Colour Palette and Animation Commands

Instructions and functions to animate and manage colours

Get Palette INDEX, MASK
Grab the palette of another screen in the current screen
Parameters:
INDEX: The index of the source screen
MASK: A mask of bits in which bits set to one will enforce the copy of the colour and 0 will ignore the color. Only for the first 32 colours (optional)

Get Images Palette MASK
Copy the colour palette from the Images bank to the current screen
Parameters:
MASK: Mask of bits where each bit set to 1 represent a colour to copy and 0 a colour to ignore, up to 32 (optional)

Colour COLORINDEX, RGB
Set the RGB value of a colour in the palette the curretn screen
Parameters:
COLORINDEX: The index of the colour in the palette
RGB: The RGB value of the new colour (example: $FF0000 will change the color to RED)

Colour COLORINDEX
Return the RGB value of a colour from the palette of the current screen
Parameters:
COLORINDEX: The index of the colour in the palette
Value returned:
integer: The RGB value of the new colour (example: will return $FF0000 for RED)

RGB$ R, G, B
Return a string with the given RGB colors
Parameters:
R: The value of the red component, from 0 to 255
G: The value of the green component, from 0 to 255
B: The value of the blue component, from 0 to 255
Value returned:
string: The RGB string in the form: "RRGGBB"

RGB R, G, B
Return a number with the RGB value encoded
Parameters:
R: The value of the red component, from 0 to 255
G: The value of the green component, from 0 to 255
B: The value of the blue component, from 0 to 255
Value returned:
integer: The RGB value to use in Set Color

Colour COLORINDEX
Return the RGB value of a colour from the palette of the current screen

Parameters:

COLORINDEX: The index of the colour in the palette

Value returned:

integer: The RGB value of the new colour (example: will return $FF0000 for RED)

Flash Off

TODO! Turns off the flashing colours sequence

Flash COLORINDEX, DEFINITION$

TODO! Set the flashing color sequence

Parameters:

COLORINDEX: The index of the colour in the current screen palette

DEFINITION$: The definition of the flashing sequence

Flash

TODO! Turn off all colour shifts for current screen

Shift Up DELAY, FIRSTCOLOR, LASTCOLOR, ROTATELASTCOLOUR

TODO! Rotate the colour values of the current screen upwards

Parameters:

DELAY: Number of 1/1000th of second between shifts (PC mode) and 1/50th of the second (Amiga and retro machine emulation)

FIRSTCOLOR: Index of the first colour in the palette to shift

LASTCOLOR: Index of the last colour in the palette to shift

ROTATELASTCOLOUR: Flag indicate whether to copy the last color back to the first color (True) or discard it (false)

Shift Down DELAY, FIRSTCOLOR, LASTCOLOR, ROTATEFIRSTCOLOUR

TODO! Rotate the colour values of the current screen downwards

Parameters:

DELAY: Number of 1/1000th of second between shifts (PC mode) and 1/50th of the second (Amiga and retro machine emulation)

FIRSTCOLOR: Index of the first colour in the palette to shift

LASTCOLOR: Index of the last colour in the palette to shift

ROTATEFIRSTCOLOUR: Flag indicate whether to copy the first color back to the last color (True) or discard it (false)

Fade DELAY, INDEX, MASK

TODO! Blend the colours of the current screen to the values of the palette of another screen

Parameters:

DELAY: Number of 1/1000th of second between changes (PC mode) and 1/50th of a second for Amiga and retro machine emulation

INDEX: The index of the screen to fade the palette to

MASK: A mask of bits indicating which colour to fade, bits set to 1 beign taken into account by their number in the mask

fade DELAY, COLOURLIST

TODO! Blend colours to new values

Parameters:

DELAY: Number of 1/1000th of second between changes (PC mode) and 1/50th of a second for Amiga and retro machine emulation

COLOURLIST: A list of RGB values, separated by commmas indicating the new colours to fade to (op[tional)

Default

Restore the display to the state it was when the application start. Close all screen, all windows, detroys all bobs and sprites and open the default screen

Developing for mobile devices

Set of instructions and functions to manage mobile devices with AOZ.

Touch Screen
Returns True if the device screen is touch sensitive, otherwise False.
Value returned:
integer: Returns True if the device screen is touch sensitive, otherwise False..

Touch Count
Returns the number of touches on the device's touch screen.
Value returned:
integer: Returns an integer value which can be 0 (no touch) or the number of touches.

Touch X INDEX
Returns the horizontal position of a touch on the screen.
Parameters:
INDEX: Number of the touchdown whose horizontal position is to be recovered.
Value returned:
integer: Returns an integer value corresponding to the horizontal position of the touch.

Touch Y INDEX
Returns the vertical position of a touch on the screen.
Parameters:
INDEX: Number of the touchdown whose vertical position is to be recovered.
Value returned:
integer: Returns an integer value corresponding to the vertical position of the touch.

Touch State INDEX
Description of the function.
Parameters:
INDEX: Description of the parameter.
Value returned:
integer: Description of the value returned.

Touch On Change PROCNAME$
Assign an AOZ procedure to touch-screen events.
The procedure is called up each time an event occurs on the touch screen.
For each event, the following information is sent to the procedure
- X: Current X position of the touch
- Y: Current Y-position of the touch
- LASTX: Last X position of the touch
- LASTY: Last Y position of the touch
- STATE: State of the touch (1=START, 2=MOVE, 3=END)

Parameters:
PROCNAME$: Name of the procedure to be assigned

Orientable
Returns True if the device screen is orientable, otherwise False.

Value returned:
integer: Returns True if the device screen is orientable, otherwise False..

Orientation X
Returns the X angle of the orientation screen.
Value returned:
integer: Returns an integer value between -90 and 90 degres.

Orientation Y
Returns the Y angle of the orientation screen.
Value returned:
integer: Returns an integer value between -180 and 180 degres.

Orientation Z
Returns the Z angle of the orientation screen.
Value returned:
integer: Returns an integer value between 0 and 360 degres.

Accelerometer
Returns True if the device has an accelerometer, otherwise False.
Value returned:
integer: Returns True if the device has an accelerometer, otherwise False..

Acceleration X
Returns the acceleration value on the X-axis of the device.
Value returned:
integer: Returns the acceleration value on the X-axis of the device.

Acceleration Y
Returns the acceleration value on the Y-axis of the device.
Value returned:
integer: Returns the acceleration value on the Y-axis of the device.

Acceleration Z
Returns the acceleration value on the Z-axis of the device.
Value returned:
integer: Returns the acceleration value on the Z-axis of the device.

Orientation
Returns the orientation of the device. 0=Landscape, 1=Portrait
Value returned:
integer: Returns the orientation of the device. 0=Landscape, 1=Portrait

Lock Orientation ORIENTATION$
Description of the instruction.
Parameters:
ORIENTATION$: Description of the parameter.

Dialogs commands

Instructions and functions to display and handle dialogs. This set of instruction will be implemented the "Javascript way" in future versions of AOZ, and will allow your to create modern interface in a simple way

Dialog Box INTERFACE$, VALUE, PARAMETER$, X, Y
TODO! Display dialogue box on screen
Parameters:
INTERFACE$:
VALUE:
PARAMETER$:
X:
Y:

Dialog Open CHANNEL, INTERFACE$, NVAR, BUFFER
TODO! Display dialogue box on screen
Parameters:
CHANNEL:
INTERFACE$:
NVAR:
BUFFER:

Dialog Close CHANNEL
TODO! Display dialogue box on screen
Parameters:
CHANNEL:

Dialog Run CHANNEL, LABEL, X, Y
TODO! Display dialogue box on screen
Parameters:
CHANNEL:
LABEL:
X:
Y:
Value returned:
integer: 0 in this version

Dialog CHANNEL
TODO! Return status of an open dialogue box
Parameters:
CHANNEL:
Value returned:
integer: 0 in this version

RDialog CHANNEL, BUTTON, OBJECT
TODO! Read the status of a zone or button
Parameters:
CHANNEL:

BUTTON:
OBJECT:

Value returned:

integer: 0 in this version

RDialog$ CHANNEL, BUTTON, OBJECT

TODO! Read the status of a zone or button

Parameters:

CHANNEL:
BUTTON:
OBJECT:

Value returned:

integer: 0 in this version

EDialog

TODO! Find an error in an Interface program

Value returned:

integer: 0 in this version

Dialog Clr CHANNEL

TODO! Clear a dialogue box

Parameters:

CHANNEL:

Dialog Update CHANNEL, ZONE1, ZONE2, ZONEXXX

TODO! Uupdate a zone of a dialog box

Parameters:

CHANNEL:
ZONE1:
ZONE2:
ZONEXXX:

Dialog Freeze CHANNEL

TODO! Stop dialogue channel input

Parameters:

CHANNEL:

Dialog Unfreeze CHANNEL

TODO! Restart dialogue channel input

Parameters:

CHANNEL:

Resource Bank BANK_INDEX

TODO! Select a bank to be used for resources

Parameters:

BANK_INDEX:

Resource$ MESSAGE_INDEX

TODO! Read a message from the Resource Bank

Parameters:

MESSAGE_INDEX:

Value returned:
string: "" in this version

Resource Screen Open NUMBER, WIDTH, HEIGHT, FLASH
TODO! Open a screen using the resource settings
Parameters:
NUMBER:
WIDTH:
HEIGHT:
FLASH:

Resource Uppack NUMBER, X, Y
TODO! Unpack an image from the Resource Bank
Parameters:
NUMBER:
X:
Y:

ZDialog CHANNEL, PARAM1, PARAM2
TODO! Return the Z position of a dialog (?)
Parameters:
CHANNEL:
PARAM1:
PARAM2:
Value returned:
integer: 0 in this version

Display and Renderer Commands

Instructions and functions to manage the display and the renderer system

Freeze
Prevent the display to be updated while keeping the application runnin in the back

Unfreeze
Restore display update after "Freeze" has been used.

View
If "Auto View Off" has been used, update the display immediately

Auto View Off
Toggle viewing mode off

Auto View On
Toggle viewing mode on

Update Off
Turn off the automatic Object re-drawing system

Update On
Turn on the automatic Object re-drawing system

Update Every DELAY
Fix the rate of the automatic update system
Parameters:
DELAY: Time, in 1/1000th of second (PC mode) and 1/50th of seconds (Amiga / retro emulations) between the update of the display

Update
Force an update of the display. Will only work in "Auto Mode Off" mode.

Display Width
Return the width of the display canvas in PC mode, and the width of the emulated TV set in retro-machine emulation mode

Display Height
Return the width of the display canvas in PC mode, and the width of the emulated TV set in retro-machine emulation mode, this value being different in NTSC and PAL emulations

NTSC
Test if teh display is emulating a NTSC TV set.
Value returned:
boolean: False if the application is in PC mode or if the emulation is PAL

E-learning and SCORM API.

Connect your AOZ program to your Learning System Management (LMS).
These commands are compatible with SCORM 1.2 and 2004.

LMS Debug On
Turn On the SCORM debugger.

LMS Debug Off
Turn Off the SCORM debugger.

LMS Initialize
Find and start the SCORM API and initialize the tracking.

LMS Value$ KEYWORD$
Return the value of a SCORM variable from the API.
Parameters:
KEYWORD$: Name of the variable to find.
Value returned:
string: Value of the variable

LMS Value KEYWORD$, VALUE$
Set the value of a SCORM variable.
Parameters:
KEYWORD$: Name of the variable to set.
VALUE$: Value of this variable.

LMS Error$
Get the code of the last error of SCORM from the API.
Value returned:
string: Code of error ('0' = no error)

LMS Error Text$
Get the text of an error from this code.
Value returned:
string: The text of the error

LMS Diagnostic$
Get the text of the diagnostic for an error.
Value returned:
string: The text of the diagnostic

LMS Commit
Commit all the SCORM variables on the Learning System.

LMS Finish
Stop all communication with the SCORM API and the Learning System.

File Commands

Functions and instructions to handle file access

Load Image PATH$, INDEX, TAGS$
Load an image into a screen
Parameters:
PATH$: Path to the image to load
INDEX: Index of the screen to create. If ommited the image will be loaded in the current screen
TAGS$: List of tags indicating how to load the image. "#left", "#center", "#right" aligns the image horizontally, "#top", "#middle", "#bottom" aligns the image vertically, "fit" resizes the image to fit the screen, "paste" does not resize the image

Load Iff PATH$, INDEX, TAGS$
Load an IFF image into a screen (deprecated, use "Load Image")
Parameters:
PATH$: Path to the image to load
INDEX: Index of the screen to create. If ommited the image will be loaded in the current screen
TAGS$: List of tags indicating how to load the image. "#left", "#center", "#right" aligns the image horizontally, "#top", "#middle", "#bottom" aligns the image vertically, "fit" resizes the image to fit the screen, "paste" does not resize the image

Save Iff PATH$, INDEX
TODO! Save an IFF image out of a screen (deprecated, use "Save Image")
Parameters:
PATH$: Path to the image to save
INDEX: Index of the screen to save. If ommited the image will be saved from the current screen

Load PATH$, INDEX
Load a previously saved memory bank, or a set of banks
Parameters:
PATH$: Path to the bank(s) to load
INDEX: Index of the bank to load into. If not specified, the bank will be loaded at the same number it was saved

BLoad PATH$, INDEX
Load a binary file into a bank
Parameters:
PATH$: Path to the binary file to load
INDEX: Index of the bank to load into. This bank must be of "Data" or "Work" type.

BSave PATH$, START, END
Save the content of a bank to a binary file
Parameters:
PATH$: Path to the binary file to create
START: Memory address of the data to save. Use the Start() function to obtain the address
END: End of the memory zone to save. Use the Start() function to obtain the address and add the desired length

Bsave FILE$, NUMBER1, NUMBER2
Description of the instruction.

Parameters:
FILE$: Description of the parameter.
NUMBER1: Description of the parameter.
NUMBER2: Description of the parameter.

Pload FILE$, NUMBER1
Description of the instruction.
Parameters:
FILE$: Description of the parameter.
NUMBER1: Description of the parameter.

Save PATH$, START, END
Saves anything to a file, using the most appropriate format, based on the file extension
Parameters:
PATH$: Path to the file to create
START: Index of the first bank to save
END: Index of the last bank to save

Save Bank PATH$, START, END
Save one or several memory banks to a file. This file can later be loaded with "Load Bank"
Parameters:
PATH$: Path to the file to create
START: Index of the first bank to save
END: Index of the last bank to save

DFree
Returns the amount of free space of the disc pointed to by the current path
Value returned:
integer: The amount of free space

MkDir PATH$
Creates a new directory
Parameters:
PATH$: Path to the new directory to create

Open Random CHANNEL, PATH$
Open a random access file
Parameters:
CHANNEL: Number of the channel
PATH$: Path to the file to open

Open Random CHANNEL, PATH$
Open a file for input only
Parameters:
CHANNEL: Number of the input channel
PATH$: Path to the file to open

Open Out CHANNEL, PATH$
Open a file for input only. The file is replaced by the new one
Parameters:
CHANNEL: Number of the input channel

PATH$: Path to the file to open

Append CHANNEL, PATH$
Open a file for output, and add content at the end of it
Parameters:
CHANNEL: Number of the channel
PATH$: Path to the file to open

Assign NAME$, PATH$
Assign a name to a file or device
Parameters:
NAME$: The name to assign
PATH$: The path or drive name to assign it to

Put CHANNEL, RECORD_NUMBER
Output a record to a random access file
Parameters:
CHANNEL: The index of the channel with the open file
RECORD_NUMBER: The number of the record to output

Get CHANNEL, RECORD_NUMBER
Read a record from a random access file
Parameters:
CHANNEL: The index of the channel with the open file
RECORD_NUMBER: The number of the record to read

LOF CHANNEL
Return the length of a "Out", "In" or "Random access" channel
Parameters:
CHANNEL: Index of the channel

EOF CHANNEL
Indicates if the file pointer of a channel is located at the end of the file
Parameters:
CHANNEL: Index of the channel
Value returned:
boolean: True if the file pointer has reached the end, False if not

POF CHANNEL
Set the position of the file pointer in an open channel (type must be "Out" ort "Append")
Parameters:
CHANNEL: Index of the channel

POF CHANNEL
Return the position of the file pointer in an open channel
Parameters:
CHANNEL: Index of the channel
Value returned:
integer: the current position of the file pointer

Close CHANNEL

Close one or all opened files, and in case of output, save the buffered data into it
Parameters:
CHANNEL: Number of the channel (optional)

Parent
Change the current directory to the parent directory

Rename PATH$, NEWNAME$
Rename a file
Parameters:
PATH$: Path to the file to rename
NEWNAME$: New name of the file

Kill PATH$
Delete a file
Parameters:
PATH$: Path to the file to delete

Open Port CHANNEL, PORT$
TODO! Open a communication port on the machine
Parameters:
CHANNEL: Number of the input channel
PORT$: Name of the port to open

Port CHANNEL
TODO! Return the content of an opened harware port
Parameters:
CHANNEL: Number of the port
Value returned:
integer: Value reported by the port

Drive DRIVE$
Description of the function.
Parameters:
DRIVE$: Description of the parameter.
Value returned:
integer: Description of the value returned.

