$RESULT

Return value for some functions like FIND etc.
$RESULT_1 and $RESULT_2 are available for some commands.

$VERSION

Contains current version of OllyScript
Example
```
      cmp $VERSION, "0.8"
      ja version_above_08
```

#INC file

Includes a script file in another script file
Example:
```
      #inc "anotherscript.txt"
```

#LOG

Enables logging of executed commands.
The commands will appear in OllyDbg log window, and will be prefixed with
-->
Example:
```
      #log
```

ADD dest, src

Adds src to dest and stores result in dest
Example:
```
      add x, 0F
      add eax, x
      add [401000], 5
      add y, " times" // If y was 1000 before this command then y is
"1000 times" after it
```

AI
--
Executes "Animate into" in OllyDbg
Example:
```
      ai
```

ALLOC size

Allocate new memory page, you can read/write and execute.
Example
```
    alloc 1000
      free $RESULT, 1000
```

AN addr

Analyze module which contains the address addr.
Example:
```
      an eip // Same as pressing CTRL-A
```

```
AND dest, src
-------------
ANDs src and dest and stores result in dest
Example:
      and x, 0F
      and eax, x
      and [401000], 5

AO
--
Executes "Animate over" in OllyDbg
Example:
      ao

ASK question
------------
Displays an input box with the specified question and lets user enter a
response.
Sets the reserved $RESULT variable (0 if cancel button was pressed).
You have also the length in $RESULT_1 (divised by 2 for hex entries)
Example:
      ask "Enter new EIP"
      cmp $RESULT, 0
      je cancel_pressed
      mov eip, $RESULT

ASM addr, command [,version]
----------------------------
Assemble a command at some address.
Change version number (0,1,...) to get alternative code bytes, if
possible.
Returns bytes assembled in the reserved $RESULT variable
Example:
      asm eip, "mov eax, ecx"

ASMTXT addr, file
-----------------
Assemble a text asm file at some address.
Example:
      asmtxt EIP, "myasm.txt"

ATOI str [, base=16.]
-----------------
Converts a string to integer
Returns the integer in the reserved $RESULT variable
Example:
      itoa "F"
      itoa "10", 10.

BC addr
-------
Clear unconditional breakpoint at addr.
Example:
      bc 401000
      bc x
      bc eip
```

BD addr

Disables breakpoint at addr.
Example:
```
      bp 401000
      BD 401000
```

BEGINSEARCH [start]

Create a Copy of Debugged App Memory, Find commands will use this data faster.
You need to use ENDSEARCH before writing to memory and to free this memory copy.
Optimization time is 20% for 5000 loops... but could maybe be optimized
Example:
```
      mov count, 0
      mov start, eip
      beginsearch start
  next:
      find #00#, start
      cmp $RESULT,0
      je end
      mov start, $RESULT+1
      add count, 1
      jmp next
  end:
      endsearch
      msg count
```

BP addr

Set unconditional breakpoint at addr.
Example:
```
      bp 401000
      bp x
      bp eip
```

BPCND addr, cond

Set breakpoint on address addr with condition cond.
Example:
```
      bpcnd 401000, "ECX==1"
```

BPD callname

Remove breakpoint on dll call set by BPX

BPGOTO addr, label

Automatic Jump at label on Breakpoint (Standard(INT3) and Hardware).
EOB Like Command
Example:
```
      bphws addr
      bpgoto addr, MyLabel
  NextBP:
      RUN
      ...
  MyLabel:
      ...
      jmp NextBP
```

BPHWC addr

Delete hardware breakpoint at a specified address
Example:
```
bphwc 401000
```

BPHWS addr, [mode]

Set hardware breakpoint. Mode can be "r" - read, "w" - write or "x" - execute (default).
Example:
```
bphws 401000, "x"
```

BPL addr, expr

Sets logging breakpoint at address addr that logs expression expr
Example:
```
bpl 401000, "eax" // logs the value of eax everytime this line is passed
```

