30 POKE@A,128
40 NEXT
```

When `PEEK@` or `POKE@` is used, the location that is accessed will always be within page 1 of the address space as defined by the MCX128 hardware ($4000 - $BFFF). The computer calculates the actual address using the formula:

address = $4000 + (argument AND $3FFF)

NOTE:
The graphics pages (but not the screen) are located in different areas of memory depending on which configuration is selected.

DETECTING MCX BASIC

A program that wants to know whether or not MCX Basic is present can examine memory locations $FFDA..DB. MCX Basic places the ASCII characters 'EX' in these locations.

BASIC'S SYSTEM VARIABLES

MicroColor Basic positioned its system variables in two separate regions of memory. The direct page locations 128 - 255 ($0080 - $00FF) were used for those variables which are frequently accessed or would benefit from the use of Direct Page addressing. The remaining system variables were positioned at addresses 16896 to 17221 ($4200 - $4345) which immediately follows the 512 bytes of RAM used for the Text / Semigraphics screen.

MCX Basic keeps all of the direct page variables in exactly the same locations, but moves the others so that they no longer conflict with the area needed to display a graphics screen. The new address for one of these variables is calculated by subtracting 16640 ($4100) from the original address. They now reside immediately following the direct page, starting at 256 ($0100).

GRAPHICS MODES

Three of the 6847's graphics modes are now available from Basic. Unfortunately, Tandy's decision to limit the internal RAM to 4K means that modes RG6 and CG6 (aka PMODEs 3 and 4) can't be displayed without wrap-around. For this reason, MCX Basic supports only RG2, CG3 and RG3 (aka PMODEs 0, 1 and 2).

Although the maximum horizontal resolution in these modes is 128 pixels, MCX Basic has adopted the Extended Color Basic convention that all modes are addressed using horizontal coordinate values from 0 to 255 and vertical coordinate values from 0 to 191. These values are then scaled to fit the actual 128 x 96 or 128 x 192 screen size for the current mode.

The system allocates either 5 or 8 *pages* of memory for graphics use depending on which configuration was selected from the Boot Menu. The graphics pages are always available in MCX Basic. There is no PCLEAR statement to control the allocation of graphics pages. Each page is 3K in size (as opposed to the 1.5K pages used in Extended Color Basic) and can accommodate a full graphics screen for any of the supported modes.

NOTE:

The graphics pages are located in different areas of memory depending on which configuration was selected from the Boot Menu.

KEYBOARD SHORTCUTS

The keyboard shortcuts which produce the `CSAVE` (Ctrl-3) and `CLOAD` (Ctrl-4) keywords under MicroColor Basic have been changed to produce `SAVE` and `LOAD` (without the C).

MCX Basic will also interpret the combination of `SHIFT-BREAK` to be the equivalent of pressing the `CLEAR` key on a CoCo (character code 12).

MAXIMUM LINE LENGTH

The maximum line length allowed during program entry or in response to an INPUT statement has been increased to 249 characters. This matches the line length permitted on the CoCo.

HEX and OCTAL CONSTANTS

MCX Basic adds the ability to use hexadecimal constants prefixed with &H and octal constants prefixed with &O.

```
POKE &HBFFF, &H4C
K = &O777
```

COMPUTED LINE NUMBERS

MCX Basic allows you to use variables and/or expressions for line number references. To utilize this feature you must place the expression inside of parenthesis. For example:

```
GOTO (N)
GOSUB (100+(J*50))
```

If you use computed line numbers in your program then you should be wary of using the RENUM command. RENUM does not modify GOTO and GOSUB statements which use the computed line number capability.

RESTORE TO LINE

In MCX Basic, the `RESTORE` statement will accept an optional line number argument. This causes the system to start searching for `DATA` statements at the specified line number when the next `READ` statement is executed.

```
RESTORE 500
RESTORE (1000+K*100)
```

INKEY$ CORRECTION

The `INKEY$` function in MicroColor Basic would sometimes return the character for the BREAK key (03) rather than allow the BREAK key to stop execution of the program. This problem has been corrected in MCX Basic.