FSel$ PATH$, DEFAULT$, TITLE1$, TITLE2$
TODO! Open a file selector and return the name of the selected file
Parameters:
PATH$: Path to the directory to display at start
DEFAULT$: Name of the default file to display as selected (optional)
TITLE1$: String to display as title of the selector (optional)
TITLE2$: String to display as secondary title of the selector (optional)
Value returned:
integer: Value reported by the port

Dir First$ PATH$
List the indicated path internally and return the first file in the list
Parameters:

PATH$: Path to the directory to scan, can include * and ? wildcards
Value returned:
string: The name of the first file found

Dir Next$
Return the next file in the list generated by Dir First$
Value returned:
string: The name of the next file found

Exist PATH$
Indicates if a file or directory exists on the disc
Parameters:
PATH$: Path to the file or directory to check
Value returned:
boolean: True if the file or directory exist, False if not

File Name$ PATH$
Returns the name of a file, without extension
Parameters:
PATH$: Path to the file
Value returned:
string: The name of the file

File Extension$ PATH$
Returns the extension of a path to a file, without the dot
Parameters:
PATH$: Path to the file
Value returned:
string: The extension of the path to a file

File Extension$ PATH$
Returns the extension of a path to a file, without the dot
Parameters:
PATH$: Path to the file
Value returned:
string: The extension of the path to a file

File Length PATH$
Return the length in bytes of a file
Parameters:
PATH$: Path to the fileto check
Value returned:
integer: The length of the file

Dir PATH$
List a directory in the current screen
Parameters:
PATH$: Path to the directory to list, can include wildcards * and ?

Set Dir PATH$
Changes the current y

Parameters:
PATH$: Path to the new directory

Disc Info$ PATH$
Return information about the current drive
Parameters:
PATH$: Path to the drive or a directory on the drive (optional)

LDir PATH$
TODO: Output the content of a directory to the printer
Parameters:
PATH$: Path to the drive or a directory on the drive (optional)

Mask Iff MASK
TODO:Indicate what sections of an IFF file to load the next time the Load IFF instruction is used
Parameters:
MASK: Bitmask indicating the load. Example %100: Load palette of picture only, %10000: Load bitmaps only

Command Line$ LINE
Reserved variable. Return or set the parameters of the command that has been used to launch thhe application. For HTML applications, will return the section of the URL after "?"
Parameters:
LINE: The text to use as parameters, separated with commansss inside of the string

Command Line$
Reserved variable. Return or set the parameters of the command that has been used to launch thhe application. For HTML applications, will return the section of the URL after "?"

Set Input CHAR1, CHAR2
Set the end of line characters detected when you input from a random access file
Parameters:
CHAR1: Ascii value of the first character to detect as new line
CHAR2: Ascii value of the first character to detect as new line

Frame Load CHANNEL, BANK_OR_ADDRESS, LENGTH
Load frames of an IFF channel into memory
Parameters:
CHANNEL: The index of the IFF channel
BANK_OR_ADDRESS: The number of a memory bank or it's addres
LENGTH: The number of frames to load

Frame Play BANK_OR_ADDRESS, NUMBER, SCREEN
Play IFF frameson screen
Parameters:
BANK_OR_ADDRESS: The number of a memory bank or it's addres
NUMBER: The number of frames to play
SCREEN: The index of the screen to play to, or the current screen if not specified

IFF Anim PATH$, _SCREEN_INDEX, NTIME
Play IFF frameson screen
Parameters:

PATH$: The path to the file to play
_SCREEN_INDEX: The index of the screen to create to play the file
NTIME: The number of times to play, once if ommited

Frame Length CHANNEL, NUMBER_OF_FRAMES
Indicates if a file or directory exists on the disc
Parameters:
CHANNEL: The index of the IFF channel
NUMBER_OF_FRAMES: An eventual number of frames to calculate
Value returned:
integer: The length in bytes of the frames in memory. 0 in this version

Frame Skip BANK_OR_ADDRESS, NUMBER_OF_FRAMES
Skip past an animation frame
Parameters:
BANK_OR_ADDRESS: The index of a memory bank or an address in one
NUMBER_OF_FRAMES: An eventual number of frames to skip
Value returned:
integer: The address of the next frame. 0 in this version

Frame Param
Return a parameter after playing a frame
Value returned:
integer: The amount of time needed to successfully display an animation on screen, measured in 50ths of a second.. 0 in this version

Font Commands

Instructions and functions to handle fonts

Get Fonts
Scan the system for new fonts, will scan both Google and Amiga fonts.

Get Disc Fonts
Scan the system for Amiga fonts.

Get Rom Fonts
Scan the system for Google fonts.

Get Font Number TYPE$
Return the number of fonts available to the application
Parameters:
TYPE$: Either "google" or "amiga"
Value returned:
integer: The number of fonts found

Font$ FONT_NUMBER
Return details of available fonts
Parameters:
FONT_NUMBER: The index of the fon in the list of fonts

Set Bitmap Font NUMFONT, CHARWIDTH, CHARHEIGHT, NUMIMAGE, SCALEX, SCALEY
Set a bitmapped font from an image of the images bank
Parameters:
NUMFONT: Number of the bitmap font to set
CHARWIDTH: Width of a char in pixels
CHARHEIGHT: Height of a char in pixels
NUMIMAGE: Number of the image from the images bank
SCALEX: Value of the horizontal scaling of the bitmap font
SCALEY: Value of the vertical scaling of the bitmap font

Set Bitmap Font NUMFONT, CHARWIDTH, CHARHEIGHT, IMAGENAME$, SCALEX, SCALEY
Set a bitmapped font from an image of the images bank
Parameters:
NUMFONT: Number of the bitmap font to set
CHARWIDTH: Width of a char in pixels
CHARHEIGHT: Height of a char in pixels
IMAGENAME$: Image of the image from the images bank
SCALEX: Value of the horizontal scaling of the bitmap font
SCALEY: Value of the vertical scaling of the bitmap font

Set Bitmap Font FONTNAME$, CHARWIDTH, CHARHEIGHT, NUMIMAGE, SCALEX, SCALEY
Set a bitmapped font from an image of the images bank
Parameters:
FONTNAME$: Name of the bitmap font to set

CHARWIDTH: Width of a char in pixels
CHARHEIGHT: Height of a char in pixels
NUMIMAGE: Number of the image from the images bank
SCALEX: Value of the horizontal scaling of the bitmap font
SCALEY: Value of the vertical scaling of the bitmap font

Set Bitmap Font FONTNAME$, CHARWIDTH, CHARHEIGHT, IMAGENAME$, SCALEX, SCALEY
Set a bitmapped font from an image of the images bank
Parameters:
FONTNAME$: Name of the bitmap font to set
CHARWIDTH: Width of a char in pixels
CHARHEIGHT: Height of a char in pixels
IMAGENAME$: Name of the image from the images bank
SCALEX: Value of the horizontal scaling of the bitmap font
SCALEY: Value of the vertical scaling of the bitmap font

Bitmap Text X, Y, FONTNAME$, FONTSTYLE
Draw a Text on the current screen with a bitmapped font
Parameters:
X: Position X of the text in pixels
Y: Position Y of the text in pixels
FONTNAME$: Name of the bitmap font to use
FONTSTYLE: Index of the style of the bitmap font to use

Bitmap Text X, Y, FONTNAME$, FONTSTYLE
Draw a Text on the current screen with a bitmapped font
Parameters:
X: Position X of the text in pixels
Y: Position Y of the text in pixels
FONTNAME$: Name of the bitmap font to use
FONTSTYLE: Index of the style of the bitmap font to use

Game Controllers Commands

Instructions and functions to get input from game controllers

Joy JOYSTICK
Return the current state of the joystick
Parameters:
JOYSTICK: The number of the joystick
Value returned:
integer: A binary number representing where each bit represents a direction. Bit 0: Joystick has been moved Up. Bit 1: Joystick has been moved Down. Bit 2: Joystick has been moved Left. Bit 3: Joystick has been moved Right. Bit 4: Fire-button has been pressedthe direction

Joykey DIRECTION$, KEY$
Description of the instruction.
Parameters:
DIRECTION$: Description of the parameter.
KEY$: Description of the parameter.

Jup PORT, LOCK
Description of the function.
Parameters:
PORT: Description of the parameter.
LOCK: Description of the parameter.
Value returned:
integer: Description of the value returned.

Jdown PORT, LOCK
Description of the function.
Parameters:
PORT: Description of the parameter.
LOCK: Description of the parameter.
Value returned:
integer: Description of the value returned.

Jleft PORT, LOCK
Description of the function.
Parameters:
PORT: Description of the parameter.
LOCK: Description of the parameter.
Value returned:
integer: Description of the value returned.

Jright PORT, LOCK
Description of the function.
Parameters:
PORT: Description of the parameter.
LOCK: Description of the parameter.
Value returned:
integer: Description of the value returned.

Jupleft PORT, LOCK
Description of the function.
Parameters:
PORT: Description of the parameter.
LOCK: Description of the parameter.
Value returned:
integer: Description of the value returned.

Jupright PORT, LOCK
Description of the function.
Parameters:
PORT: Description of the parameter.
LOCK: Description of the parameter.
Value returned:
integer: Description of the value returned.

Jdownleft PORT, LOCK
Description of the function.
Parameters:
PORT: Description of the parameter.
LOCK: Description of the parameter.
Value returned:
integer: Description of the value returned.

Jdownright PORT, LOCK
Description of the function.
Parameters:
PORT: Description of the parameter.
LOCK: Description of the parameter.
Value returned:
integer: Description of the value returned.

Fire PORT, LOCK
Description of the function.
Parameters:
PORT: Description of the parameter.
LOCK: Description of the parameter.
Value returned:
integer: Description of the value returned.

Gamepad Threshold
Description of the function.
Value returned:
integer: Description of the value returned.

Gamepad Threshold T
Description of the instruction.
Parameters:
T: Description of the parameter.

Gamepad Button GAMEPAD, BUTTON
Test for a button depressed on a complex gamepad
Parameters:
GAMEPAD: The number of the gamepad
BUTTON: The number of the button
Value returned:
boolean: True if th ebutton is pressed, False if it is not

Gamepad Buttons GAMEPAD
Description of the function.
Parameters:
GAMEPAD: Description of the parameter.
Value returned:
integer: Description of the value returned.

Gamepad Axes GAMEPAD
Description of the function.
Parameters:
GAMEPAD: Description of the parameter.
Value returned:
integer: Description of the value returned.

Gamepad Axis GAMEPAD, AXIS
Description of the function.
Parameters:
GAMEPAD: Description of the parameter.
AXIS: Description of the parameter.
Value returned:
integer: Description of the value returned.

Gamepad trigger GAMEPAD, TRIGGER
Return the position of one of the triggers of a gamepad
Parameters:
GAMEPAD: The number of the gamepad
TRIGGER: The number of the trigger
Value returned:
float: The position of the trigger, from 0 to 1

Gamepad Name$ GAMEPAD
Return the brand name of the gamepad
Parameters:
GAMEPAD: The number of the gamepad
Value returned:
string: The name of the gamepad if connected, an empty string is no gamepad is connected

Gamepad Vendor$ GAMEPAD
Description of the function.
Parameters:
GAMEPAD: Description of the parameter.
Value returned:
string: Description of the value returned.

Gamepad Product$ GAMEPAD
Description of the function.
Parameters:
GAMEPAD: Description of the parameter.
Value returned:
string: Description of the value returned.

Gamepad Connected GAMEPAD
Test if a gamepad is connected to the computer
Parameters:
GAMEPAD: The number of the gamepad
Value returned:
boolean: True if a gamepad is connected and working, False if not

Globales AOZ Instructions

All instructions globales for the AOZ programming.

instruction NAME$, VARIABLE1, VARIABLE2
Define a user instruction. You can create your own instructions and functions in AOZ, and then use them as
you would normal instructions and functions.
Here's an example of a simple instruction that clears the screen with horizontal BARs of thickness THK, in colour CLR

```
Wipe 10,5  // An example of the instruction usage. Note brackets are not required for instruction parameters.

Instruction "wipe", THK,CLR
 Ink CLR
 for y=0 to 720 step THK
 Bar 0,y,1280,THK
 Wait VBL
 Next y
End Instruction
```

Parameters:
NAME$: the name of your instruction
VARIABLE1: variable carrying a number, float or string required by function.
VARIABLE2: variable carrying a number, float or string required by function.

end instruction
Ends the user instruction definition.

function
Define a user function. You can create your own instructions and functions in AOZ, and then use them as
you would normal instructions and functions. Program the language with the language!
Here's an example function that calculates the distance between two sets if coordinates, x1,y1 and x2,y2.

```
D=Distance(12,20,100,48)    // An example of the function usage. Note brackets are required for function parameters.

Function "Distance", x1, y1, x2, y2
   dist# = sqr((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1))
End Function ( dist# )
```

end function VARIABLE, VARIABLE#, string$
Ends the function definition and returns result in the form of the stated variable, either integer, float integer or string.
Parameters:
VARIABLE: Carries the integer value of the result of the calculation carried out in the function.
VARIABLE#: Carries the float integer value of the result of the calculation carried out in the function.
string$: Carries the charcter string value of the result of the calculation carried out in the function.

mod VARIABLE1, VARIABLE2, VARIABLE1#, VARIABLE2#
Mod - the modulus operator - or more precisely, the modulo operation - is a way to determine the remainder of a division operation.
Instead of returning the result of the division, the modulo operation returns the whole number remainder.

Example> Print 17 mod 5 - answer 2 (divides 3 times with remainder of 2)
Print 4.4 mod 1.1 - answer 0 (divides 4 times exactly)

Parameters:

VARIABLE1: The numerator of the division sum.
VARIABLE2: The denominator of the division sum.
VARIABLE1#: The numerator of the division sum.
VARIABLE2#: The denominator of the division sum.

to CONSTANT1, CONSTANT2

A preposition used in conjunction with other instructions to define a range of values to be used.

Examples> For X=1 to 10
Bob Col (2,A to B)

Parameters:

CONSTANT1: The first value in the range. Can be either a constant, variable or expression.
CONSTANT2: The last value in the range. Can be either a constant, variable or expression.

not

NOT is used to swap over every digit in a binary number from a 0 to a 1, and vice versa.

Example> Print Bin$(Not%11110000,8)

Since -1 (True) can be expressed in binary as %1111111111111111, then NOT TRUE must be equal to FALSE, and a logical NOT operation is achieved.

swap VARIABLE1, VARIABLE2, VARIABLE1#, VARIABLE2#, STRING1$, STRING2$

Use the SWAP command to exchange the data between any two variables of the same type.

Parameters:

VARIABLE1: First of two integer variables being swapped
VARIABLE2: Second of two interger variables being swapped
VARIABLE1#: First of two float variables being swapped
VARIABLE2#: Second of two float variables being swapped
STRING1$: First of two string variables being swapped
STRING2$: Second of two string variables being swapped

def fn

Define a user defined function. To create a user-defined function, give it a name and follow the name with a list of variables.