BPLCND addr, expr, cond

Sets logging breakpoint at address addr that logs expression expr if condition cond is true
Example:
```
bplcnd 401000, "eax", "eax > 1" // logs the value of eax everytime this line is passed and eax > 1
```

BPMC

Clear memory breakpoint.
Example:
```
bpmc
```

BPRM addr, size

Set memory breakpoint on read. Size is size of memory in bytes.
Example:
```
bprm 401000, FF
```

BPWM addr, size

Set memory breakpoint on write. Size is size of memory in bytes.
Example:
```
bpwm 401000, FF
```

BPX callname

Set breakpoint on dll call

BUF var

Converts string/dword variable to a Buffer
Example:
```
mov s, "123"
buf s
log s // output "#313233#
```

```
CMP dest, src [,size]
-------------
Compares dest to src. Works like it's ASM counterpart.
see SCMP to compare strings or memory data
Example:
      cmp y, x
      cmp eip, 401000
      je label
      cmp cx, x, 2
      je label

CMT addr, text
--------------
Inserts a comment at the specified address
Example:
      cmt eip, "This is the entry point"

COB
---
Makes script continue execution after a breakpoint has occured (removes
EOB)
Example:
      COB

COE
---
Makes script continue execution after an exception has occured (removes
EOE)
Example:
      COE

DBH
---
Hides debugger
Example:
      dbh

DBS
---
Unhides debugger
Example:
      dbs

DEC var
-------
Substracts 1 from variable
Example:
      dec v

DIV op1, op2
------------
Sets op1 with op1/op2
Example:
      div var, 2

DM addr, size, file
-------------------
Dumps memory of specified size from specified address to specified file
(default path set from opened app.)
Example:
      dm 401000, 1F, "c:\dump.bin"
```

DMA addr, size, file

Dumps memory of specified size from specified address to specified file appending to that file if it exists

Example:

```
dma 401000, 1F, "c:\dump.bin"
```

DPE filename, ep

Dumps the executable to file with specified name.
Entry point is set to ep.

Example:

```
dpe "c:\test.exe", eip
```

EOB label

Transfer execution to some label on next breakpoint.

Example:

```
eob SOME_LABEL
```

EOE label

Transfer execution to some label on next exception.

Example:

```
eob SOME_LABEL
```

ERUN

Executes SHIFT-F9 in OllyDbg. Run with Ignore Exceptions

Example:

```
erun
```

ESTI

Executes SHIFT-F7 in OllyDbg.

Example:

```
esti
```

ESTO

Executes SHIFT-F9 in OllyDbg. (OLD COMMAND, COULD BE REMOVED, USE ERUN)

Example:

```
esto
```

EVAL

Evaluates a string expression that contains variables.
The variables that are declared in the current script can be enclosed in curly braces {} to be inserted.
Sets the reserved $RESULT variable

Example:

```
var x
mov x, 1000
eval "The value of x is {x}" // after this $RESULT is "The value of x is 1000"
```

EXEC/ENDE

Executes instructions between EXEC and ENDE in the context of the target process.
Values in curly braces {} are replaced by their values.
Examples:

```
// This does some movs
mov x, "eax"
mov y, DEADBEEF
exec
      mov {x}, {y} // mov eax, 0DEADBEEF will be executed
      mov ecx, {x} // mov ecx, eax will be executed
ende

// This calls ExitProcess in the debugged application
exec
      push 0
      call ExitProcess
ende
ret
```

FILL addr, len, value

Fills len bytes of memory at addr with value
Example:

```
fill 401000, 10, 90 // NOP 10h bytes
```

FIND addr, what

Searches memory starting at addr for the specified value.
When found sets the reserved $RESULT variable. $RESULT == 0 if nothing found.
The search string can also use the wildcard "??" (see below).