VARPTR CORRECTION

The `VARPTR` function in MicroColor Basic was returning a signed integer result. This caused a negative value to be returned for variables located at addresses above $7FFF. MCX Basic fixes this problem by always interpreting the result as an unsigned integer.

COMMANDS

AUTO *start, interval*

Begins program entry with automatic line numbering. The first line number to use is indicated by *start*. Each subsequent line number is produced by adding *interval*. Both arguments are optional and default to 10 if omitted. To exit the auto line numbering mode press the BREAK key (or press ENTER without typing any additional characters on the line).

CSAVEM "*filename*", *start*, *end*, *transfer*

Saves a block of memory to cassette. The *start* and *end* arguments specify the inclusive address range of the memory block to save. The *transfer* argument is optional and specifies the default address to use for the EXEC command when the file is later loaded using CLOADM. If *transfer* is omitted then *start* will be used instead.

DEL *start-end*

Deletes one or more program lines. The program line(s) to be deleted are specified as a range of line numbers in the same format accepted by LIST.

Examples:

`DEL 50`	delete line 50
`DEL -30`	delete all lines before and including line 30
`DEL 100-200`	delete all lines from 100 to 200 inclusive
`DEL 500-`	delete line 500 and all following lines

DIR "*directory name*"

List the names of files contained in a directory on the server. If the *directory name* is omitted then files from the default directory are listed (see the SETDIR command on page 7).

DIRLIST

List the names of directories available from the server.

EDIT *line*

Enters Edit mode for the specified *line* number. You may also type just the letter '**E**' followed by a line number to invoke the editor. The following key commands are available in Edit mode:

A	Abort any changes made to the line and start over.
*n*B	Move cursor back *n* characters.
*n*C	Change *n* number of characters. (type the new characters after C)
*n*D	Delete *n* number of characters.
I	Begin *Insert* mode.
H	Delete remainder of the line and begin *Insert* mode. (HACK)
*n*K*c*	Delete characters up to the *n*th occurrence of character *c*. (KILL)
L	List the line (including changes made so far) and continue editing.
*n*N	Move cursor ahead *n* characters. The spacebar may also be used for this.
*n*S*c*	Move ahead to the *n*th occurrence of character *c*. (SEARCH)
X	Move to the end of the line and begin *Insert* mode. (EXTEND)
Cntrl-Z	Exit from *Insert* mode.
BREAK	Cancel the edit operation without saving changes.

NOTE: *The Editor is automatically invoked when a Syntax Error occurs in a program line.*

LOAD "*filename*"

Load a new BASIC program into memory from a file on the server. Refer to the server documentation for specific details on it's features and capabilities.

LOADM "*filename*", *offset*

Load a machine language file from the server. The *offset* argument is optional and, if provided, will be added to the load and execution addresses specified within in the file. Refer to the server documentation for more details on it's features and capabilities.

MERGE "*filename*"

Load a BASIC program from a file on the server and merge it with the program currently in memory. Program lines from the file will replace lines in memory that have the same line number.

NOTE: *MERGE can be quite slow when working with large programs.*

RUN "*filename*", *line*

Loads a new BASIC program into memory from a file on the server and begins execution. The optional *line* argument may be provided to start execution from somewhere other than the beginning of the program.

RUNM "*filename*", *offset*

Loads a machine language file from the server and begins execution. The optional *offset* argument will be added to the load and execution addresses specified within the file.

SETDIR "*directory name*"

Set the default server directory to the one identified by *directory name*. Refer to the server documentation for more details on it's features and capabilities.

RENUM *new*, *start*, *interval*

Renumbers program lines and updates all literal line number references in statements such as GOTO, GOSUB, etc.

Renumbering starts with the first program line whose current line number is greater than or equal to *start*. If *start* is omitted then all of the program's lines will be renumbered.

The first new line number to be assigned is specified by *new*. If *new* is omitted then 10 will be used. Subsequent line numbers are assigned by adding the value of *interval*. If *interval* is omitted then a default interval of 10 is used.