These variables must be held inside a pair of round brackets, and separated from one another by commas, like these examples:

Examples> Def Fn NAME$(A$)=LOWER$(A$)
Def Fn X(A,B,C)=A*B*C

When a user-defined function is called up the variables that are entered with it will be substituted in the appropriate positions.

fn

Call a user defined function. The following examples show how DEF FN is first used to define a function, and how FN calls it up:

Example> Def Fn NAME$(A$,X,Y)=Mid$(A$,X,Y)
Print Fn NAME$("AOS Studio",4,3)

Example> Def Fn X(A,B,C)=A+B+C
Print Fn X(1,10,100)

The expression that equals the user-defined function can include any of the standard AOZ functions,

and it is limited to a single line of a program.

resume next
After an error, and any error handling instructions, return to the instruction following the one that caused an error.

every on
Turns on regular timed calls to the procedure or sub-routine defined with EVERY instruction.

every off
Turns off regular timed calls to the procedure or sub-routine defined with EVERY instruction.

as
Description of the instruction. ######## REMOVE? ######## Part of other instructions

for VARIABLE, START, END, INCREMENT
FOR is used to create a range values for a specified variable. When used in conjunction with NEXT, a repeating loop is created and any code between the FOR and NEXT instructions will be repeated until the end value is reached. The VARIABLE's initial value will be START when the code is first executed, then it increases by 1 (unless a different step is specified) and repeats. This process is repeated until the VARIABLE's value reaches END and if the value is exactly the same, the code will be executed one last time, if it is any more, the loop will end (see second example below).
This example will print the numbers from 1 to 10 on the screen.:-

```
For x=1 to 10
 Print x
Next x
```

When a STEP is specified, the variable will increase (or decrease if step is negative) by the specified amount each time the loop is repeated.
This example will print the numbers 1,3,5,7 and 9 on the screen. The next number would be 11, but that is out of the range.

```
For x=1 to 10 step 2
 Print x
Next x
```

A FOR/NEXT loop may be left by using EXIT.
Parameters:
VARIABLE: Name of integer variable or float variable.
START: Definition of the starting point of the range defined. Can be a number, integer variable, float variable or expression.
END: Definition of the finishing point of the range defined. Can be a number, integer variable, float variable or expression.
INCREMENT: Definition of the increment step through which the variable range progresses. Can be a number, integer variable, float variable or expression.

next VARIABLE
NEXT is used to mark the end of the code to be repeated in the loop. Can be used with or without the variable defined in FOR statement.
If variable is omitted, the most recently defined loop will be repeated.

A FOR/NEXT loop may be left by using EXIT.
Parameters:
VARIABLE: Variable defined in FOR statement.

repeat

REPEAT is used in conjuction with UNTIL, and defines the start of a loop of code that will be repeated until the condition defined with UNTIL is met. For example:-

```
A=0        // Give variable A a value of 0
Repeat     // Start the loop
 Locate 10,10 : Print A  // Position cursor at 10,10 and print value of variable A
 Inc A     // Increase variable A by 1
Until A=21     //  Repeat loop until variable A=21
End        // End the program
```

A REPEAT/UNTIL loop may be left by using EXIT.

until VARIABLE, VARIABLE#, STRING$

UNTIL is used in conjuction with REPEAT, and defines the end of a block of code that will be repeated until the condition defined with UNTIL is met. For example:-

```
A=0        // Give variable A a value of 0
Repeat     // Start the loop
 Locate 10,10 : Print A  // Position cursor at 10,10 and print value of variable A
 Inc A     // Increase variable A by 1
Until A=21     //  Repeat loop until variable A=21
End        // End the program
```

A REPEAT/UNTIL loop may be left by using EXIT.

Parameters:

VARIABLE: Any integer variable or expression.
VARIABLE#: Any float variable or expression.
STRING$: Any integer variable or expression.

while

WHILE is used in conjunction with WEND, and defines the start of a loop of code that will be repeated until the condition defined with WHILE is met. Once the condition is met, the loop will end and the program will continue from the line after WEND. For example:-

```
A=0        // Give variable A a value of 0
While A<21     // Start the loop and set condition to authorise the code to be executed.
 Locate 10,10 : Print A  // Position cursor at 10,10 and print value of variable A
 Inc A     // Increase variable A by 1
Wend      //  Repeat loop
End        // End the program
```

A WHILE/WEND loop may be left by using EXIT.

wend

WEND is used in conjuction with WHILE, and defines the end of a block of code that will be repeated until the condition defined with WHILE is met. For example:-

```
A=0        // Give variable A a value of 0
While A<21     // Start the loop and set condition to authorise the code to be executed.
 Locate 10,10 : Print A  // Position cursor at 10,10 and print value of variable A
 Inc A     // Increase variable A by 1
Wend      //  Repeat loop
End        // End the program
```

A WHILE/WEND loop may be left by using EXIT.

do
DO is used in conjunction with LOOP, and defines the start of a block of code that will be repeated until the program is given an instruction EXIT. For example:-

```
A=0        // Give variable A a value of 0
Do         // Start the loop.
 Locate 10,10 : Print A  // Position cursor at 10,10 and print value of variable A
 Inc A      // Increase variable A by 1
 If A=101 then Exit   // Check if A has reached 101, and EXIT the loop if it has
Loop       //  Repeat loop
End        // End the program
```

A DO/LOOP loop may be left by using EXIT.

loop
Description of the instruction.

exit if NUMBER
EXIT IF forces the program to leave a loop if the condition in the EXPRESSION is met, and it can be used to escape from all the types of loop employed in AOZ, such as FOR ... NEXT, REPEAT ... UNTIL, WHILE ... WEND and DO ... LOOP.
Any number of loops may be nested inside of one another, and when used on its own, EXIT If will leave the innermost loop only. By including an optional number after the EXPRESSION, that number of nested loops will be taken into
account before the EXIT is made, and the program will jump directly to the instruction immediately after the relevant loop. Try the following example. If 2 is entered, the program will display "Left Middle Loop", as 2 loops are EXITed.

```
Do
 Do
  Do
   Input "Type in a number";X
   Exit If X=1
   Exit If X=2,2
   Exit If X=3,3
  Loop       // 1st loop exited
  Print "Left the Inner Loop"
 Loop        // 2nd loop exited
 print "Left the Middle Loop"
Loop         // 3rd loop exited
Print "Left the Outer Loop"
End
```

Parameters:
NUMBER: The number of loops to be EXITed.

exit NUMBER
EXIT forces the program to leave a loop immediately, and it can be used to escape from all the types of loop employed in AOZ, such as FOR ... NEXT, REPEAT ... UNTIL, WHILE ... WEND and DO ... LOOP.
Any number of loops may be nested inside of one another, and when used on its own, EXIT will leave the innermost loop only. By including an optional number after EXIT, that number of nested loops will be taken into account before the EXIT is made, and the program will jump directly to the instruction immediately after the

relevant loop. Try the following example. If 2 is entered, the program will display "Left Middle Loop", as 2 loops are EXITed.

```
Do
 Do
  Do
   Input "Type in a number";X
   If X=1 Then Exit
   If X=2 Then Exit 2
   If X=3 then Exit 3
  Loop        // 1st loop exited
  Print "Left the Inner Loop"
 Loop         // 2nd loop exited
 print "Left the Middle Loop"
Loop          // 3rd loop exited
Print "Left the Outer Loop"
End
```

Parameters:

NUMBER: The number of loops to be EXITed.

goto LABEL, LABEL$

GOTO forces the program to jump to a specified destination. In AOZ, these destinations can be a unique line number, a label, or an expression.
Label markers can consist of names that use any string of letters or numbers, as well as the underscore character "_", and they must be ended with the colon character ":".

Examples of label markers: 12 // Number only labels can be written like this
12: // or like this, with a colon ":"
Bullet: // Named labels must have a colon
Shoot_Bullet1: // and can be anything you like, so long as it doesn't conflict with a variable name.

And corresponding GOTO calls: Goto 12 //
Goto Bullet // Note the colon is not required
Goto Shoot_Bullet1 //

Using an expression is a little more complicated. A named label may be made up from string parts, like this:

```
Goto "Shoot_Bullet"+"1"
```

Or an expression containing a variable and an integer like this:-

```
A=10
Goto A+2  // goto line number 12
```

Note: You cannot use a variable name alone, such as 'A' above, to call line 10. This will generate an error because AOZ doesn't know whether you want to call line 10 or label 'A'

Parameters:

LABEL: The number of the line to be jumped to.
LABEL$: The name of the label to be jumped to.

gosub LABEL

GOSUB is used to call a sub-routine (a part of the program that performs a specific task) at a specified destination. In AOZ, these destinations can be a unique line number, a label, or an expression. Once the sub-routine has been

executed, the program returns to the line following the GOSUB call.
Label markers can consist of names that use any string of letters or numbers, as well as the underscore character "_", and they must be ended with the colon character ":".

Examples of label markers: 12   // Number only labels can be written like this
    12:     // or like this, with a colon ":"
    Bullet: // Named labels must have a colon
    Shoot_Bullet1: // and can be anything you like, so long as it doesn't conflict with a variable name.

And corresponding GOSUB calls: Gosub 12   //
    Gosub Bullet   // Note the colon is not required
    Gosub Shoot_Bullet1 //

Using an expression is a little more complicated. A named label may be made up from string parts, like this:

```
Gosub "Shoot_Bullet"+"1"
```

Or an expression containing a variable and an integer like this:-

```
A=10
Gosub A+2   // Call Sub-routine at line number 12
Print "Completed"
End
12:
 Print "This is the sub-routine call"
Return
```

Note: You cannot use a variable name alone, such as 'A' above to call line 10. This will generate an error because AOZ doesn't know whether you want to call line 10 or label 'A'.

if
Description of the instruction.

then
Description of the instruction.

else if
Description of the instruction.

else
Description of the instruction.

end if
Description of the instruction.

on error
Description of the instruction.

on break proc
Description of the instruction.

on menu
Description of the instruction.

on
Description of the instruction.

resume label
Description of the instruction.

resume
Description of the instruction.

pop proc
Description of the instruction.

every
Description of the instruction.

step
Description of the instruction.

return
Description of the instruction.

pop
Normally you cannot exit from a GOSUB statement using a standard GOTO, and this may be inconvenient. For example, there could be an error that makes it unacceptable to return to the program exactly where you left it. In such circumstances, the POP command can be used to remove the return address generated by a GOSUB, allowing you to leave the sub-routine without the final RETURN statement being executed. For example:

```
Do
 Gosub THERE
Loop
HERE:
Print "I've just popped out!"
End
THERE:
Print "Hello There!"
If Mouse Key Then Pop : Goto HERE
Return
```

procedure
Description of the instruction.

proc
Description of the instruction.

end proc
Description of the instruction.

shared
Description of the instruction.

global
Description of the instruction.

param#
Description of the instruction.

param$
Description of the instruction.

param
Description of the instruction.

error
Description of the instruction.

errn
Description of the instruction.

data
Description of the instruction.

read
Description of the instruction.

restore
Description of the instruction.

print using
Description of the instruction.

print
Description of the instruction.

lprint
Description of the instruction.

input
Description of the instruction.

line input
Description of the instruction.

mid$
Description of the instruction.

left$ STRING2, STRING3, STRING4, VARIABLE
LEFT$ is used to return or define a number of the leftmost characters in a string.

Parameters:

STRING2: The source string of characters that will be returned.

STRING3: The source string of characters that will be changed.

STRING4: The string of characters that will replace the leftmost VARIABLE characters in STRING3.

VARIABLE: The number of characters to be changed or returned, either an integer, variable or expression.

Value returned:

STRING1: string:The leftmost VARIABLE characters of string STRING2

right$ STRING2, STRING3, STRING4, VARIABLE

RIGHT$ is used to return or define a number of the rightmost characters in a string.

Parameters:

STRING2: The source string of characters that will be returned.

STRING3: The source string of characters that will be changed.

STRING4: The string of characters that will replace the rightmost VARIABLE characters in STRING3.

VARIABLE: The number of characters to be changed or returned, either an integer, variable or expression.

Value returned:

STRING1: string:The rightmost VARIABLE characters of string STRING2

dim

Description of the instruction.

rem

Description of the instruction.

sort

Description of the instruction.

match

Description of the instruction.

edit

Description of the instruction.

direct

Description of the instruction.

default palette

Description of the instruction.

palette

Description of the instruction.

colour back

Description of the instruction.

timer

Description of the instruction.

picture

Description of the instruction.

polyline
Description of the instruction.

polygon
Description of the instruction.

trap
Description of the instruction.

include
Description of the instruction.

channel
Description of the instruction.

amreg
Description of the instruction.

rain
Description of the instruction.

field
Description of the instruction.

dir$
Description of the instruction.

set transparent COLOUR
Sets the colour identified by number COLOUR to transparent. Normally, colour 0 is transparent,
Parameters:
COLOUR: the palette number of the colour to be set to transparent.

stop transparent
Switches off any transparent colours.

set alpha
Description of the instruction.

inc VARIABLE
INC increases the value of any integer or float variable by 1.
Parameters:
VARIABLE: Any integer or float variable.

dec VARIABLE
DEC decreases the value of any integer or float variable by 1.
Parameters:
VARIABLE: Any integer or float variable.

add VARIABLE1, VARIABLE2, VARIABLE3, VARIABLE4

##sort!!##

Parameters:

VARIABLE1: Any integer or float variable that you want to ADD to.
VARIABLE2: Any integer or float variable or expression that will be ADDed to VARIABLE1.
VARIABLE3: Any integer or float variable or expression that defines the start of the range for VARIABLE1.
VARIABLE4: Any integer or float variable or expression that defines the end of the range for VARIABLE1.

end

END stops the program from running and a flashing message is displayed saying "Program ended successfully".

wait vbl

WAIT VBL tells the program to pause for a specific amount of time. This time is directly linked to the refresh rate of your display, so at 60hz, the time is approximately 0.017 seconds, or 1/60th.
It is essential that you have a WAIT instruction in the main loop of your program, otherwise it will run incredibly quickly and take up a huge amount processor time. A WAIT instruction is also essential to allow the display to be rendered and updated.
WARNING: If you use WAIT VBL for timing purposes, it's worth noting that a program that runs at the correct speed on a 60hz display, will run 2.4 times faster on a 144hz display. For games, it's better to use the WAIT instruction.

wait key

WAIT KEY tells the program to pause until a key is pressed.

wait NUMBER

WAIT tells the program to pause for a specified length of time in seconds defined by NUMBER.
AOZ can accept values in 1/1000ths of a second, for example:

```
Wait 0.015   // Wait for 15/1000ths of a second
```

It is essential that you have a WAIT instruction in the main loop of your program, otherwise it will run incredibly quickly and take up a huge amount processor time. A WAIT instruction is also essential to allow the display to be rendered and updated.
Note: Interupt driven processes such as EVERY and TOUCH ON CHANGE will still be carried out if required during a WAIT instruction.

Parameters:

NUMBER: Number of seconds to wait.

stop

STOP stops the program from running and a flashing message is displayed saying "Program ended successfully".

error NUMBER

Generates a deliberate error so error handling can be tested.