Example:

```
find eip, #6A00E8# // find a PUSH 0 followed by some kind of call
find eip, #6A??E8# // find a PUSH 0 followed by some kind of call
```

FINDCALLS addr [,name]

Find all intermodular calls (dll calls) in the disasm area.
You can filter results by label (case insensitive) with the optionnal second parameter.
Reference Window is used and its content changed
Then can use GREF to get results count and retrieve them.

Example:

```
findcalls eip, "exit"
gref
msg $RESULT
```

FINDCMD addr, cmdstr

Search for asm command(s), you can search for series also with ";" separator.
This command uses "Search for All Sequences" Ollydbg function so could find relative calls/jmp
Reference Window is used and its content changed
You can use GREF to get next results in disasm window range

Example 1:
```
      mov line,1
      findcmd eip, "xor R32,R32"
  next:
      gref line
      cmp $RESULT,0
      je finished
      inc line
      jmp next
  finished:
```

Example 2:
```
      findcmd 401000, "nop;nop;nop"
      msg $RESULT
```

FINDOP addr, what

Searches code starting at addr for an instruction that begins with the specified bytes.
When found sets the reserved $RESULT variable. $RESULT == 0 if nothing found.
The search string can also use the wildcard "??" (see below).
Example:
```
      findop 401000, #61# // find next POPAD
      findop 401000, #6A??# // find next PUSH of something
      findop 401000, "1" // = #61#
```

FINDMEM what [, StartAddr]

Searches whole memory for the specified value.
When found sets the reserved $RESULT variable. $RESULT == 0 if nothing found.
The search string can also use the wildcard "??" (see below).
Example:
```
      findmem #6A00E8# // find a PUSH 0 followed by some kind of call
      findmem #6A00E8#, 00400000 // search it after address 0040.0000
```

FREE addr [, size]

Free memory bloc allocated by ALLOC (or not).
If size not given, drop whole memory bloc.
Example
```
    alloc 1000
      free $RESULT
```

GAPI addr #BETA#

Chinese Translation
Obtains the code place API call information
The API information saves in preservation variable $RESULT.
If the symbolic name is a API function, then
$RESULT saves the API information
$RESULT_1 save link base/storehouse (for instance kernel32)
$RESULT_2 save symbolic name (for instance ExitProcess).
$RESULT_3 save calling location (for instance call xxxxx)
$RESULT_4 save destination

Notice: This and the GN difference is GN must point to the IAT address
But GAPI gives the code address to be possible directly to obtain API
Also has, if you have gotten down the software break point in here,
please first clear the break point to use this sentence again, because
the software break point modified the code is CC
If here does not clear here the software break point, will create this
not to be able the very good recognition.
Example:
```
GAPI 401000 (call kernel32.ExitProcess)
GAPI the EIP // examined whether the current code is API calls, is
```
not then returns to 0

GBPM (beta)

Get last memory breakpoint address, affects $RESULT with dword value

GBPR

Get last breakpoint reason, affects $RESULT with dword value
Example:
```
GBPR
cmp $RESULT, 10
je SelectNormalBP
cmp $RESULT, 20
je SelectMemBP
cmp $RESULT, 40
je SelectHwBP
jmp NextBP
```

GCI addr, info

Gets information about asm command
"info" can be :
- COMMAND for asm command string (like OPCODE)
- DESTINATION for Destination of jump/call/return
- SIZE for number of command bytes
- TYPE for asm command string (one of C_xxx, see OllyDbg Plugin
API)

Example:
```
GCI eip, DESTINATION
```

GCMT addr

Gets the comment, automatic comment or analyse's comment at specified
code address

GMA name, info

Calls GMI, but parameter is short name of the module

GMEMI addr, info

Gets information about a memory block to which the specified address belongs.
"info" can be MEMORYBASE, MEMORYSIZE or MEMORYOWNER (if you want other info in the future versions plz tell me).
Sets the reserved $RESULT variable (0 if data not found).
Example:
```
      GMEMI addr, MEMORYBASE // After this $RESULT is the address to the memory base of the memory block to which addr belongs
```