WARNING!: *The RENUM command cannot update computed line number references.*

Examples:

`RENUM`	Renumber all lines. Use 10 for the first new number and interval.
`RENUM 1,,1`	Renumber all lines. Use 1 for the first new number and interval.
`RENUM 500,100,5`	Renumbers lines 100 and beyond to 500, 505, 510...

SAVE "*filename*"

Save the BASIC program currently in memory to a file on the server. Refer to the server documentation for specific details on it's features and capabilities.

SAVEM "*filename*", *start*, *end*, *transfer*

Save a block of memory to a file on the server. The *start* and *end* arguments specify the inclusive address range of the memory block to save. The *transfer* argument is optional and specifies the default execution address to use when the file is later loaded using RUNM or LOADM / EXEC. If *transfer* is omitted then *start* will be used instead.

TROFF

Turns off program Tracing.

TRON

Turns on program Tracing. Running a BASIC program while tracing is on will cause each line number to be printed to the screen as it is executed.

STATEMENTS

’

The apostrophe character can be used as shorthand for REM.

CHAIN "*filename*", *line*

Load and execute a BASIC program from a file on the server without erasing any variables that are currently in memory. The optional *line* argument may be provided to start execution from somewhere other than the beginning of the program.

NOTE: *CHAIN can produce an ?OS ERROR if the current program has created many string variables using literal assignments.*

CIRCLE (*cX*,*cY*), *radius*, *color*, *scaleY*, *start*, *end*

Draws a circle, ellipse or arc on the current graphics page. The (*cX*, *cY*) and *radius* arguments are required to specify the center point and radius.

The optional *color* argument, if provided, must be a color value (0 to 8). If *color* is omitted then the current foreground color is used instead (see COLOR on page 9).

The *scaleY* argument is optional and can be used to draw an ellipse instead of a circle. The argument’s value is a scaling factor that is applied to the radius for the height only. A value of 0.5 will draw an ellipse whose height is half its width. A value of 2 will draw an ellipse whose height is twice its width.

The *start* and *end* arguments are optional and specify the angles for the starting and ending points of an arc. The value for these arguments may range from 0 to 1, where 0 is at 3 o’clock, 0.25 is at 6 o’clock, 0.5 is at 9 o’clock and 0.75 is at 12 o’clock. The arc is always drawn clockwise from *start* to *end*.

CLEAR ERROR

This statement can be used to reset the values returned by the ERRN and ERRL functions to -1 and 65535 respectively. These are the same values which those functions return when no error has occurred since the program was RUN.

CLOSE *fileNumber*

Closes the specific data file indicated by *fileNumber*. If *fileNumber* is omitted then all open data files are closed.

NOTE: *Entering a RUN command or executing an END or FILES statement will also close all data files.*

COLOR *foreground, background*

The COLOR statement sets the current *foreground* and/or *background* colors to use in the current graphics mode. Either value may be omitted if only one of the colors is to be changed.

NOTE: *Changing the graphics mode with PMODE will reset the foreground and background colors to the system's default settings for that mode.*

DEFUSR = *address*

Defines the entry address for the machine language USR function.

DRAW *string*

Draws one or more lines on the current graphics page as instructed by the contents of the *string* argument. Commands within the string may include:

PEN MOTION:

M*x,y*	Move to absolute position *x,y* or move relative distance +/-*x* and +/-*y*.
U*n*	Move Up *n* positions.
D*n*	Move Down *n* positions.
L*n*	Move Left *n* positions.
R*n*	Move Right *n* positions.
E*n*	Move North East *n* positions.
F*n*	Move South East *n* positions.
G*n*	Move South West *n* positions.
H*n*	Move North West *n* positions.

SETTINGS:

A*n*	Set Angle of rotation to *n**90 degrees (clockwise).
C*n*	Set foreground Color to *n* (0-8).
S*n*	Set the Scale factor to *n*/4 (1-62).

OPTIONS:

B	Use as a motion command prefix to move without drawing.
N	Use as a motion command prefix to draw without updating the current position.