Parameters:

NUMBER: Number of error to be generated.

break off

Using Break Off prevents a program from being stopped by pressing CTRL C.

break on

When Break On is used, a program can be stopped by pressing CTRL C.

areg
Description of the instruction.

dreg
Description of the instruction.

vdialog CHANNEL, VARIABLE, VALUE
VDIALOG is a function that can assign or read an Interface value. This function can be used to either read or change the variables in any active Interface program. The active channel number is selected, followed by the number of the variable you are interested in. The value refers to the new integer value that has been selected for this variable.
Parameters:
CHANNEL: The number of the channel where variable number VARIABLE is to be assigned integer VALUE
VARIABLE: The number of the variable on channel CHANNEL which is to be assigned integer VALUE
VALUE: Integer value to be assigned to variable number VARIABLE on channel number CHANNEL
Value returned:
VALUE: integer:The value of variable VARIABLE on channel CHANNEL

vdialog$ CHANNEL, VARIABLE, STRING$
VDIALOG$ is a function that can assign or read an Interface string
The VDIALOG$ function is exactly the same as the Vdialog function, except that it works with strings rather than numbers. Specify the channel number and the variable number, and then string$ holds a new string to be stored in the Interface variable array.
Parameters:
CHANNEL: The number of the channel where variable number VARIABLE is to be assigned string STRING$
VARIABLE: The number of the variable on channel CHANNEL which is to be assigned string STRING$
STRING$: String value to be assigned to variable number VARIABLE on channel number CHANNEL
Value returned:
STRING$: string:The string value of string variable VARIABLE on channel CHANNEL

set font number, name$, size
Sets the font to be used by Text operations to the font corresponding to the number or name defined at the point size stated.
Parameters:
number: the number of the font given to it by AOZ as it is loaded into memory. The default font is numbered 0, then any you load are numbered 1,2 and so on.
name$: the name of the font in lower case.
size: the point size of the font.

input$ variable$, number
Request a specified number of characters to be typed. Typed characters will be added to the string variable 'variable$' in turn until the number of characters specified by 'number' is reached. Note - typed characters are not visible and cannot be made
visible whilst being typed. Example:-

```
Print "Type 10 Characters"
v$=input$(10)
Print "You typed > ";v$
```

Parameters:
variable$: Name of sting variable that will contain the specified number of characters typed.
number: Number, variable or expression that provides the number of characters to be typed.

key$
Description of the instruction.

x mouse
The x mouse function reports the calculated x coordinate of the mouse pointer within the AOZ screen.
The coordinate given represents the pixel position of the mouse within the AOZ screen.
If the mouse pointer leaves the AOZ screen, the last position of the mouse within the AOZ screen will be reported.

y mouse
The y mouse function reports the calculated y coordinate of the mouse pointer within the AOZ screen.
The coordinate given represents the pixel position of the mouse within the AOZ screen.
If the mouse pointer leaves the AOZ screen, the last position of the mouse within the AOZ screen will be reported.

bset number, value
The bset function (abbreviated from Bit Set) sets the state of a single binary bit in a given value to 1.
Specify the number of the bit to be changed, from 0 to 63, then give the chosen variable or expression.
If the bit is currently a 0, it will change to 1, if 1 it will remain unchanged. For example:

```
B=11
bset 2,B
Print B
```

Parameters:
number: number of bit to set
value: variable or expression

bclr number, value
The bclr function (abbreviated from Bit Clear) sets the state of a single binary bit in a given value to 0.
Specify the number of the bit to be changed, from 0 to 63,then give the chosen variable or expression.
If the bit is currently a 1, it will change to 0, if 0 it will remain unchanged. For example:

```
B=15
bclr 2,B
Print B
```

Parameters:
number: number of bit to clear
value: variable or expression

bchg number, value
The bchg function (abbreviated from Bit Change) changes the state of a single binary bit in a given value.
Specify the number of the bit to be changed, from 0 to 63, then give the chosen variable or expression.
If the bit is currently a 1, it will change to 0, 0 will change to 1. For example:

```
B=15
bchg 3,B
Print B
```

Parameters:
number: number of bit to change
value: Variable or expression

btst number, value
The btst function (abbreviated from Bit Test) tests a single binary bit in a given value.
Specify the number of the bit to be tested, from 0 to 63, then give the chosen variable or expression.
If the test is successful, a value of 1 (True) is returned, otherwise a zero (False) is given. For example:

```
B=%1010
Print Btst(3,B)
Print Btst(2,B)
```
Parameters:

number: Number of bit to test
value: Variable or expression.

ror.b number, bin value

Ror.b is the AOZ version of the Ror.b command used in the 68000 assembly language. It takes binary value of the given number, expression or variable, and rotates the first 8 bits of the specified number of places to the right.

Ror.b can be used as a fast way of dividing any positive number by a power of two, like this:

```
A=8
Ror.b 1,A
Print A
```
Parameters:

number: The number of places to be moved
bin value: Any number, expression or variable

ror.w number, bin value

Ror.w is the AOZ version of the Ror.w command used in the 68000 assembly language. It takes binary value of the given number, expression or variable, and rotates the first 16 bits of the specified number of places to the right.

Ror can be used as a fast way of dividing any positive number by a power of two, like this:

```
A=1024
Ror.w 1,A
Print A
```
Parameters:

number: The number of places to be moved
bin value: Any number, expression or variable

ror.l number, bin value

Ror.l is the AOZ version of the Ror.l command used in the 68000 assembly language. It takes binary value of the given number, expression or variable, and rotates the all the bits of the specified number of places to the right.

Ror can be used as a fast way of dividing any positive number by a power of two, like this:

```
A=1024
Ror.l 1,A
Print A
```
Parameters:

number: The number of places to be moved
bin value: Any number, expression or variable

rol.b number, bin value

Rol.b is the AOZ version of the Rol.b command used in the 68000 assembly language. It takes binary value of the given number, expression or variable, and rotates the first 8 bits of the specified number of places to the left.

Rol.b can be used as a fast way of multiplying any positive number by a power of two, like this:

```
A=8
Rol.b 1,A
Print A
```
Parameters:

number: The number of places to be moved
bin value: Any number, expression or variable

rol.w number, bin value

Rol.w is the AOZ version of the Rol.w command used in the 68000 assembly language. It takes binary value of the given number, expression or variable, and rotates the first 16 bits of the specified number of places to the right.

Rol.w can be used as a fast way of multiplying any positive number by a power of two, like this:

```
A=1024
Rol.w 1,A
Print A
```

Parameters:

number: The number of places to be moved
bin value: Any number, expression or variable

rol.l number, bin value

Rol.l is the AOZ version of the Rol.l command used in the 68000 assembly language. It takes binary value of the given number, expression or variable, and rotates the all the bits of the specified number of places to the right.

Rol.l can be used as a fast way of multiplying any positive number by a power of two, like this:

```
A=1024
Rol.l 1,A
Print A
```

Parameters:

number: The number of places to be moved
bin value: Any number, expression or variable

Graphical Commands

Instructions and functions to draw graphics in AOZ screens

Cls INK, X1, Y2, X2, Y2
Clear an area of the current screen
Parameters:
INK: The index of the color in the palette to clear with
X1: The horizontal coordinate of the top-left pixel of the rectangle to clear
Y2: The vertical coordinate of the top-left pixel of the rectangle to clear
X2: The horizontal coordinate of the bottom-right pixel of the rectangle to clear
Y2: The vertical coordinate of the bottom-right pixel of the rectangle to clear

Cls INK, X, Y, WIDTH, HEIGHT
Clear an area of the current screen
Parameters:
INK: The index of the color in the palette to clear with (optional, if not specified will use the current INK index)
X: The horizontal coordinate of the top-left pixel of the rectangle to clear
Y: The vertical coordinate of the top-left pixel of the rectangle to clear
WIDTH: The width in pixels of the rectangle to clear
HEIGHT: The height in pixels of the rectangle to clear

Cls INK
Clear the current screen
Parameters:
INK: The index of the color in the palette to clear with (optional, if not specified will use the current INK index)

Plot X, Y, INK
Draw a pixel in the current screen
Parameters:
X: The horizontal coordinate of the point to draw
Y: The vertical coordinate of the point to draw
INK: The index of the color in the screen palette (optional, will use the latest Ink value if ommited)

Point X, Y
Return the palette index of a point within the current screen (warning: may return wrong result in PC modedue to anti-aliasing)
Parameters:
X: The horizontal coordinate of the point to test
Y: The vertical coordinate of the point to test
Value returned:
INDEX: integer:The index of the color in the screen palette

Draw X, Y
Draw a line with the current Ink from the last graphical position
Parameters:
X: The horizontal coordinate of the point to test
Y: The vertical coordinate of the point to test

Draw X1, Y1, X2, Y2
Draw a line with the current Ink between two points in the current screen
Parameters:
X1: The horizontal coordinate of the first point
Y1: The vertical coordinate of the first point
X2: The horizontal coordinate of the second point
Y2: The vertical coordinate of the second point

Draw X, Y, WIDTH, HEIGHT
Draw a line with the current Ink from the top-left to bottom-right corners of an area in the curretn screen
Parameters:
X: The horizontal coordinate of the first point
Y: The vertical coordinate of the first point
WIDTH: The width of the area in pixels
HEIGHT: The height of the area in pixels

Ellipse X, Y, XRADIUS, YRADIUS
Draw an ellipse with the current Ink in the current screen
Parameters:
X: The horizontal coordinate of the centre of the ellipse
Y: The vertical coordinate of the centre of the ellipse
XRADIUS: The horizontal radius in pixels
YRADIUS: The vertical radius in pixels

Circle X, Y, RADIUS
Draw a circle with the current Ink in the current screen
Parameters:
X: The horizontal coordinate of the centre of the circle
Y: The vertical coordinate of the centre of the circle
RADIUS: The radius of the circle in pixels

Bar X1, Y1, X2, Y2
Draw a filled rectangle with the current Ink, Pattern in the current screen
Parameters:
X1: The horizontal coordinate of the top-left corner of the rectangle
Y1: The vertical coordinate of the top-left corner of the rectangle
X2: The horizontal coordinate of the bottom-right corner of the rectangle
Y2: The vertical coordinate of the bottom-right corner of the rectangle

Bar X, Y, WIDTH, HEIGHT
Draw a filled rectangle with the current Ink, Pattern in the current screen
Parameters:
X: The horizontal coordinate of the top-left corner of the rectangle
Y: The vertical coordinate of the top-left corner of the rectangle
WIDTH: The width of the rectangle to draw in pixels
HEIGHT: The height of the rectangle to draw in pixels

Box X1, Y1, X2, Y2
Draw an empty rectangle with the current Ink, Line width and Line Pattern in the current screen
Parameters:
X1: The horizontal coordinate of the top-left corner of the rectangle
Y1: The vertical coordinate of the top-left corner of the rectangle

X2: The horizontal coordinate of the bottom-right corner of the rectangle
Y2: The vertical coordinate of the bottom-right corner of the rectangle

Box X, Y, WIDTH, HEIGHT
Draw an empty rectangle with the current Ink, Line width and Line Pattern in the current screen
Parameters:
X: The horizontal coordinate of the top-left corner of the rectangle
Y: The vertical coordinate of the top-left corner of the rectangle
WIDTH: The width of the rectangle to draw in pixels
HEIGHT: The height of the rectangle to draw in pixels

Paint X, Y, COLOUR
TODO! Performs a flood-paint of a closed area in the current screen
Parameters:
X: The horizontal coordinate where to start the paint process
Y: The horizontal coordinate where to start the paint process
COLOUR: The indxex in the palette of the screen of the color to use (optional, will use the current Ink index if ommited)

Polygon X1, Y1, WIDTH1, HEIGHT1, WIDTH2, HEIGHT2
Draw a closed empty shape using the current Ink and line parameters for the outline and the current fill parameters for the inside. The last point is joined to the first point of the list
Parameters:
X1: Horizontal coordinate of the first point
Y1: Vertical coordinate of the first point
WIDTH1: Horizontal signed displacement from the first point to the second
HEIGHT1: Vertical signed displacement from the first point to the second
WIDTH2: Horizontal displacement from the second point to the third, etc.
HEIGHT2: Vertical displacement from the second point to the third, etc.

Polyline X1, Y1 [, WIDTH, HEIGHT]
Description of the instruction.
Parameters:
X1: Description of the parameter.
Y1 [: Description of the parameter.
WIDTH: Description of the parameter.
HEIGHT]: Description of the parameter.

Polygon X1, Y1 [, X2, Y2]
Description of the instruction.
Parameters:
X1: Description of the parameter.
Y1 [: Description of the parameter.
X2: Description of the parameter.
Y2]: Description of the parameter.

Polygon X1, Y1 [, WIDTH, HEIGHT]
Description of the instruction.
Parameters:
X1: Description of the parameter.
Y1 [: Description of the parameter.
WIDTH: Description of the parameter.

HEIGHT]: Description of the parameter.

Gr Locate X, Y

Position the graphical cursor in the current screen. Next instructions like "Draw To" (or any graphical instruction wheree the X and Y coordinates are ommited) will starrt from this position

Parameters:

X: The horizontal coordinate in pixels

Y: The horizontal coordinate in pixels

Text Length TEST$

Calculate the width in pixels of a string when draw inthe current screen with the current selected font and font size

Parameters:

TEST$: The text to evaluate the width from

Value returned:

integer: The width in pixels of the text on screen using the current font and font-size

Text Base

Return the position of the base line of a graphical text displayed on the current screen with the current font attributes name and size

Value returned:

integer: The position of the base line of the text calculated from the top of the characters

Text X, Y, TEXT$, TAGS$

Draw a graphical text on th current screen using the current font and font size

Parameters:

X: The horizontal position of the text in pixels

Y: The vertical position of the text in pixels

TEXT$: The text to draw

TAGS$: A string containing tags facilitating the display of the text. Tags can be any logical combination of the following tags and have the same meaning as their Javascritp equivalents: '#left', '#center', '#right', '#start' or '#end' for horizontal alignement, '#top', '#hanging', '#middle', '#alphabetic', '#ideographic' or '#bottom' for vertical alignement, '#ltr', '#rtl' or '#inherit' for drawing direction

Set Paint ONOFF

Set the painting mode for filled drawing forms like "Polygon" and "Bar"

Parameters:

ONOFF: True to fill the area with the current paitn pattern and inks, False to leave them empty and only draw the borders

Set Pattern PATTERN

Set the filled area painting pattern

Parameters:

PATTERN: If positive, a number from 0 to 34 indicate the default pattern to use, if negative, the inverse value represents the index of an image in the Images bank to use as drwaing pattern

Set Line PATTERN

Set the patterm of the future lines drawn by the "Draw", "Draw To", "Box" etc. instructions

Parameters:

PATTERN: A binary mask of bits where each bit set to one will be visible and invisible if zero, repeated along the line

Ink INDEX, PATTERN, NUMBER

Set the index of the colour from the palette of the current screen to use in all future graphical operations
Parameters:
INDEX: The index of the colour in the palette to use
PATTERN: An optional pattern to use
NUMBER: TODO! find what this parameter is cannot remember and AMOS manual is confusing

Ink
Function that returns the graphic ink color index.
Value returned:
:

Gr Writing STYLE
Set all graphical operations drawing mode. Warning work in progress incomplete support TODO!
Parameters:
STYLE: A set of bits indicating the process. Bit 0 = 0 only draw graphics using the current ink colour, Bit 0 = 1 replace any existing graphics with new graphics (default condition), Bit 1 = 1 change old graphics that overlap with new graphics, using XOR, Bit 2 = 1 reverse ink and paper colours, creating inverse video effect

Clip X1, Y1, X2, Y2
Clip all further graphical operations in the current screen to a rectangle
Parameters:
X1: The horizontal coordinate of the top-left corner of the clipping rectangle
Y1: The vertical coordinate of the top-left corner of the clipping rectangle
X2: The horizontal coordinate of the bottom-right corner of the clipping rectangle
Y2: The vertical coordinate of the bottom-right corner of the clipping rectangle

Clip X, Y, WIDTH, HEIGHT
Clip all further graphical operations in the current screen to a rectangle
Parameters:
X: The horizontal coordinate of the top-left corner of the clipping rectangle
Y: The vertical coordinate of the top-left corner of the clipping rectangle
WIDTH: The width of the clipping rectangle
HEIGHT: The height of the clipping rectangle

Clip
Removes the clipping rectangle and allow drawing on the whole surface of the screen

Set Tempras
Deprecated, has no effect

HTML manipulation commands

Put any DOM elements in your AOZ program to create your web application

AOZ Hide
Hide the AOZ display canvas

AOZ Show
Show the AOZ display canvas

Dom Element ELEMENTTYPE$, ELEMENTID$
Create a HTML Element
Parameters:
ELEMENTTYPE$: Name of the DOM element to create. May be "img", "div", "canvas", "textarea", "button"...
ELEMENTID$: ID of this element.

Dom Attribute ELEMENTID$, ATTRIBUTE$, VALUE$
Set an attribute of a HTML Element
Parameters:
ELEMENTID$: ID of this element.
ATTRIBUTE$: Name of the attribute. May be "class", "src", "style"...
VALUE$: Value of the attribute to assign

Dom Attribute ELEMENTID$, ATTRIBUTE$
Get the value of a attribute of a HTML Element
Parameters:
ELEMENTID$: ID of this element.
ATTRIBUTE$: Name of the attribute. May be "class", "src", "style"...
Value returned:
string: The value of the attribute

Dom Add CHILDID$, PARENTID$
Add an HTML element create with "Dom Create" command into an other HTML Element
Parameters:
CHILDID$: ID of the element to add.
PARENTID$: ID of the HTML element container.

Dom Content ELEMENTID$, HTML$
Set the content of a HTML Element
Parameters:
ELEMENTID$: ID of this element.
HTML$: HTML code to insert into the HTML element.

Dom Content ELEMENTID$
Return the code HTML contains in a HTML Element
Parameters:
ELEMENTID$: ID of this element.
Value returned:
string: The HTML code into the HTML element.

Dom Value ELEMENTID$, VALUE$
If the HTML Element is a element of form (input, list, radio, option...), you can to set the value of this element (text, selected...)
Parameters:
ELEMENTID$: ID of this element.
VALUE$: Value to assign at this element.