GMI addr, info

Gets information about a module to which the specified address belongs.
"info" can be :
MODULEBASE, MODULESIZE, CODEBASE, CODESIZE, MEMBASE, MEMSIZE,
ENTRY, NSECT, DATABASE, RELOCTABLE, RELOCSIZE
RESBASE, RESSIZE, IDATABASE, IDATATABLE, EDATATABLE, EDATASIZE
and strings NAME, PATH, VERSION
(if you want other info in the future versions plz tell me).
Sets the reserved $RESULT variable (0 if data not found).
Example:
```
      GMI eip, CODEBASE // After this $RESULT is the address to the codebase of the module to which eip belongs
```

GN addr

Gets the symbolic name of specified address (ex the API it poits to)
Sets the reserved $RESULT variable to the name. If that name is an API $RESULT_1 is set to the library (ex kernel32) and $RESULT_2 to the name of the API (ex ExitProcess).
Example:
```
      gn 401000
```

GO addr

Executes to specified address (like G in SoftIce)
Example:
```
      go 401005
```

GPA proc, lib, [0,1]

Gets the address of the specified procedure in the specified library.
When found sets the reserved $RESULT variable. $RESULT == 0 if nothing found.
Useful for setting breakpoints on APIs.
Set third param to 1 if you want to keep library in memory
Example:
```
      gpa "MessageBoxA", "user32.dll" // After this $RESULT is the address of MessageBoxA and you can do "bp $RESULT".
```

GPI key

Gets process information, one of :
HPROCESS,PROCESSID,HMAINTHREAD,MAINTHREADID,MAINBASE,PROCESSNAME,EXEFILENAME,CURRENTDIR,SYSTEMDIR

GREF [line]

Get Address from Reference Window at Line. First line is 1 because 0 is CPU Initial EIP.
Without parameter, GREF results the Reference Window number of entries.
Example:

```
FINDCMD "push eax"
GREF 1
msg $RESULT
GREF 2
msg $RESULT
```

GRO addr

Get Relative Offset
When found sets the reserved $RESULT variable. $RESULT == 0 if nothing found.

HANDLE x, y, class

Returns the handle of child window of specified class at point x,y (remember: in hex values).

HISTORY (0,1)

Enables or Disables Value history in Script Progress Window, could optimize loops
Example:

```
history 0 //disable
history 1 //enable
```

INC var

Adds 1 to variable
Example:

```
inc v
```

ITOA n [, base=16.]

Converts an integer to string
Returns the string in the reserved $RESULT variable
Example:

```
itoa F
itoa 10., 10.
```

JA label

Use this after cmp. Works like it's asm counterpart.
Example:

```
ja SOME_LABEL
```

JAE label

Use this after cmp. Works like it's asm counterpart.
Example:

```
jae SOME_LABEL
```

JB label

Use this after cmp. Works like it's asm counterpart.
Example:
```
jb SOME_LABEL
```

JBE label

Use this after cmp. Works like it's asm counterpart.
Example:
```
jbe SOME_LABEL
```

JE label (JZ)

Use this after cmp. Works like it's asm counterpart.
Example:
```
je SOME_LABEL
```

JMP label

Unconditionally jump to a label.
Example:
```
jmp SOME_LABEL
```

JNE label (JNZ)

Use this after cmp. Works like it's asm counterpart.
Example:
```
jne SOME_LABEL
```

KEY vkcode [, shift [, ctrl]]

Emulates global keyboard shortcut.
Example:
```
key 20
key 20, 1 //Shift+space
key 20, 0, 1 //Ctrl+space
```