X*s$*;	Execute commands in string variable *s$*

ELSE

MCX Basic adds the ability to include an ELSE clause in an IF statement. A line number immediately following ELSE is an implied GOTO (just like THEN).

Examples:

```
IF I >= 25 THEN PRINT I ELSE PRINT I*2
IF SGN(X)= -1 THEN 100 ELSE 200
```

NOTE: *Computed line number expressions cannot be used as implied GOTOs following THEN or ELSE.*

ERROR *number*

Simulate an error of the type specified by the error *number*. See ÒERROR CODESÓ on page 19 for a list of error numbers.

FILES *count*

Changes the number of File Control Blocks allocated to the quantity specified by *count*. The system allocates two FCBs when the computer is turned on. You can allocate a minimum of 0 and a maximum of 15 FCBs. Each FCB that is allocated reduces the amount of RAM available for your programs and data by 260 bytes.

NOTE:
Executing a FILES statement will close any files that are currently open.

INPUT #*fileNumber, var1, var2, ...*

Reads one or more items from the specified data file into the variables listed. The file must have previously been opened for Input using the OPEN statement (see page 12). The EOF function (see page 16) can be used to determine whether or not the end-of-file has been reached.

LINE (*x1,y1*)-(*x2,y2*), *color*, BF

Draw a line or box on the current graphics page.

The optional (*x1*, *y1*) point specifies either the starting point of the line or one corner of the box. If (*x1*, *y1*) is omitted then the end point of the previously drawn line (or 128,96) is used.

The -(*x2*, *y2*) point is required and specifies either the end point of the line or the opposite corner of the box.

The color argument is required and may be a color value (0-8), or one of the keywords PSET or PRESET. Using PSET or PRESET requests the current foreground or background colors respectively.

To draw a box (rectangle) instead of a line, include the **B** or **BF** option following the *color* argument. The **B** option will draw the outline of a box as defined by the two points. The **BF** option will draw the box and fill its interior.

The following statements draw the three sides of a triangle:

```
LINE (50,140)-(128,20),PSET
LINE -(206,140),PSET
LINE -(50,140),PSET
```

LINE INPUT "*prompt*"; *stringVar*
LINE INPUT #*fileNumber*, *stringVar*

The LINE INPUT statement is similar to INPUT, but has the following differences:

* A question mark (?) is not displayed when requesting input from the keyboard.
* The input data can only be assigned to one string variable.
* Commas, colons, quotation marks and leading spaces are all considered part of the string.

The *prompt* and *fileNumber* are both optional.

```
LINE INPUT A$.
LINE INPUT "READY> ";C$
LINE INPUT #1,FL$
```

LOAD* *array*, "*filename*"

Load data for the designated numeric *array* from a file on the server. An ?OM ERROR will occur if the file contains more data than can fit in the array.

LOAD SCREEN "*filename*", *page*

Load a file containing a graphics screen image from the server into the specified *page* of graphics RAM. A *page* argument of 0 will load the image into whichever page is currently being displayed. Page numbers from 0 to 5 may used when running MCX Basic in the Standard configuration. The Large configuration supports page numbers from 0 to 8.

MID$(*oldString*, *position*, *length*) = *newString*

The MID$ assignment statement allows you to replace a portion of one string with the contents of another. *OldString* is the name of the string variable to be modified. *Position* specifies the position of the first character in *oldString* to be replaced.

The *length* argument is optional and specifies the number of characters to replace. If *length* is omitted then the computer uses either the length of *newString* or the number of positions remaining in *oldString*, whichever is smaller.

NOTES:

If *length* is larger than the length of *newString* then all of *newString* is used.

The length of *oldString* never changes. The number of replacement characters is clipped to the number of positions remaining in *oldString*.

OPEN *mode, fileNumber, fileName*

Opens a data file using the specified *mode* and assigns the *fileNumber* used to access the file. The *fileNumber* must be within the range of 1 and the number File Control Blocks that have been allocated. The system allocates two FCBs when the computer is turned on. You can allocate up to 15 FCBs using the FILES statement (see page 10).