Dom Value ELEMENTID$
If the HTML Element is a element of form (input, list, radio, option...), return this value (text, selected...)
Parameters:
ELEMENTID$: ID of this element.
Value returned:
string: Value of this element.

Dom Event ELEMENTID$, EVENTNAME$, PROCNAME$
Link an Event Listener on a HTML Element with a AOZ Procedure
Parameters:
ELEMENTID$: ID of this element.
EVENTNAME$: Name of the event ("click", "keydown", "mousedown", "mousemove"...).
PROCNAME$: Name of the AOZ procedure receiving the result of the event.

Load CSS URL$
Load a file CSS to add styles rules
Parameters:
URL$: Name of file CSS to load.

Load CSS URL$, MEDIA$
Load a file CSS to add styles rules for a specific media ("screen", "print"...)
Parameters:
URL$: Name of the CSS file to load.
MEDIA$: Name of the media.

Add CSS Rule CSSRULE$
Add a CSS Rule directly in AOZ code
Parameters:
CSSRULE$: CSS Code of the rule.

Load JS URL$
Load a Javascript file
Parameters:
URL$: Url of the Javascript file to load.

Dom Visible ELEMENTID$, VISIBLE
Show/Hide a HTML Element
Parameters:
ELEMENTID$: ID of this element.
VISIBLE: State of the visibility (true or false).

Dom Position ELEMENTID$, X, Y
Set the Position of a HTML element

Parameters:

ELEMENTID$: ID of this element.

X: Horizontal position in pixel of the HTML element.

Y: Horizontal position in pixel of the HTML element.

Dom Position ELEMENTID$, X, Y, UNITMEASURE$

Set the Position of a HTML element with a specific measeure unit ('px', 'pt', 'em', '%'...)

Parameters:

ELEMENTID$: ID of this element.

X: Horizontal position of the HTML element.

Y: Vertical position of the HTML element.

UNITMEASURE$: Code of the unit measure.

Dom Size ELEMENTID$, WIDTH, HEIGHT

Resize a HTML element

Parameters:

ELEMENTID$: ID of this element.

WIDTH: Width of the HTML element in pixel.

HEIGHT: Height of the HTML element in pixel.

Dom Size ELEMENTID$, WIDTH, HEIGHT, UNITMEASURE$

Resize a HTML element with a specific measeure unit ('px', 'pt', 'em', '%'...)

Parameters:

ELEMENTID$: ID of this element.

WIDTH: Width of the HTML element.

HEIGHT: Height of the HTML element.

UNITMEASURE$: Code of the unit measure.

Dom Enabled ELEMENTID$, ENABLED

Show/Hide a HTML Element

Parameters:

ELEMENTID$: ID of this element.

ENABLED: State of the element (true or false).

Dom Button ELEMENTID$, LABEL$

Create a HTML Button

Parameters:

ELEMENTID$: ID of this element.

LABEL$: Label of the button.

Dom Button ELEMENTID$, LABEL$, CLASSNAME$

Create a HTML Button with CSS Rules

Parameters:

ELEMENTID$: ID of this element.

LABEL$: Label of the button.

CLASSNAME$: Names of CSS rules to this element

Dom Layer ELEMENTID$

Create a HTML Layer (div)

Parameters:

ELEMENTID$: ID of this element.

Dom Button ELEMENTID$, CLASSNAME$
Create a HTML Button with CSS Rules
Parameters:
ELEMENTID$: ID of this element.
CLASSNAME$: Names of CSS rules to this element

Js Execute JAVASCRIPT$
Description of the instruction.
Parameters:
JAVASCRIPT$: Description of the parameter.

Information Commands

Functions for returning application and system information.

Manifest$
Function that returns the name of the selected manifest platform.
Value returned:
:

Browser Name$
Function that returns the simple name of the browser in which an AOZ application is running.
Value returned:
:

Browser Language$
Function that returns the language used by the client browser.
Value returned:
:

Browser Version$
Description of the function.
Value returned:
string: Description of the value returned.

Browser Agent$
Description of the function.
Value returned:
string: Description of the value returned.

Browser Engine$
Description of the function.
Value returned:
string: Description of the value returned.

Browser Ancestor$
Description of the function.
Value returned:
string: Description of the value returned.

Browser Codename$
Description of the function.
Value returned:
string: Description of the value returned.

Browser Online
Description of the function.
Value returned:
integer: Description of the value returned.

Browser Platform$
Description of the function.
Value returned:
string: Description of the value returned.

Geo Latitude
Description of the function.
Value returned:
integer: Description of the value returned.

Geo Longitude
Description of the function.
Value returned:
integer: Description of the value returned.

AMAL Commands

Instructions and functions to animate and move sprites, bobs, screens, rainbows or any other graphical objects

Chanan INDEX
Test an AMAL channel for an active animation
Parameters:
INDEX: Index of the channel
Value returned:
boolean: string:True if the animation is actice, false if not

Chanmv INDEX
Test an AMAL channel for an active movement
Parameters:
INDEX: Index of the channel
Value returned:
boolean: True if the movement is actice, false if not

Amal On INDEX
Start one or all AMAL channels
Parameters:
INDEX: Index of the channel (optional)

Amal Off INDEX
Stop one or all AMAL channels
Parameters:
INDEX: Index of the channel (optional)

Amal Freeze INDEX
Pause one or all AMAL channels
Parameters:
INDEX: Index of the channel (optional)

Amalerr$
Return the name of an eventual error in AMAL string when running it and advances the pointer to the next error
Value returned:
string: The text of the error

Amalerr
Return the position of an eventual error in AMAL string when running it and advances the pointer to the next error
Value returned:
string: The position of the error in the string

Amal INDEX, AMAL$
TOTEST! Assigns an AMAL program to an animation channel
Parameters:
INDEX: Index of the animation channel
AMAL$: A string containing the AMAL program to be ran

Amal INDEX, NUMBER
TOTEST! Assigns an AMAL program to an animation channel
Parameters:
INDEX: Index of the animation channel
NUMBER: The number of the AMAL program in the AMAL bank

Amal INDEX, AMAL$, ADDRESS
TODO! Assigns an AMAL program to an animation channel and direct the output to a memory address
Parameters:
INDEX: Index of the animation channel
AMAL$: A string containing the AMAL program to execute
ADDRESS: The memeory address where to redirect the output

Synchro On
Turn ON the automatic execution of the AMAL programs

Synchro Off
Turn OFF the automatic execution of the AMAL programs

Synchro
Executes on step of animation of all AMAL channels. "Synchro Off" must have been used before calling this instruction

Amiga-specific Commands

Instructions and functions related specifically to the Amiga. Implementation will be done as best as possible, yet, not all will be possible to emulate. Most of the instruction will compile, yet have simply have no effect in runtime.

Set Hardcol BITMAP1, BITMAP2
Set hardware register for hardware Sprite collision detection. No effect in AOZ.
Parameters:
BITMAP1: ...
BITMAP2: ...

Set Hardcol
Return collision status after a Set Hardcol instruction. No effect in AOZ.
Value returned:
integer: 0

Copper On
Re-start automatic copper generation. May be implemented in a future enhanced Amiga renderer.

Copper Off
Stop automatic copper generation. May be implemented in a future enhanced Amiga renderer.

Cop Swap
Swap logical and physical copper lists. May be implemented in a future enhanced Amiga renderer.

Cop Reset
Re-set copper list pointer. May be implemented in a future enhanced Amiga renderer.

Cop Wait
Insert a WAIT instruction into copper list. May be implemented in a future enhanced Amiga renderer.

Cop Movel
Write a long MOVE instruction to copper list. May be implemented in a future enhanced Amiga renderer.

Cop Move
Write a MOVE instruction to current copper list. May be implemented in a future enhanced Amiga renderer.

Cop Logic
Give address of logical copper list. May be implemented in a future enhanced Amiga renderer.
Value returned:
integer: 0

PSel$
Used to handle multiple applications in the AMOS IDE. Will not be implemented.
Value returned:
string: ""

Multi Wait
managed multi-tasking on the Amiga. May be implemented.

AMOS To Front
Bring AMOS IDE in the front of display. Will not be implemented.

AMOS To Back
Bring AMOS IDE in the back of display. Will not be implemented.

AMOS Here
Detect if AMOS is running. Emulation returns TRUE
Value returned:
boolean: True

AMOS Lock
Blocks Amiga-A key. Will not be implemented.

Close Workbench
Cclose the Workbench. Will not be implemented.

Set Buffer
Set the size of the variable area. Will not be implemented.

Equ
Get an equate. Will not be implemented.
Value returned:
string: ""

Lvo
Get the Library Vector Offset. Will not be implemented.
Value returned:
integer: 0

Set Double Precision
Engage double precision accuracy. Will not be implemented.

Request WB
Use the Workbench system requester. Will not be implemented.

Request ON
Use the AMOS Professional requester routine. Will not be implemented.

Request OFF
Used to cancel the requester. Will not be implemented.

Animation Commands

Instructions and functions to animate and move sprites, bobs, screens, rainbows or any other graphical objects, different than AMAL

Anim NUMBER, DEFINITION$
Description of the instruction.
Parameters:
NUMBER: Description of the parameter.
DEFINITION$: Description of the parameter.

Anim Off NUMBER
Description of the instruction.
Parameters:
NUMBER: Description of the parameter.

Anim On NUMBER, FREQUENCY
Description of the instruction.
Parameters:
NUMBER: Description of the parameter.
FREQUENCY: Description of the parameter.

Anim Freeze NUMBER
Description of the instruction.
Parameters:
NUMBER: Description of the parameter.

Move NUMBER, DEFINITIONX$, DEFINITIONY$
Description of the instruction.
Parameters:
NUMBER: Description of the parameter.
DEFINITIONX$: Description of the parameter.
DEFINITIONY$: Description of the parameter.

Move X NUMBER, DEFINITION$
Description of the instruction.
Parameters:
NUMBER: Description of the parameter.
DEFINITION$: Description of the parameter.

Move Y NUMBER, DEFINITION$
Description of the instruction.
Parameters:
NUMBER: Description of the parameter.
DEFINITION$: Description of the parameter.

Move Y NUMBER, DEFINITION$
Description of the instruction.
Parameters:

NUMBER: Description of the parameter.
DEFINITION$: Description of the parameter.

Move Off NUMBER
Description of the instruction.
Parameters:
NUMBER: Description of the parameter.

Move On NUMBER, FREQUENCY
Description of the instruction.
Parameters:
NUMBER: Description of the parameter.
FREQUENCY: Description of the parameter.

Move Freeze NUMBER
Description of the instruction.
Parameters:
NUMBER: Description of the parameter.

Movon NUMBER
Description of the function.
Parameters:
NUMBER: Description of the parameter.
Value returned:
integer: Description of the value returned.

Assets Commands

Some commands to simplify the loading of different file formats.

Load Asset FILENAME$, NUMBER
Load an external file for Image, Audio or video. If the file is an image, it can be used with the "Bob" command.
Parameters:
FILENAME$: Name of the file to load.
NUMBER: Index associated at

Load Asset FILENAME$, NAME$
Description of the instruction.
Parameters:
FILENAME$: Description of the parameter.
NAME$: Description of the parameter.

Del Asset TYPE$, NUMBER
Delete an asset from the memory.
Parameters:
TYPE$: Type of asset to delete. Must be "image", "audio" or "video".
NUMBER: Index of asset to delete

Del Asset TYPE$, NAME$
Description of the instruction.
Parameters:
TYPE$: Description of the parameter.
NAME$: Description of the parameter.

Json Asset Property INDEX, PATH$
Description of the function.
Parameters:
INDEX: Description of the parameter.
PATH$: Description of the parameter.
Value returned:
integer: Description of the value returned.

Json Asset Property$ INDEX, PATH$
Description of the function.
Parameters:
INDEX: Description of the parameter.
PATH$: Description of the parameter.
Value returned:
string: Description of the value returned.

Bank Commands

Functions and instruction to manage memory banks

Bank Swap
Description of the instruction.

Bank Shrink
Description of the instruction.

Start
Description of the function.
Value returned:
integer: Description of the value returned.

Length
Description of the function.
Value returned:
integer: Description of the value returned.

Erase
Description of the instruction.

Erase All
Description of the instruction.

Erase Temp
Destroy all temporary memory banks (bank marked as "Work")

List Bank
Description of the instruction.

Reserve As Work
Description of the instruction.

Reserve As Chip Work
Description of the instruction.

Reserve As Data
Description of the instruction.

Reserve As Chip Data
Description of the instruction.

Image Width
Description of the function.
Value returned:
integer: Description of the value returned.

Image Height
Description of the function.
Value returned:
integer: Description of the value returned.

Hrev
Description of the function.
Value returned:
integer: Description of the value returned.

Vrev
Description of the function.
Value returned:
integer: Description of the value returned.

Rev
Description of the function.
Value returned:
integer: Description of the value returned.

Ins Sprite
Description of the instruction.

Ins Sprite
Description of the instruction.

Del Sprite
Description of the instruction.

Del Sprite
Description of the instruction.

Ins Bob
Description of the instruction.

Ins Bob
Description of the instruction.

Del Bob
Description of the instruction.

Del Bob
Description of the instruction.

Ins Image
Description of the instruction.

Ins Bob
Description of the instruction.

Del Bob
Description of the instruction.

Del Bob
Description of the instruction.

Make Mask
Description of the instruction.

No Mask
Description of the instruction.

Hot Spot
Description of the instruction.

Hot Spot
Description of the instruction.

Hot Spot
Description of the function.
Value returned:
integer: Description of the value returned.

Basic Langage structure instructions.

To be documented later.

ErrTrap
Return the number of the last Trapped error, and reset the number
Value returned:
integer: The number of the last trapped error, 0 if no error occured

Err$ ERROR_NUMBER
Return the text of an error message from it's number
Parameters:
ERROR_NUMBER: The number of the error
Value returned:
string: The text of the message

Bob Commands

Instructions and functions to display moveable objects inside AOZ screens (Bitmap Objects)

Bob
Description of the instruction.

Bob
Description of the instruction.

Bob Off
Description of the instruction.

Bob Off
Description of the instruction.

Bob Update Off
Turns off the automatic bob coordinate update system. After it, all "Bob" instruction will no longer have a visible effect until an "Bob Update" instruction is useD

Bob Update On
Turns on the automatic bob coordinate update system. After it, the effect of all "Bob" instructions will be visible on display

Bob Clear
This AMOS-compatible instruction has no other effect in AOZ than to call "Bob Update"

Bob Draw
This AMOS-compatible instruction has no effect in AOZ

Limit Bob
Turns off all previous Limit bob instructions

Limit Bob X1, Y1, X2, Y2
Clip the display of all bobs to a limited area in the current screen
Parameters:
- X1: Horizontal coordinate of the top-left corner of the bounding box
- Y1: Vertical coordinate of the top-left corner of the bounding box
- X2: Horizontal coordinate of the bottom-right corner of the bounding box
- Y2: Vertical coordinate of the bottom-right corner of the bounding box

Limit Bob INDEX, X1, Y1, X2, Y2
Clip the display of a specific bob to a limited area in the current screen
Parameters:
- INDEX: Index of the bob to clip in the list of active bobs
- X1: Horizontal coordinate of the top-left corner of the bounding box
- Y1: Vertical coordinate of the top-left corner of the bounding box
- X2: Horizontal coordinate of the bottom-right corner of the bounding box
- Y2: Vertical coordinate of the bottom-right corner of the bounding box

Limit Bob X, Y, WIDTH, HEIGHT
Clip the display of all bobs to a limited area in the current screen
Parameters:
- X: Horizontal coordinate of the left of the bouding box
- Y: Vertical coordinate of the top of the bouding box
- WIDTH: Width of the bounding box
- HEIGHT: Height of the bounding box

Limit Bob INDEX, X, Y, WIDTH, HEIGHT
Clip the display of a specific bob to a limited area in the current screen
Parameters:
- INDEX: Index of the bob to clip in the list of active bobs
- X: Horizontal coordinate of the left of the bouding box
- Y: Vertical coordinate of the top of the bouding box
- WIDTH: Width of the bounding box
- HEIGHT: Height of the bounding box

Set Bob X, Y, WIDTH, HEIGHT
Description of the instruction.
Parameters:
- X: Description of the parameter.
- Y: Description of the parameter.
- WIDTH: Description of the parameter.
- HEIGHT: Description of the parameter.

