



MGME has its own scripting language. This allows you to control many aspects of MGME and use the information it contains. The scripting language is a mix between BASIC and C style. In the future there may be some basic 'Wizards' to help you write the code, but for now, you do it yourself. Function names are BASIC style generally.

Basic Syntax

MGMEScript has the following properties:

- Strings are contained within "" or ''
- All calls to functions are in the form:
FunctionName([Variable1],[Variable2],[Variable3]...,[VariableX])
- Even if a function has not variables, you must have the brackets in place
- You must use () brackets
- You can set variables using syntax like:

variable = value

OR

variable += value

as well as -=.

Maths Functions

Sqr

This function square roots the number supplied

Syntax:

Sqr(*[Number]*)

Example:

Sqr(16)

Returns:

4

Round

This function rounds a number up or down to a certain number of decimal places

Syntax:

Round(*[Number]*,*[Number of decimal places]*)

Example:

Round(5.46, 1)

Returns:

5.5

DecToHex

This function converts a decimal number to hexadecimal value.

Syntax:

DecToHex(*[Decimal Number]*)

Example:

DecToHex(77)

Returns:

4D

HexToDec

This function converts a hexadecimal value to decimal number

Syntax:

HexToDec(*[Hexadecimal value]*)

Example:

HexToDec (4D)

Returns:

77

String Functions

Len

This function returns the length of a string.

Syntax:

`Len([String])`

Example:

`Len("Hello World")`

Returns:

11

Split

This function splits a string by using a specified substring and returns a specified part. The part number is 0 based, so if you specify the part number as 0, the first part will be returned, if you specify it as 1, the 2nd part will be returned etc.

Syntax:

`Split([Main String],[String to split with],[Part Number - 1])`

Example:

`Split("Hello,World,Testing,1,2,3","",2)`

Returns:

Testing

HexToASCII

This function will take a series of Hex values separated by spaces and convert each one into an ascii character and return them as a string.

Syntax:

`HexToASCII([Hex Number List Separated by spaces])`

Example:

`HexToASCII("64 68 69 66")`

Returns:

dhif

Memory Functions

Write

This function will write a value into one of the values in MGME's sets

Syntax:

Write([Set Name],[Value Name or Offset],[New value])

Example:

Write("Player","XPos",55)

Returns:

True

WritePure

This function will write a value into to anywhere in the game's memory

Syntax:

WritePure([Memory Address],[Length],[New value],[Data Type])

Valid data types are:

"Long"

"Single"

"Hex" - This will take a series of hexadecimal values separated by spaces

"Text" - This will write pure text

Example:

WritePure(0120302,4,64,"Long")

Returns:

True

SetAccess

This function sets the memory access rights for a specific memory address

Syntax:

`SetAccess([Address],[Length],[Access Rights])`

Valid Access rights are:

"READWRITE" – Read and write access

"READWRITEEXECUTE" - Read, write and execute access

"READ" – Read access only

"NOACCESS" – No access to the memory

Example:

`SetAccess(0120302,4,"READWRITEEXECUTE")`

Returns:

0 if unsuccessful

Winsock Functions

These functions allow you to connect to a remote computer and exchange data with it.

Sock.Connect

This function connects the socket to a specified remote address/port.

Syntax:

`Sock.Connect([Address],[Port])`

Example:

`Sock.Connect("192.168.0.8",774)`

Returns:

True

Sock.IP

This function returns the computer's local IP address.

Syntax:

`Sock.IP()`

Example:

`Sock.IP()`

Returns:

The local IP address

Sock.RemoteIP

This function returns the remote computer's IP address or host name if you are connected to it.

Syntax:

`Sock.RemoteIP()`

Example:

`Sock.RemoteIP()`

Returns:

The remote computer's IP address

Sock.State

This returns a winsock state code that indicated the state of the connection.

Syntax:

Sock.State()

Example:

Sock.State()

Returns:

An integer that indicated the state of the connection

Sock.Listen

This makes the socket listen for an incoming connection on the specified port

Syntax:

Sock.Listen(*[Port]*)

Example:

Sock.Listen(774)

Returns:

True

Sock.SendData

This sends data to the remote computer if the socket is connected

Syntax:

Sock.SendData(*[Data]*)

Example:

Sock.SendData("Hello World")

Returns:

True if the connection is connected, False if the socket was not connected

Sock.GetData

This retrieves all the data that has been sent to the socket since the last time GetData was called.

Syntax:

```
Sock.GetData()
```

Example:

```
Sock.GetData()
```

Returns:

The data that has been sent to the socket since the last time GetData was called.

Sock.Close

This closes the socket.

Syntax:

```
Sock.Close()
```

Example:

```
Sock.Close()
```

Returns:

```
True
```