The *mode* argument is a string value in which the first character must be one of:

I	Input
O	Output (erases current contents)
A	Append

Once a file has been opened, you can read from or write to the file using the INPUT# or PRINT# statements respectively. With a file that has been opened for Input, you can use the EOF function to test for an End-Of-File condition.

Example:

```
OPEN "I", 1, "MYDATA"
```
Opens file "MYDATA" for input as #1.

ON BREAK...

The ON BREAK statement permits trapping of the BREAK key so that your program can respond in some way other than stopping. There are three variations of ON BREAK:

`ON BREAK CONT`	Continue without stopping (disable BREAK).
`ON BREAK GOTO` *line*	Go to the specified line whenever BREAK is pressed.
`ON BREAK STOP`	Restore normal operation (stop when BREAK is pressed).

ON ERROR...

The ON ERROR statement permits the trapping of errors so that your program can respond in some way other than stopping. There are three variations of ON ERROR:

`ON ERROR CONT`	Continue with the next statement instead of stopping.
`ON ERROR GOTO` *line*	Go to the specified line whenever an error occurs.
`ON ERROR STOP`	Restore normal operation (stop and report the error).

The `ERROR` statement (see page 10) can be used to simulate an error condition. The `ERRN` and `ERRL` functions (see page 16) can be used to determine the type of error and the line number in which it occurred.

PAINT (*x, y*), *fill, border*

Fills a region in the current graphics page with "paint" of the specified *fill* color. Painting starts at the point indicated by (*x*,*y*) and will flood the surrounding area until it encounters pixels of the *border* color. If *fill* is omitted then the current foreground color is used. If *border* is omitted then painting will only stop at pixels which are already the fill color.

WARNING:

PAINT can require a substantial amount of stack space for complex images. When using MCX Basic in the Large configuration, the Paint stack is permitted to invade graphics pages numbered higher than the current page, possibly destroying any images in those pages.

PCLS *color*

Clears the current graphics page by setting all pixels to the specified *color*. If *color* is omitted then the current background color is used.

PCOPY *page* TO *page*

Copies the contents of one graphics page to another. A page number of 0 refers to the current display screen. Page numbers from 0 to 5 may used when running MCX Basic in the Standard configuration. The Large configuration supports page numbers from 0 to 8.

PLAY *string*

Plays musical notes through the TV speaker as instructed by the contents of the *string* argument. Commands within the string may include:

NOTES AND PAUSES:

A..G	Note letters A through G with an optional # (sharp) or - (flat) suffix.
1..12	Note numbers 1 through 12 (use a prefix of 'N' or a suffix of ';').
P*n*	A Pause of 1/*n* note length.

SETTINGS:

O*n*	Select Octave *n* (1-5)
L*n.*	Set note Length to 1/*n* of normal length plus 1/8 times number of dots (L4**.** = 1/4 + 1/8)
T*n*	Set Tempo to *n* (1-255). Normal tempo is 2.

OPERATORS: (may be used instead of absolute value *n* in the above settings)

+	Add 1 to the current value.
-	Subtract 1 from the current value.
>	Double the current value.
<	Halve the current value.
=*v*;	Use value from variable *v*.

SUB-STRINGS:

X*s$*;	Execute commands in string variable *s$*

PMODE *mode, page*

The PMODE statement sets the current graphics mode and/or the current page on which graphics operations take place. Either argument may be omitted if only one setting needs to be changed. When *mode* is provided, the current foreground and background colors are also reset to the system defaults for that mode.

Graphics Modes Supported in MCX Basic

MODE	RESOLUTION	COLORS	DEFAULT FOREGROUND	DEFAULT BACKGROUND
0	128 x 96	2	Green / Buff	Black
1	128 x 96	4	Red / Orange	Green / Buff
2	128 x 192	2	Green / Buff	Black

The *page* argument selects which page of graphics memory will be used for subsequent graphic operations. After a new page is selected, it will not be displayed until a SCREEN statement is executed (see page 15). You can draw into a graphics page without displaying it on the screen.

NOTES:

The mode and page are both set to 1 at system startup.