Paste Bob X, Y
Description of the instruction.
Parameters:
- X: Description of the parameter.
- Y: Description of the parameter.

Bob Alpha
Description of the instruction.

Bob Show
Description of the instruction.

Bob Hide
Description of the instruction.

Bob Scale
Description of the instruction.

Bob Rotate
Description of the instruction.

Bob Skew
Description of the instruction.

Put Bob

Description of the instruction.

Priority On
Turns on automatic sorting of the Z-order of the bobs in the current screen

Priority Off
Turnsoff automatic sorting of the Z-order of the bobs in the current screen

X Bob
Description of the function.
Value returned:
integer: Description of the value returned.

Y Bob
Description of the function.
Value returned:
integer: Description of the value returned.

I Bob
Description of the function.
Value returned:
integer: Description of the value returned.

I Bob$
Description of the function.
Value returned:
string: Description of the value returned.

Priority Reverse On
Inverts the Z-Order of the bobs in the current screen if Priority is activated. After this instruction, bobs with the highest Y coordinate will be displayed in the back of others

Priority Reverse Off
Stops the inversion of the Z-Order of the bobs in the current screen if Priority is activated. After this instruction, bobs with the lowest Y coordinate will be displayed in the back of others

Isbob
Description of the function.
Value returned:
integer: Description of the value returned.

Get Bob
Description of the instruction.

Get Bob SCREENNUMBER
Description of the instruction.
Parameters:
SCREENNUMBER: Description of the parameter.

Get Bob

Description of the instruction.

Get Bob SCREENNUMBER
Description of the instruction.
Parameters:
SCREENNUMBER: Description of the parameter.

Get Bob Palette MASK
Copy the colour palette from the Images bank to the current screen. (Deprecated, use "Get Images Palette")
Parameters:
MASK: Mask of bits where each bit set to one represent a colour to capture and 0 a colour to ignore, up to 32 (optional)

Bob Add
Description of the instruction.

Bob Move
Description of the instruction.

Bob Move X
Description of the instruction.

Bob Move Y
Description of the instruction.

Bob Move Off
Description of the instruction.

Bob Move Off
Description of the instruction.

Bob Move On
Description of the instruction.

Bob Move On
Description of the instruction.

Bob Move Freeze
Description of the instruction.

Bob Move Freeze
Description of the instruction.

Bob Movon
Description of the function.
Value returned:
integer: Description of the value returned.

Bob Movon
Description of the function.

Value returned:
integer: Description of the value returned.

Bob Anim
Description of the instruction.

Bob Anim Off
Description of the instruction.

Bob Anim Off
Description of the instruction.

Bob Anim On
Description of the instruction.

Bob Anim On
Description of the instruction.

Bob Anim Freeze
Description of the instruction.

Bob Anim Freeze
Description of the instruction.

Bob Collide With
Description of the instruction.

Bob Collide
Description of the instruction.

Collision Commands

Functions to test collisions between graphical moveable objects, bobs and sprites

Bob Col INDEX, START, END
Test if one bob is colliding with a set of bobs
Parameters:
INDEX: Index of the bob to test
START: Index of the first bob to test with
END: Index of the last bob to test with
Value returned:
boolean: True if the first bob is colliding with one of the other bobs, False if not

BobSprite Col INDEX, START, END
Test if one bob is colliding with a set of sprites
Parameters:
INDEX: Index of the bob to test
START: Index of the first sprite to test with
END: Index of the last sprite to test with
Value returned:
boolean: True if bob is colliding with one of the sprites, False if not

Sprite Col INDEX, START, END
Test if one sprite is colliding with a set of sprites
Parameters:
INDEX: Index of the sprite to test
START: Index of the first sprite to test with
END: Index of the last sprite to test with
Value returned:
boolean: True if the first sprite is colliding with one of the other sprites, False if not

SpriteBob Col INDEX, START, END
Test if one sprite is colliding with a set of bobs
Parameters:
INDEX: Index of the sprite to test
START: Index of the first bob to test with
END: Index of the last bob to test with
Value returned:
boolean: True if the sprite is colliding with one of the bobs, False if not

Col INDEX
Test the status of an object after collision detection
Parameters:
INDEX: Index of the object (sprite or bob) to test. Note that this function has been extended in AOZ, if you use -1 for this parameter, it will return the INDEX of the object in collision, to avoid having to perform a loop
Value returned:
boolean: True if the object is colliding with one of the other objects, False if not

Icon Commands

Instructions and functions to handle strings

Paste Icon X, Y, INDEX, SCALEX#, SCALEY#, ANGLE#
Draw an icon from the Icons bank into a screen
Parameters:
X: Horizontal coordinate of drawing
Y: Vertical coordinate of drawing
INDEX: Index of the icon in the Icons bank
SCALEX#: Horizontal scaling of the icon, 1 = original size, 0.5 = half, 2 = twice, etc (optional)
SCALEY#: Vertical scaling of the icon, 1 = original size, 0.5 = half, 2 = twice, etc (optional)
ANGLE#: Angle of rotation, in radian (default) or degree after the "Degree" instruction has been used (optional)

Paste Icon X, Y, NAME$, SCALEX#, SCALEY#, ANGLE#
Draw an icon from the Icons bank into a screen
Parameters:
X: Horizontal coordinate of drawing
Y: Vertical coordinate of drawing
NAME$: Name of the icon in the Icons bank
SCALEX#: Horizontal scaling of the icon, 1 = original size, 0.5 = half, 2 = twice, etc (optional)
SCALEY#: Vertical scaling of the icon, 1 = original size, 0.5 = half, 2 = twice, etc (optional)
ANGLE#: Angle of rotation, in radian (default) or degree after the "Degree" instruction has been used (optional)

Make Icon Mask INDEX
Create a transparency mask out of the full black color of an image in Icons bank (RGB = $000000)
Parameters:
INDEX: Index of the image work on

Make Icon Mask NAME$
TOTEST! Create a transparency mask out of the full black color of an image in the Images bank (RGB = $000000)
Parameters:
NAME$: Name of the image work on

No Icon Mask INDEX
TOTEST! Removes the transparency mask of an icon in the Icons bank, turning the icon fully opaque
Parameters:
INDEX: Index of the icon work on

No Icon Mask NAME$
TOTEST! Removes the transparency mask of an image in the Images bank, turning the image full opaque
Parameters:
NAME$: Name of the image work on

Icon POSITION
Inserts an empty icon in Icons bank
Parameters:
POSITION: Position to insert at

Ins Icon START, END
Inserts a number of empty icons in the Icons bank
Parameters:
START: Position of insersion
END: End position of insertion

Del Icon INDEX
Delete an icon Icons bank
Parameters:
INDEX: Index of the icon to delete

Del Icon START, END
Delete a range of icons from the image bank
Parameters:
START: Position of deletion
END: End position of deletion

Get Icon X1, Y1, X2, Y2, TAGS$
Captures a portion of the current screen and add the image to the Icons bank.
Parameters:
X1: Horizontal coordinate of the top-left corner of the capture area
Y1: Vertical coordinate of the top-left corner of the capture area
X2: Horizontal coordinate of the bottom-right corner of the capture area
Y2: Vertical coordinate of the bottom-right corner of the capture area
TAGS$: Unused for the moment

Get Icon SCREENNUMBER, ICONINDEX, X1, Y1, X2, Y2, TAGS$
Captures a portion of the given screen and add the image to the Icons bank.
Parameters:
SCREENNUMBER: Index of the screen to capture from
ICONINDEX: Index of the image to create in the Icons bank
X1: Horizontal coordinate of the top-left corner of the capture area
Y1: Vertical coordinate of the top-left corner of the capture area
X2: Horizontal coordinate of the bottom-right corner of the capture area
Y2: Vertical coordinate of the bottom-right corner of the capture area
TAGS$: Unused for the moment

Get Icon ICONINDEX, X, Y, WIDTH, HEIGHT, TAGS$
Captures a portion of the current screen and add the image to the Icons bank
Parameters:
ICONINDEX: Index of the image to create in the Icons bank
X: The horizontal coordinate of the top-left pixel of the capture rectangle
Y: The vertical coordinate of the top-left pixel of the capture rectangle
WIDTH: The width of the capture rectangle
HEIGHT: The height of the capture rectangle
TAGS$: Unused for the moment

Get Sprite SCREENNUMBER, ICONINDEX, X, Y, WIDTH, HEIGHT, TAGS$
Captures a portion of the given screenand add the image to the image bank
Parameters:
SCREENNUMBER: Index of the screen to capture from
ICONINDEX: Index of the image to insert in the Images bank

X: The horizontal coordinate of the top-left pixel of the capture rectangle
Y: The vertical coordinate of the top-left pixel of the capture rectangle
WIDTH: The width of the capture rectangle
HEIGHT: The height of the capture rectangle
TAGS$: Unused for the moment

Get Icon Palette MASK
Copy the colour palette from the Icons bank to the current screen
Parameters:
MASK: Mask of bits where each bit set to 1 represent a colour to capture and 0 a colour to ignore, up to 32 (optional)

Icon Base ICON_INDEX
Returns the address of the Icon whose number is specified in brackets. Not implemented in AOZ
Parameters:
ICON_INDEX: The index of the icon in the bank
Value returned:
integer: 0 in this version

IDE Commands

Instructions and functions to communicate with the future IDE and create extensions for it

IDE End Accessory
Terminates the current application if it is running as an accessory, and restore the previous running one

IDE Connect TAGS
Connects to the IDE
Parameters:
TAGS: Unused, for future expansion

IDE Disconnect TAGS
Stops the connection with the IDE
Parameters:
TAGS: Unused, for future expansion

IDE Is Connected
Indicates if the application is connected to the IDE
Value returned:
boolean: True if the application is connected to the IDE, False if not

IDE Send Command PARAM1$, PARAM2$, PARAM3$
Send a command to the IDE
Parameters:
PARAM1$: Optional first parameter (will be stored in the "parameters" array)
PARAM2$: Optional second parameter (will be stored in the "parameters" array)
PARAM3$: Optional third parameter (will be stored in the "parameters" array)

IDE Commands
Starts the exploration of the last messages, and switch to the next one if more the next times it is called. Old messages are discarted.
Value returned:
boolean: True if there are messages to read, false if not. To be used with While - Wend loops

IDE Command$
Returns the command of the current message
Value returned:
string: The command

IDE Responses
Starts the exploration of the last responses to one, or all messages, in order of reception, and switch to the next one if more the next times it is called. Old reponses are discarted.
Value returned:
boolean: True if there are reponses to read, false if not. To be used with While - Wend loops

IDE Response$
Returns the current reponse
Value returned:

string: The current response

IDE Property$ PATH$, DEFAULT$
Returns one of the string properties of the last message
Parameters:
PATH$: A string with the path to the property
DEFAULT$: An optional string to use if the property is not found. In this case, AOZ will not generate any error
Value returned:
string: The property

IDE Property PATH$, DEFAULT
Returns the value of one of the properties of the last message
Parameters:
PATH$: A string with the path to the property
DEFAULT: An optional value to use if the property is not found. In this case, AOZ will not generate any error
Value returned:
number: The value of the property

IDE Property Type$ PATH$
Returns the type one of the property of the last message as a string
Parameters:
PATH$: A string with the path to the property
Value returned:
string: Either 'number', 'string', 'object' or 'array'. If the property is not found, then an empty string

Keyboard and mouse Commands

Instructions and functions to handle the keyboard and the mouse

Inkey$
Checks to see if a key has been pressed, and reports back its value in a string
Value returned:
string: The value of the last key pressed

Inkey$
Checks to see if a key has been pressed, and reports back its value in a string
Value returned:
string: The value of the last key pressed

Wait Key
Description of the instruction.

Key Speed TIME_LAG, DELAY_SPEED
Change key repeat speed. Not implented in HTML applications, maybe later in executable applications
Parameters:
TIME_LAG: The time-lag before repeat, in 1/50th of second
DELAY_SPEED: The delay before each repeated key, in 1/50th of second

Key State KEY_CODE
Test for a specific key press
Parameters:
KEY_CODE: The Javascript code of the key to test. The Amiga Scan Code in Amiga mode
Value returned:
boolean: true if the key is pressed, False if not

Key Shift
Return the state of the modifier keys
Value returned:
boolean: A set of flags indicating the state of the various modifier keys

Key Name$
Return the state of the modifier keys
Value returned:
boolean: A set of flags indicating the state of the various modifier keys

ScanCode
Return the scancode of a key entered with INKEY$
Value returned:
integer: The code of the key that was entered by the user and detected by the last INKEY$. Will report Javascript key-codes for PC, and Amiga ScanCodes in Amiga emulation

ScanShift
Return shift status of key entered with INKEY$
Value returned:

integer: A mask of bits indicating the sate of the SHIFT key. =0 no Shift key pressed, <>0 one of the Shift key is pressed

Put Key TEXT$
Load a string of characters directly into the keyboard buffer, enabling you to simulate user typing (in INPUT functions for example)
Parameters:
TEXT$: The texte to put in the buffer

Clear Key
Re-set the keyboard buffer. Might not have an effect on all platforms.

Scan$ SCANCODE, MASK
Return a scan-code for use with Key$
Parameters:
SCANCODE: The Javascript code of the key for PC, and the Amiga scan code for Amiga emulation
MASK: A mask of bits, wioth the same format as KEY SHIFT
Value returned:
string: The string to use with Key$

Hide On
Hide the mouse pointer (Deprecatred, use "Hide")

Hide
Hide the mouse pointer

Show On
Show the mouse pointer (Deprecatred, use "Hide")

Show
Show the mouse pointer

Change Mouse SHAPE
Change the shape of the pointer arrow
Parameters:
SHAPE: Number of the shape to use

Change Mouse SHAPE$
Description of the instruction.
Parameters:
SHAPE$: Description of the parameter.

Mouse Key
Return the status of the mouse buttons
Value returned:
integer: A mask of bits. Bit 0: Left mouse button, Bit 1: Right mouse button, Bit 2: Middle mouse button if it exists

Mouse Click
Check for click of mouse button. This is similar to MOUSE KEY, but instead of checking to see whether or not a mouse button is held down, MOUSE CLICK is only interested in whether the user has just made a single click on a mouse button.

Value returned:

integer: A mask of bits. Bit 0: Left mouse button, Bit 1: Right mouse button, Bit 2: Middle mouse button if it exists

Limit Mouse X1, Y1, X2, Y2

Change the shape of the pointer arrow

Parameters:

X1: Horizontal coordinate of the top-left corner of the bounding area

Y1: Vertical coordinate of the top-left corner of the bounding area

X2: Horizontal coordinate of the bottom-right corner of the bounding area

Y2: Vertical coordinate of the bottom-right corner of the bounding area

Limit Mouse X, Y, WIDTH, HEIGHT

Change the shape of the pointer arrow

Parameters:

X: Horizontal coordinate of the top-left corner of the bounding area

Y: Vertical coordinate of the top-left corner of the bounding area

WIDTH: Width of the bounding area

HEIGHT: Height of the bounding area

Limit Mouse

Without parameters, restore the displacement of the mouse to the whole screen

Mouse Wheel

Return the movement of the mouse wheel

Value returned:

integer: A displacement, positive or negative and dependant of the system

Maps and Tiles

Insert scenery for your games, with these commands that allow you to create and manage maps

Createmaps WIDTH, HEIGHT
Create a maps set width a specific size
Parameters:
WIDTH: Number of tiles in width
HEIGHT: Number of tiles in height

Addmap
Add a new map to the current maps set.

Removemap MAPINDEX
Remove a map from the current maps set.
Parameters:
MAPINDEX: Index of map to remove.

Insertmap MAPINDEX
Insert a new map int the current maps set at the position given.
Parameters:
MAPINDEX: Position of the new map in the current maps set.

Loadmaps MAPSNAME$
Load a maps file. File must be one of these formats : aozmap or tmx (Tiled)
Parameters:
MAPSNAME$: Filename to load.

Getmapindex VIEWID$
Return the index of map associated to a map view.
Parameters:
VIEWID$: Name of the map view.
Value returned:
integer: DIndex of the map.

Setmapindex VIEWID$, MAPINDEX
Link a map to a map view.
Parameters:
VIEWID$: Name of the map view.
MAPINDEX: Index of the map.

Getmapwidth VIEWID$
Return the number of tiles in width of a map view.
Parameters:
VIEWID$: Name of the map view.
Value returned:
integer: Number of tiles in width.