LBL addr, text

Inserts a label at the specified address
Example:
```
lbl eip, "NiceJump"
```

LC

Clear Main Log Window

LCLR

Clear Script Log Window

LEN str

Get length of a string
Example:
```
len "NiceJump"
msg $RESULT
```

LM addr, size, filename

---

load Dm file to mem
LM is the opposite of the DM command
Example:

```
  lm 0x401000, 0x100, "test.bin"
```

LOG src [,prefix]

Logs src to OllyDbg log window.
If src is a constant string the string is logged as it is.
If src is a variable or register its logged with its name.
You can replace default prefix with the optional second parameter.
Example:

```
      log "Hello world" // The string "Hello world" is logged
      var x
      mov x, 10
      log x // The string "x: 00000010" is logged.
      log x, "" // The string "00000010" is logged.
```

LOGBUF var [,linecount [,separator]]

Logs a string or buffer like a memory dump, usefull for long data

MOV dest, src [,size]

Move src to dest.
Src can be a long hex string in the format #<some hex numbers>#, for example #1234#.
Remember that the number of digits in the hex string must be even, i.e. 2, 4, 6, 8 etc.
Example:

```
      mov x, 0F
      mov y, "Hello world"
      mov eax, ecx
      mov [ecx], #00DEAD00BEEF00#
      mov !CF, 1
      mov !DF, !PF
      mov [403000], "Hello world"
```

MEMCPY dest,src,size

Copy app. memory from "src" address to "dst" address.
This function is same as mov [dst],[src],size
Example:

```
      gma "OLE32",CODEBASE
      mov base, $RESULT
      gma "OLE32",CODESIZE
      mov size, $RESULT
      alloc size
      mov dst, $RESULT
      MEMCPY dst,base,size
      ...
      free dst
```

MSG message

Display a message box with specified message
Example:

```
      MSG "Script paused"
```

MSGYN message

Display a message box with specified message and YES and NO buttons.
Sets the reserved $RESULT variable to 1 if YES is selected and 0
otherwise.
Example:
```
MSGYN "Continue?"
```

MUL op1, op2

Sets op1 with op1*op2
Example:
```
mul op1, 10
```

NEG op

Assembly Operation "neg eax"

NOT op

Assembly Operation "not eax"

OR dest, src

ORs src and dest and stores result in dest
Example:
```
or x, 0F
or eax, x
or [401000], 5
```

OPCODE addr

OPCODE sets the $RESULT variable to the opcode bytes, $RESULT_1 variable
to mnemonic opcode (i.e. "MOV ECX,EAX")
and $RESULT_2 to the length of the opcode.
If an invalid opcode appears, $RESULT_2 should be 0.
addr is increased by the length of the opcode (disassemble command).
With this function you can step forward through code.
Example:
```
opcode 00401000
```

OPENDUMP addr [,base,size]

Create a new Dump Window with data at address.

OPENTRACE

Opens run trace window

PAUSE

Pauses script execution. Script can be resumed from plugin menu.
Example:
```
pause
```

POP dw

Retrieve dword from stack

PREOP addr

Get asm command line address just before specified address.
Attention: Will not give real executed command eip before the jump.
Example:
```
preop eip
```

PUSH dw

Add dword to stack

READSTR str, len

Copy len chars of str into $RESULT

REF addr

REF addr works as "Find references to .. Selected command" and "Find references", Ctrl R, in OllyDbg.
$RESULT variable is set to the first reference addr
$RESULT_1 to the opcode (text asm command)
$RESULT_2 to the comment (like reference window).
Repeat "REF addr" until $RESULT=0 to get next refs
Example:
```
continue:
    REF eip
    log $RESULT
    log $RESULT_1
    log $RESULT_2
cmp $RESULT,0
jne continue
```

REPL addr, find, repl, len

Replace "find" with "repl" starting at "addr" for "len" bytes.
Wildcards are allowed
Example:
```
repl eip, #6a00#, #6b00#, 10
repl eip, #??00#, #??01#, 10
repl 401000, #41#, #90#, 1F
```

RET

Exits script.
Example:
```
ret
```

REV what

Reverse dword bytes.
Example:
```
rev 01020304
//$RESULT = 04030201
```

ROL op, count

Assembly Operation "rol eax, cl"
save in the target (first) operand.