A timing problem in the MC-10 can cause modes 0 and 2 to produce a very noisy display. This depends on whether the VDG syncs with the rising or falling edge of the clock.

PRESET (*x, y*)

Reset the pixel identified by *x* and *y* in the current graphics page to the background color.

PRINT #*fileNumber, expresssion1, expression2, ...*

Writes the value of each expression to the designated data file. Each expression, whether string or numeric, is written to the file as a separate line of ASCII text terminated by a carriage return. The choice of delimiter used to separate the expressions (comma, semicolon or none) has no effect when PRINTing to a data file.

The file identified by *fileNumber* must have previously been opened in Output or Append mode using the OPEN statement (see page 12).

NOTE: *MCX Basic also accepts PRINT #-2 as an alternate for LPRINT.*

PRINT OFF
PRINT ON

Turns virtual printing On or Off. When virtual printing is On, any output directed to the printer is provided to the server instead. Virtual printing is enabled by default. Enter `PRINT OFF` if you want to use a printer connected directly to the MC-10 serial port.

PSET (*x, y, color*)

Set the pixel identified by *x* and *y* in the current graphics page to the specified *color*. If *color* is omitted then the current foreground color is used.

SAVE* *array, "filename"*

Save the contents of the numeric *array* to a file on the server.

SAVE SCREEN "*filename*", *page*

Saves a file containing the specified graphics *page* to the server. A *page* argument of 0 will save the current display screen. Page numbers from 0 to 5 may used when running MCX Basic in the Standard configuration. The Large configuration supports page numbers from 0 to 8.

SCREEN *type, colorSet*

The SCREEN statement updates the screen to show the specified display *type* and/or the new *color set*. Either argument may be omitted if only one setting needs to change.

When *type* is 0, the Text / Semigraphics screen is displayed. When *type* is 1, the Graphics screen is displayed. The Graphics screen is a combination of the mode and page most recently selected through a PMODE statement.

Each of the display types can be viewed using one of two color sets. You can change the active color set by providing either 0 or 1 for the *colorSet* argument. The new color set is only applied to the specified (or current) display type.

Color Sets

SET	TEXT	PMODE 0 or 2	PMODE 1
0	Black / Green	Black / Green	Green / Yellow / Blue / Red
1	Black / Amber	Black / Buff	Buff / Cyan / Magenta / Orange

NOTES:

`PRINT` and `INPUT` statements automatically switch the display type to Text, as does stopping or ending the program.

The contents of the Text screen are lost when the display switches to a Graphics screen. For this reason, a `CLS` is automatically performed when the display switches from Graphics to Text.

SWAP *var1, var2*

Exchanges the values of two variables. The variables must be of the same type (string or numeric). Individual elements within arrays may also be swapped.

Examples:

```
SWAP J,K
SWAP A$,B$
SWAP D$(N),D$(N+1)
```

TIMER = *number*

Sets the computer's TIMER value to *number*, which may be any integer from 0 to 65535. See the description of the TIMER function on page 19 for more information.

WAIT *milliseconds*

Causes the computer to pause for the specified number of milliseconds (0-65535).

NOTE: *The wait time is approximate.*

FUNCTIONS

ATN (*angle*)

Computes the arctangent of *angle*, which is given in radians. This is the inverse of the TAN function.

EOF (*fileNumber*)

Returns -1 if the End-Of-File has been reached. Returns 0 if more data is available. The *fileNumber* argument must refer to a data file that has been opened for Input.

ERRL

Returns the program line number in which the last error occurred. If no error has occurred since the program started running or since the last CLEAR ERROR statement was executed then 65535 will be returned.

NOTE: *An error produced in Direct Mode will also reset the value to 65535.*

ERRN

Returns a number representing the type of error which last occurred. See "ERROR CODES" on page 19 for a list of possible error numbers.

If no error has occurred since the program started running or since the last `CLEAR ERROR` statement was executed then -1 will be returned.