Getmapheight VIEWID$

Return the number of tiles in height of a map view.
Parameters:
VIEWID$: Name of the map view.
Value returned:
integer: Number of tiles in height.

Gettilevalue$ VIEWID$, TILEINDEX
Return the value of a tile of a Map View.
Parameters:
VIEWID$: Name of the map view.
TILEINDEX: Index of tile.
Value returned:
string: The value of tile.

Gettilevalue$ VIEWID$, X, Y
Return the value of a tile of a Map View for a position XY.
Parameters:
VIEWID$: Name of the map view.
X: Horizontal position of the tile.
Y: Vertical position of the tile.
Value returned:
string: The value of tile.

Settilevalue VIEWID$, TILEINDEX, VALUE$
Set the value of a tile of a Map View.
Parameters:
VIEWID$: Name of the map view.
TILEINDEX: Index of tile.
VALUE$: Value of the tile.

Settilevalue VIEWID$, X, Y, VALUE$
Set the value of a tile of a Map View.
Parameters:
VIEWID$: Name of the map view.
X: Horizontal position of the tile.
Y: Vertical position of the tile.
VALUE$: Value of the tile.

Maptile Replace VIEWID$, TILESEARCH$, TILEREPLACE$
Replaces all tiles with the searched value by the desired value int a Map View.
Parameters:
VIEWID$: Name of the map view.
TILESEARCH$: Value of tile searched.
TILEREPLACE$: Value of replacement.

Resetmaps
Reset the maps set and clean all the maps datas.

Mapcount
Return the number of maps in the current maps set.
Value returned:

integer: The number of maps.

Mapview Copy VIEWID$
Store the actual state of a map view.
Parameters:
VIEWID$: Name of the map view.

Mapview Restore VIEWID$
Restore the state of a map view stored by "MapView Copy" commands.
Parameters:
VIEWID$: Name of the map view.

Mapview Open VIEWID$, X, Y, WIDTH, HEIGHT, MAPINDEX
Open a view to display a map on the current screen.
Parameters:
VIEWID$: Name of the map view.
X: Position X in pixel of the view on the current Screen of the map view.
Y: Position Y in pixel of the view on the current Screen of the map view.
WIDTH: Width of the view in pixel of the map view.
HEIGHT: Height of the view in pixel of the map view.
MAPINDEX: Index of the map.

Mapview Close VIEWID$
Close a map view opened with the "MapView Open" commands.
Parameters:
VIEWID$: Name of the map view.

Mapview Offset VIEWID$, X, Y, Z
Scrolling of the map into the map view.
Parameters:
VIEWID$: Name of the map view.
X: Position X in pixel of the scrolling of the map view.
Y: Position Y in pixel of the scrolling of the map view.
Z: Position Z in pixel of the scrolling of the map view (not used).

Mapview Display VIEWID$, X, Y
Set the position of the map view on the current screen.
Parameters:
VIEWID$: Name of the map view.
X: Position X in pixel of the view on the current Screen of the map view.
Y: Position Y in pixel of the view on the current Screen of the map view.

Mapview Display VIEWID$, X, Y, WIDTH, HEIGHT
Set the position and the size of the map view on the current screen.
Parameters:
VIEWID$: Name of the map view.
X: Position X in pixel of the view on the current Screen of the map view.
Y: Position Y in pixel of the view on the current Screen of the map view.
WIDTH: Width of the view in pixel of the map view.
HEIGHT: Height of the view in pixel of the map view.

Mapview Offset X VIEWID$
Returns the position X of the map view on the screen.
Parameters:
VIEWID$: Name of the map view.
Value returned:
integer: Position X in pixel.

Mapview Offset Y VIEWID$
Returns the position Y of the map view on the screen.
Parameters:
VIEWID$: Name of the map view.
Value returned:
integer: Position Y in pixel.

Mapview Offset Z VIEWID$
Returns the position Z of the map view on the screen.
Parameters:
VIEWID$: Name of the map view.
Value returned:
integer: Position Z.

Mapview Redraw VIEWID$
Redraw the map view on the screen.
Parameters:
VIEWID$: Name of the map view.

Mapview Redraw VIEWID$, BACKCOLOR
Redraw the map view on the screen with a background color.
Parameters:
VIEWID$: Name of the map view.
BACKCOLOR: Color RGB of the background.

Maptile Ref VALUE$, NUMIMAGE
Link a image of the images bank to a tile value.
Parameters:
VALUE$: Tile value to associated.
NUMIMAGE: Index of image of the images bank.

Maptile Ref VALUE$, NUMIMAGE, OBJECTNAME$
Link a image of the images bank to a tile value for a specific map view (not used).
Parameters:
VALUE$: Tile value to associated.
NUMIMAGE: Index of image of the images bank.
OBJECTNAME$: Name of object.

Maptile Ref VIEWID$, VALUE$, NUMIMAGE
Link a image of the images bank to a tile value for a specific map view.
Parameters:
VIEWID$: Name of the map view
VALUE$: Tile value to associated.
NUMIMAGE: Index of image of the images bank.

Maptile Size WIDTH, HEIGHT
Set the width and height of the map tiles in pixel.
Parameters:
WIDTH: Width of the map tiles in pixel.
HEIGHT: Height of the map tiles in pixel.

Maptile Size VIEWID$, WIDTH, HEIGHT
Set the width and height of the map tiles in pixel for a specific map view.
Parameters:
VIEWID$: Name of map view.
WIDTH: Width of the map tiles in pixel.
HEIGHT: Height of the map tiles in pixel.

Maptile Width VIEWID$
Return the width of the tiles map for a specific map view.
Parameters:
VIEWID$: Name of the map view.
Value returned:
integer: Width of the map tiles.

Maptile Height VIEWID$
Return the height of the tiles map for a specific map view.
Parameters:
VIEWID$: Name of the map view.
Value returned:
integer: Height of the map tiles.

Maptile Count VIEWID$, TILEVALUE$
Return the number of a specifis tile value in map view.
Parameters:
VIEWID$: Name of the map view.
TILEVALUE$: Value of the searched tile.
Value returned:
integer: Number of tiles found.

Maptile Find VIEWID$, TILEVALUE$
Return the first number of tile on the map view where is tile value asked.
Parameters:
VIEWID$: Name of the map view.
TILEVALUE$: Value of the searched tile.
Value returned:
integer: The first index of tile found. If -1 is returned then no tile found.

Maptile Next
Return the next number of tile on the map view after to had called the "MapTile Find" command.
Value returned:
integer: The next index of tile found. if -1 is returned then no tile found.

Maptile X VIEWID$, TILEINDEX
Return the horizontal position in pixel of a tile in the map view.

Parameters:

VIEWID$: Name of map view.

TILEINDEX: Index of tile.

Value returned:

integer: The horizontal position in pixel.

Maptile Y VIEWID$, TILEINDEX

Return the vertical position in pixel of a tile in the map view.

Parameters:

VIEWID$: Name of map view.

TILEINDEX: Index of tile.

Value returned:

integer: The vertical position in pixel.

Maptile Test VIEWID$, X, Y

Return the index of tile at the position x and y on a map view.

Parameters:

VIEWID$: Name of map view.

X: Position X in pixel to test.

Y: Position Y in pixel to test.

Value returned:

integer: Index of tile found.

Mathematical Commands

Mathmatical and calculation instructions and functions

Fix NUMBER
Set the number of decimal of floating point numbers to ASCII conversion (used in both Print and =Str$())
Parameters:
NUMBER: The number of decimals

Radian
Set the default angle representation in AOZ to Radians.

Degree
Set the default angle representation in AOZ to degrees.

Rnd CEILING
Return a random number
Parameters:
CEILING: An optional value indicating to generate the value as an integer betwwen 0 and CEILING. If not specified the function will return a floating point value between 0 and 1
Value returned:
integer: A value between 0 and CEILING (excluded) if the parameter is specified

Randomize SEED
Switch the random number generator to Mersenne Twist and generate a new seed.
Parameters:
SEED: An optional SEED to initiate the generation of random numbers. If not specified a value callated out of the TIMER will be used.

Randomize NUMBER
Return the sign of a number
Parameters:
NUMBER: The value to get the sign of
Value returned:
integer: -1 if the number is negative, 1 if it is greater than 0, 0 if it is equal to zero

Abs NUMBER
Return the absolute value of a number
Parameters:
NUMBER: The number to get the absolute value from
Value returned:
number: NUMBER if the number is positive, -NUMBER if the number is negative

Int FLOATNUMBER
Return the greatest integer number below the given value
Parameters:
FLOATNUMBER: The number to get the integral part from
Value returned:
integer: If FLOATNUMBER is greater than zero, return the integral part, if negative, return integral(FLOATNUMBER)

Pi#
Description of the function.
Value returned:
decimal: Description of the value returned.

Min number, number
Return the minimal value of two numbers
Parameters:
number: The first value to test
number: The second value to test
Value returned:
number: The minimal value of the two numbers

Min string, string
Compares two strings and return the one beginning with the minimal ASCII representation value
Parameters:
string: The first string to test
string: The second string to test
Value returned:
number: The string beginning with the minimal ASCII representation

Max number, number
Return the maximal value of two numbers
Parameters:
number: The first value to test
number: The second value to test
Value returned:
number: The maximal value of the two numbers

Max string, string
Compares two strings and return the one beginning with the maximal ASCII representation value
Parameters:
string: The first string to test
string: The second string to test
Value returned:
number: The string beginning with the maximal ASCII representation

Sin angle
Return the Sine of an angle
Parameters:
angle: The angle to calculate the Sine from, in Radian by default and degrees after the "Degree" instruction has been used
Value returned:
float: The value of the Sine of the angle

Cos angle
Return the Cosine of an angle
Parameters:
angle: The angle to calculate the Cosine from, in Radian by default and degrees after the "Degree" instruction has

been used
Value returned:
float: The value of the Cosine of the angle

Tan angle
Return the Tangent of an angle
Parameters:
angle: The angle to calculate the Tangent from, in Radian by default and degrees after the "Degree" instruction has been used
Value returned:
float: The value of the Tangent of the angle

ASin number
Return the Arc Sine of a number
Parameters:
number: The number from which to extract the Arc Sine from
Value returned:
angle: The value of the Arc Sine, an angle expressed in Radian by default or degree after the "Degree" instruction has been used

ACos number
Return the Arc Cosine of a number
Parameters:
number: The number from which to extract the Arc Cosine from
Value returned:
angle: The value of the Arc Cosine, an angle expressed in Radian by default or degree after the "Degree" instruction has been used

ATan number
Return the Arc Tangent of a number
Parameters:
number: The number from which to extract the Arc Tangent from
Value returned:
angle: The value of the Arc Tangent, an angle expressed in Radian by default or degree after the "Degree" instruction has been used

HSin angle
Return the Hyperbolic Sine of a number
Parameters:
angle: The angle to calculate the Hyperbolic Sine from, in Radian by default and degrees after the "Degree" instruction has been used
Value returned:
float: The value of the Hyperbolic Sine of the angle

HCos angle
Return the Hyperbolic Cosine of a number
Parameters:
angle: The angle to calculate the Hyperbolic Cosine from, in Radian by default and degrees after the "Degree" instruction has been used
Value returned:
float: The value of the Hyperbolic Cosine of the angle

HTan angle
Return the Hyperbolic Tangent of a number
Parameters:
angle: The angle to calculate the Hyperbolic Tangent from, in Radian by default and degrees after the "Degree" instruction has been used
Value returned:
float: The value of the Tangent of the angle

Sqr number
Return the square root of a number
Parameters:
number: The positive number out of which to calculate the square root
Value returned:
float: The value of the Square Root of the number

Log number
Return the base 10 logarythm of a number
Parameters:
number: The number out of which to calculate the base 10 logarythm
Value returned:
float: The value of the base 10 logarythm

Ln number
Return the Neperian Logarythm of a number
Parameters:
number: The number out of which to calculate the Neperian Logarythm
Value returned:
float: The value of the Neperian Logarythm

Exp number
Return the exponantial of a number
Parameters:
number: The number out of which to calculate the Exponantial
Value returned:
float: The value of the Exponantial

Angle# X1, Y1, X2, Y2
Description of the function.
Parameters:
X1: Description of the parameter.
Y1: Description of the parameter.
X2: Description of the parameter.
Y2: Description of the parameter.
Value returned:
decimal: Description of the value returned.

Distance# X1, Y1, X2, Y2
Description of the function.
Parameters:
X1: Description of the parameter.

Y1: Description of the parameter.
X2: Description of the parameter.
Y2: Description of the parameter.

Value returned:

decimal: Description of the value returned.

Memory Commands

Instructions and functions to handle memory access.

Poke ADDRESS, VALUE
Change a one-byte word at a memory address
Parameters:
ADDRESS: The address to change, should be resolved in AOZ by using the "Start" function to get the address of a bank (magical number) and adding the offset to the address. Normal computer memory is inaccessible.
VALUE: The value to set

Doke ADDRESS, VALUE
Change a two-byte word at a memory address. Value is set in little-endian in PC mode and big-endian in Amiga mode
Parameters:
ADDRESS: The address to change, should be resolved in AOZ by using the "Start" function to get the address of a bank (magical number) and adding the offset to the address. Normal computer memory is inaccessible.
VALUE: The value to set

Loke ADDRESS, VALUE
Change a four-byte word at a memory address. Value is set in little-endian in PC mode and big-endian in Amiga mode
Parameters:
ADDRESS: The address to change, should be resolved in AOZ by using the "Start" function to get the address of a bank (magical number) and adding the offset to the address. Normal computer memory is inaccessible.
VALUE: The value to set

Poke$ ADDRESS, TEXT$
Write the ascii values of a string in memory
Parameters:
ADDRESS: The address to write to, should be resolved in AOZ by using the "Start" function to get the address of a bank (magical number) and adding the offset to the address. Normal computer memory is inaccessible.
TEXT$: The string to write

Peek$ ADDRESS, LENGTH, STOP
Read a string from memory
Parameters:
ADDRESS: The address to read, should be resolved in AOZ by using the "Start" function to get the address of a bank (magical number) and adding the offset to the address. Normal computer memory is inaccessible.
LENGTH: The number of bytes to read.
STOP: The ascii code of the character signifying the end of the string. Default is 0
Value returned:
string: The string contained at the address

Peek ADDRESS
read a byte from an address
Parameters:
ADDRESS: The address to change, should be resolved in AOZ by using the "Start" function to get the address of a bank (magical number) and adding the offset to the address. Normal computer memory is inaccessible.
Value returned:
integer: The value contained at the address

Deek ADDRESS
Read a two-bytes value from an address. Value is read in little-endian in PC mode and big-endian in Amiga mode
Parameters:
ADDRESS: The address to change, should be resolved in AOZ by using the "Start" function to get the address of a bank (magical number) and adding the offset to the address. Normal computer memory is inaccessible.
Value returned:
integer: The value contained at the address

Leek ADDRESS
Read a four-bytes value from an address. Value is read in little-endian in PC mode and big-endian in Amiga mode
Parameters:
ADDRESS: The address to change, should be resolved in AOZ by using the "Start" function to get the address of a bank (magical number) and adding the offset to the address. Normal computer memory is inaccessible.
Value returned:
integer: The value contained at the address

Free
Return the amount of free memory on the machine. Warnign, not tested and maybe deprecated in the future./ The original AMOS instruction reported the amount of free memory in the variable buffer area
Value returned:
integer: The amount of free memory

Chip Free
Return the amount of free memory on the machine. Warnign, not tested and maybe deprecated in the future./ The original AMOS instruction reported the amount of free chip memory of the machine, which has no meaning tocay
Value returned:
integer: The amount of free memory

Chip Free
Return the amount of free memory on the machine. Warnign, not tested and maybe deprecated in the future./ The original AMOS instruction reported the amount of free chip memory of the machine, which has no meaning tocay
Value returned:
integer: The amount of free memory

Fill START, FINISH, PATTERN
fill memory block with the contents of a variable.
Parameters:
START: The address to change, should be resolved in AOZ by using the "Start" function to get the address of a bank (magical number) and adding the offset to the address. Normal computer memory is inaccessible.
FINISH: The address of the end of the area to fill
PATTERN: The four-byte pattern that will be repeated when filling

Copy START, FINISH, DESTINATION
Copy a memory block
Parameters:
START: The address of the block to copy, should be resolved in AOZ by using the "Start" function to get the address of a bank (magical number) and adding the offset to the address. Normal computer memory is inaccessible.
FINISH: The address of the end of the area to copy
DESTINATION: The destination address, should be resolved in AOZ by using the "Start" function to get the address of a bank (magical number) and adding the offset to the address. Normal computer memory is inaccessible.