ROR op, count

Assembly Operation "ror eax, cl"
Example:
```
mov x, 00000010
ROR x, 8
```

RTR

Executes "Run to return" in OllyDbg, [Ctrl+F9] operation.
Example:
```
rtr
```

RTU

Executes "Run to user code" in OllyDbg, [Alt+F9] operation.
Example:
```
rtu
```

RUN

Executes F9 in OllyDbg, you can also use ERUN to ignore exceptions
Example:
```
run
```

SCMP dest, src [,size]

Compares strings dest to src. Works like it's ASM counterpart.
Example:
```
cmp x, "KERNEL32.DLL"
cmp [eax], "Hello World", 11.
je Label
```

SCMPI dest, src [,size]

Compares strings dest to src (case insentitive). Works like it's ASM counterpart.
Example:
```
cmp sVar, "KERNEL32.DLL"
cmp [eax], "Hello", 5
jne Label
```

SETOPTION

Open the OllyDBG Options Window, to change debugging parameters.
Script will continue on close.

SHL dest, src

Shifts dest to the left src times and stores the result in dest.
Example:
```
mov x, 00000010
shl x, 8 // x is now 00001000
```

SHR dest, src

Shifts dest to the right src times and stores the result in dest.
Example:
```
mov x, 00001000
shr x, 8 // x is now 00000010
```

STI

Execute F7 in OllyDbg. STep Into.
Example:
```
sti
```

STO

Execute F8 in OllyDbg. STep Over.
Example:
```
sto
```

STR var

Converts variable to a String (buffer or dword)

SUB dest, src

Reduce src from dest.
Example:
```
sub x, 0F
sub eax, x
sub [401000], 5
```

TC
--
Cancels run trace in OllyDbg
Example:
```
tc
```

TEST dest,src

Performs a logical AND of the two operands updating the flags register
without saving the result.
(Modifies Flags: CF OF PF SF ZF (AF undefined))

TI
--
Executes "Trace into" in OllyDbg, CTRL-F7 in OllyDbg.
Example:
```
ti
```

TICK [var [,reftime]]

Set variable with script execution time (microsec)
if reftime parameter is set, set $RESULT with time since reftime.
if no parameter is set, function set $RESULT with execution time in text,
in "<ssss mmm> ms" format
var is declared automatically.
Example:
```
tick time
msg time          //time since script startup
tick time,time
msg $RESULT       //time since last TICK, DWORD value
```

TICND cond

Traces into calls until cond is true
Example:
```
ticnd "eip > 40100A" // will stop when eip > 40100A
```

TO
--
Executes "Trace over" in OllyDbg
Example:
```
      to
```

TOCND cond

Traces over calls until cond is true
Example:
```
      tocnd "eip > 40100A" // will stop when eip > 40100A
```

UNICODE enable

Set Unicode Mode, not used for the moment
Example:
```
      UNICODE 1
      ...
```

VAR

Declare a variable to be used in the script.
Example:
```
      var x
```

XOR dest, src

XORs src and dest and stores result in dest
Example:
```
      xor x, 0F
      xor eax, x
      xor [401000], 5
```

XCHG dest, src

Exchanges contents of source and destination.

WRT file, data

Write to file (replace existing one) the only accepted symbol is "\r\n"
Numbers are wrote as strings... for the moment
Example:
```
      wrt "out.txt", "Data:\r\nOk\r\n"
      wrt sFile, ebx
```

WRTA file, data [, separator]

Append to file, default separator is "\n"
Example:
```
      wrta sFile, "hello world"
      wrta sFile, ABCD, ""
      wrta sFile, "Windows CR, "\r\n"
```