FIX (*number*)

The `FIX` function is similar to `INT` in that they both return the integer portion of a *number*. `FIX` differs from `INT` when the argument is negative. In that case, `FIX` returns the first integer that is greater than or equal to the argument whereas `INT` returns the first integer that is less than or equal to the argument. In other words, `FIX` simply chops off any digits to the right of the decimal point and never changes any digits on the left.

HEX$ (*number*)

The HEX$ function returns a string representing the hexadecimal value of *number*. The argument must be within the range of 0..65535.

INSTR (*position, subject, target, instance*)

The `INSTR` function searches the *subject* string for an instance of the *target* string. The value returned is either the position where the instance was found or 0.

The *position* argument is optional and specifies the first character position within the *subject* string where searching will begin. If *position* is omitted then searching begins from the first character in the *subject* string. If *position* is greater than the number of characters in *subject* then the function returns 0.

The *instance* argument is also optional and allows you to search for a specific instance of *target* in situations where *target* may occur more than once within the *subject* string. If *instance* is omitted then INSTR will search for the first instance of *target*. You can search for the last (right-most) instance of *target* by supplying an *instance* value of 0.

MAXVAL (*num1, num2*)

The MAXVAL function compares the two numeric arguments and returns the larger value.

MEMEND

The MEMEND function returns the highest memory address that is currently being used by the MCX Basic workspace.

You can use the CLEAR statement to take memory away from Basic's workspace and use it for some other purpose such as machine language routines.

MINVAL (*num1, num2*)

The MINVAL function compares the two numeric arguments and returns the smaller value.

POS (*device*)

POS returns the current line position on the specified output device. MCX Basic supports two output devices, the screen and a printer.

The device number for the screen is 0. Passing 0 as the argument to the POS function returns the current cursor position within a line on the screen (0 - 31).

The device number for a printer is -2. Passing -2 as the argument to the POS function returns the current column position of the printer's carriage.

PPOINT (*x, y*)

PPOINT returns the color value for the pixel identified by *x* and *y* in the current graphics page. The value returned will depend on the current graphics mode (set with PMODE) and the active color set (set with SCREEN).

STRING$ (*length, character*)

The STRING$ function produces a string value from a repeated character. The *length* argument specifies how many times to repeat the character in order to produce the result. The *character* argument may be a number representing the ASCII code of the character or it can be a string from which the first character will be used.

Examples:

`A$ = STRING$(32,"-")`	Assigns a string of 32 dashes to A$.
`PRINT STRING$(4,13)`	Prints 4 carriage returns to the screen.

STRMEM (*compact*)

The STRMEM function returns the amount of string space currently available. The *compact* argument indicates whether or not compaction should take place before calculating the result.

Passing 0 for *compact* bypasses compaction and returns the number of characters that can be allocated before an automatic compaction is triggered.

Passing 1 for *compact* forces an immediate compaction of string space. In this case, the value returned represents the true amount of unused string space.

TIMER

MCX Basic provides a built-in timer. The timer is set to zero when the computer is turned on and then begins incrementing once every sixtieth of a second (approximately). When the timer value reaches 65535, it starts over from 0.

NOTE: *The timer pauses during some operations including File I/O, Printing, RENUM and MERGE.*

ERROR CODES

Abbreviation	#	Description
NF	0	NEXT without FOR
SN	2	Syntax error
RG	4	RETURN without GOSUB
OD	6	Out of DATA
FC	8	Illegal Function Call
OV	10	Overflow
OM	12	Out of Memory
UL	14	Undefined Line number
BS	16	Bad Subscript
DD	18	Doubly Dimensioned array
D0	20	Division by Zero
ID	22	Illegal in Direct mode
TM	24	Type Mismatch
OS	26	Out of String space
LS	28	Length of String (too long)
ST	30	String formula is Too complex
CN	32	Can’t continue
IO	34	Input/Output error
FM	36	Wrong File Mode
DN	38	Bad Device Number (or un-allocated file number)
NE	40	File does Not Exist
WP	42	Write Protected
FN	44	Badly formed File (or directory) Name
FS	46	File System error
IE	48	Input past End of file
FD	50	Unacceptable File Data
AO	52	File is Already Open
NO	54	File is Not Open
DS	56	Direct Statement in file