Hunt START, FINISH, TEXT$
Find a string of characters in memory
Parameters:
START: The address of the block to copy, should be resolved in AOZ by using the "Start" function to get the address of a bank (magical number) and adding the offset to the address. Normal computer memory is inaccessible.
FINISH: The address of the end of the area to copy
TEXT$: The string to look for
Value returned:
integer: The position of the first character if found, zero if not found

BTst POSITION, VALUE
Test a bit in an interger number
Parameters:
POSITION: The number of the bit to test
VALUE: The number to test
Value returned:
integer: True if the bit is set to one, False if not

Exec COMMAND$, OUTPUT$
Execute a system command. This instruction will be implemented in executable and node.js transpiled applications
Parameters:
COMMAND$: The command with it parameters
OUTPUT$: The DOS-like output, like "StdOut"

Set Stack SPACE
Deprecated, wa used to set the stack space on the Amiga
Parameters:
SPACE: The length opf the stack

Set Equate Bank BANK_NUMBER
Deprecated, was used on the Amiga
Parameters:
BANK_NUMBER:

Call ADDRESS, PARAMETER
Deprecated, was used on the Amiga
Parameters:
ADDRESS:
PARAMETER:

ExeCall OFFSET
Deprecated, was used on the Amiga
Parameters:
OFFSET:

GfxCall OFFSET
Deprecated, was used on the Amiga
Parameters:
OFFSET:

DosCall OFFSET

Deprecated, was used on the Amiga

Parameters:

OFFSET:

IntCall OFFSET

Deprecated, was used on the Amiga

Parameters:

OFFSET:

Menu Commands

Instructions and functions to handle menus in AOZ applications.

On Menu Del
Delete labels and procedures used by ON MENU. Will be implemented later.

On Menu ON
Toggle automatic menu selection. Will be implemented later.

On Menu OFF
Toggle automatic menu selection. Will be implemented later.

Menu To Bank BANK_NUMBER
Save menu definitions into a memory bank. Will be implemented later.
Parameters:
BANK_NUMBER: The number of the bank to save to

Bank To Menu BANK_NUMBER
Restore a menu definition saved in a menu bank. Will be implemented later.
Parameters:
BANK_NUMBER: The number of the bank to restore

Menu ON
Activate a menu. Will be implemented later.

Menu OFF
Deactivate a menu. Will be implemented later.

Menu Mouse ON
Display the menu at position of mouse cursor

Menu Mouse OFF
Stop displayiong the menu at position of mouse cursor

Menu Base X, Y
Move the starting position of a menu. Amiga only, will not be implemented.
Parameters:
X:
Y:

Set Menu ITEM, X, Y
Move a menu item. Amiga only, will not be implemented.
Parameters:
ITEM:
X:
Y:

Menu Key ITEM, KEY$
Assign a key to a menu item.. Will be implemented later.
Parameters:
ITEM:
KEY$:

Menu Key ITEM, SCANCODE, BITMAP
Assign a key to a menu item.. Will be implemented later.
Parameters:
ITEM:
SCANCODE:
BITMAP:

Menu Bar LEVEL
Display menu items as a vertical bar. Will not be implemented.
Parameters:
LEVEL:

Menu Line LEVEL
Display menu items as a horizontal bar. Will not be implemented.
Parameters:
LEVEL:

Menu TLine LEVEL
Display menu items as a total horizontal bar. Will not be implemented.
Parameters:
LEVEL:

Menu Movable LEVEL
Activate automatic menu movement. Will not be implemented.
Parameters:
LEVEL:

Menu Static LEVEL
Stop automatic menu movement. Will not be implemented.
Parameters:
LEVEL:

Menu Item Static LEVEL
Fix items in static positions. Will not be implemented.
Parameters:
LEVEL:

Menu Item Movable LEVEL
Move individual menu options. Will not be implemented.
Parameters:
LEVEL:

Menu Active LEVEL
Activate a menu item. Will be implemented later.
Parameters:

LEVEL:

Menu Inactive LEVEL
Deactivate a menu item. Will be implemented later.
Parameters:
LEVEL:

Menu Separate LEVEL
Separate a list of menu items. Will not be implemented.
Parameters:
LEVEL:

Menu Link LEVEL
Link a list of menu items. Will be implemented later.
Parameters:
LEVEL:

Menu Called LEVEL
Re-draw a menu item continually. Will not be implemented.
Parameters:
LEVEL:

Menu Once LEVEL
Turn off automatic re-drawing. Will not be implemented.
Parameters:
LEVEL:

Menu Del LEVEL
Delete one or more menu items. Will be implemented later.
Parameters:
LEVEL:

X Menu LEVEL
Return the graphical x-coordinate of a menu item. Will not be implemented.
Parameters:
LEVEL:
Value returned:
integer: The graphical x-coordinate of a menu item. 0 in this version.

Y Menu LEVEL
Return the graphical x-coordinate of a menu item. Will not be implemented.
Parameters:
LEVEL:
Value returned:
integer: The graphical y-coordinate of a menu item. 0 in this version.

Menu$ LEVEL
Reserved variable: define a menu title or option. Will be implemented later.
Parameters:
LEVEL:
Value returned:

integer: The text of the menu item. "" in this version.

Choice LEVEL
Read a menu item. Will be implemented later.
Parameters:
LEVEL:
Value returned:
integer: True if the menu item has been selected, False if not.

On Menu Proc PROC
Automatic menu selection. Will be implemented later.
Parameters:
PROC:

On Menu Gosub LABEL
Automatic menu selection. Will be implemented later.
Parameters:
LABEL:

On Menu Goto LABEL
Automatic menu selection. Will be implemented later.
Parameters:
LABEL:

Network

All commands to communicate with the network.

Open URL URL$
Open a URL in the current tab
Parameters:
URL$: Address will be opened into a window ("www.aoz.studio" by example)

Open URL FRAMEID$, URL$
Open a URL in a named Tab
Parameters:
FRAMEID$: Must be '_self', '_top', '_blank' or '_parent'. If empty else a iframe will be created
URL$: Address will be opened into a window ("www.aoz.studio" by example)

Call Service METHOD$, URL$, RESPONSETYPE$, PARAMETERS$, PROCSUCCESS$, PROCERROR$
Call a service and execute a request
Parameters:
METHOD$: Method used to send the request to the service ('get', 'post' or 'head')
URL$: Address of the service ("www.aoz.studio/applist.php" by example)
RESPONSETYPE$: Type of the response returned by the service ('json', 'text' or 'xml')
PARAMETERS$: List of parameters to send with the request ("?id=jdsfhjf&catalog=games&letter=a&page=22" by example)
PROCSUCCESS$: Name of the AOZ Procedure receiving the response of the request
PROCERROR$: Name of the AOZ Procedure if the request retuns an error

Call Service URL$, PARAMETERS$, PROCSUCCESS$, PROCERROR$
Call a service and execute a request in POST method and wait a JSON response
Parameters:
URL$: Address of the service ("www.aoz.studio/applist.php" by example)
PARAMETERS$: List of parameters to send with the request ("?id=jdsfhjf&catalog=games&letter=a&page=22" by example)
PROCSUCCESS$: Name of the AOZ Procedure receiving the response of the request
PROCERROR$: Name of the AOZ Procedure if the request retuns an error

Test URL URL$
Test the existing of an URL and store True or False in the response
Parameters:
URL$: Address of the service ("www.aoz.studio/applist.php" by example)

URL Exists
Return True or False if the URL tested with "Test URL" command exists
Value returned:
boolean: The state of URL

Open Track Editor MUSICFILE$
Description of the instruction.
Parameters:
MUSICFILE$: Description of the parameter.

Open Track Editor
Description of the instruction.

Open Sprite Editor
Description of the instruction.

Rainbow Commands

Rainbow instructions and functions

Set Rainbow INDEX, INK, HEIGHT, RED$, GREEN$, BLUE$, ALPHA$
Define a new Rainbow effect
Parameters:
INDEX: The index of the rainbow
INK: The index of Ink in the current screen to affect
HEIGHT: The height of the internal color buffer
RED$: A string defining the variation of the RED component of the color
GREEN$: A string defining the variation of the GREEN component of the color
BLUE$: A string defining the variation of the BLUE component of the color
ALPHA$: A string defining the variation of the ALPHA component of the color

Rainbow INDEX, OFFSET, POSITION, HEIGHT$
Define the display of a rainbow already create with "Set Rainbow"
Parameters:
INDEX: The index of the rainbow to display
OFFSET: The position within the internal color buffer to display ont he first line of display
POSITION: The horizontal position on
HEIGHT$: The height of the rainbow on display in pixels

Rainbow Del INDEX
Destroys a Rainbow
Parameters:
INDEX: The index of the rainbow to destroy

REXX Commands

Instructions and functions to call system REXX commands on the Amiga. Maybe implemented later

REXX Open PORT_NAME$
Open an REXX communication port. Maybe implemented later.
Parameters:
PORT_NAME$:

Arexx Open PORT$
Description of the instruction.
Parameters:
PORT$: Description of the parameter.

REXX Close
Close the REXX communication port. Maybe implemented later.

Arexx Close
Description of the instruction.

REXX Wait
Wait for a message from an AREXX program. Maybe implemented later.

Arexx Wait
Description of the instruction.

REXX Answer VALUE, RETURN$
answer a message from an AREXX program. Maybe implemented later.
Parameters:
VALUE:
RETURN$:

Arexx Wait VALUE, RETURN$
Description of the instruction.
Parameters:
VALUE: Description of the parameter.
RETURN$: Description of the parameter.

REXX$
Get a message from an REXX program. Maybe implemented later.
Value returned:
string:

REXX$
Get a message from an REXX program. Maybe implemented later.
Value returned:
string:

REXX

Get a message from an REXX program. Maybe implemented later.
Value returned:
integer:

REXX
Get a message from an REXX program. Maybe implemented later.
Value returned:
integer:

Samples

Instructions set to play sounds effects.

Boom
BOOM command plays a realistic explosive sound effect.
This does not delay the program at all, so it may be necessary to use WAIT between successive explosions, or to create ricochet and echo effects.

Shoot
Generate percussive sound effect.
The SHOOT command generates a simple sound effect in exactly the same way as Boom.

Bell
Generate pure tone.
Unlike the built-in explosive sound effects, Bell produces a simple pure tone.

Sam Play SAMPLE
Play a sound sample from the sample bank
The Sam Play command is used to play a digital sound sample through your audio system. Simply define the number of the required sample held in the bank. There is no limit to the number of samples that can be stored, other than available memory.
Parameters:
SAMPLE: Index of the sample into the bank to play

Sam Play NAME$
Play a sound sample from the sample bank
The Sam Play command is used to play a digital sound sample through your audio system. Simply define the number of the required sample held in the bank. There is no limit to the number of samples that can be stored, other than available memory.
Parameters:
NAME$: Name of the sample into the bank to play

Sam Play VOICE, SAMPLE, FREQUENCY
Play a sound sample from the sample bank on a voice only, with a frequency
The Sam Play command is used to play a digital sound sample through your audio system. Simply define the number of the required sample held in the bank. There is no limit to the number of samples that can be stored, other than available memory.
Parameters:
VOICE: Index of the voice where the sample will be played.
SAMPLE: Index of the sample into the bank to play
FREQUENCY: Frequency to play the sample. The setting is given in Hertz.

Sam Play VOICE, NAME$, FREQUENCY
Play a sound sample from the sample bank on a voice only, with a frequency
The Sam Play command is used to play a digital sound sample through your audio system. Simply define the number of the required sample held in the bank. There is no limit to the number of samples that can be stored, other than available memory.
Parameters:

VOICE: Index of the voice where the sample will be played.
NAME$: Name of the sample into the bank to play
FREQUENCY: Frequency to play the sample. The setting is given in Hertz.

Sam Stop VOICE
Stop one or more samples playing
This simple command is used to stop all samples playing through your loudspeaker system.
Parameters:
VOICE: Index of the voice.

Volume LEVEL, VOICE
Define the volume of a voice
Parameters:
LEVEL: Value of the volume (0-mute).
VOICE: Index of the voice.

Voice BITMASK
Activate a voice
Parameters:
BITMASK: A mask of bit, where bit 0 indicates voice 0, bit 1 voice 1 etc.

Play PITCH, DELAY
Play a voice
Parameters:
PITCH: The note to play, from 1 to 96
DELAY: The length of any pause between this PLAY command and then next, in 1/1000th of second in PC mode, and 1/50th of second in Amiga mode

Play BITMASK, PITCH, DELAY
Play a voice
Parameters:
BITMASK: A mask of bit, where bit 0 indicates voice 0, bit 1 voice 1 etc.
PITCH: The note to play, from 1 to 96
DELAY: The length of any pause between this PLAY command and then next, in 1/1000th of second in PC mode, and 1/50th of second in Amiga mode

Play OFF BITMASK
Stop playing a voice
Parameters:
BITMASK: A mask of bit, where bit 0 indicates voice 0, bit 1 voice 1 etc.

Set Wave NUMBER, SHAPE$
Define a wave form
Parameters:
NUMBER: The number of the wave to define
SHAPE$: The definition of the shape

Wave NUMBER, BITMASK
Assign a wave to sound channel
Parameters:
NUMBER: The number of the wave to define

BITMASK: A mask of bit, where bit 0 indicates voice 0, bit 1 voice 1 etc.

Noise To BITMASK
Assign noise wave to sound channel
Parameters:
BITMASK: A mask of bit, where bit 0 indicates voice 0, bit 1 voice 1 etc.

Sample NUMBER, BITMASK
Assign noise wave to sound channel
Parameters:
NUMBER: The number of the wave to define
BITMASK: A mask of bit, where bit 0 indicates voice 0, bit 1 voice 1 etc.

Del Wave NUMBER
Delete a wave
Parameters:
NUMBER: The number of the wave to delete

Set Envel NUMBER, PHASE, DURATION, VOLUME
Create a volume envelope
Parameters:
NUMBER: The number of the wave to create
PHASE: Refers to one of seven individual sections of the original wave form that is to be defined, ranging from 0 to 6
DURATION: Controls the length of this particular segment (phase number) of the wave form, and is expressed in units of one 1/1000th of a second in PC mode and 1/50th of a second in Amiga mode
VOLUME: Specifies the volume to be reached by the end of this phase

Led ON
Toggle audio filter. Only for Amiga emulation.

Led OFF
Toggle off audio filter. Only for Amiga emulation.

VuMeter VOICE
Return the volume level of a voice
Parameters:
VOICE: The number of the voice
Value returned:
integer: The current audio level of the voice

Sam Bank BANK_NUMBER
Change the current Samples bank
Parameters:
BANK_NUMBER: The index of the new bank

Sam Raw VOICES, ADDRESS, LENGTH, FREQUENCY
TODO! Play a raw sample from memory
Parameters:
VOICES: A mask of bit, where bit 0 indicates voice 0, bit 1 voice 1 etc.
ADDRESS: The address of the sound samples, must be part of an AOZ memory bank
LENGTH: The number of bytes to read

FREQUENCY: The frequency at which to play, in Herz

Sam Loop ON
TODO! Start repetition of a sample

Sam Loop OFF
TODO! Start repetition of a sample

SLoad CHANNEL, ADDRESS, LENGTH
TODO! Load a section of a sample
Parameters:
CHANNEL: The number of the file opened with "Open In" to load from
ADDRESS: The address to write to (must be inside an AOZ memory bank)
LENGTH: The number of byytes to write

SSave CHANNEL, START, END
TODO! Load a section of a sample
Parameters:
CHANNEL: The number of the channel of a file open with "Open Out" to save to
START: The address to write to (must be inside an AOZ memory bank)
END: The end of the data zone to save

Sam Swap VOICES, ADDRESS, LENGTH
Activate sample switching. Deprecated, will not have any effect, Amiga legacy.
Parameters:
VOICES: A mask of bit, where bit 0 indicates voice 0, bit 1 voice 1 etc.
ADDRESS: The address of the sound samples, must be part of an AOZ memory bank
LENGTH: The number of bytes

Sam Swap VOICE
Test for successful sample swap. Deprecated, will not have any effect, Amiga legacy.
Parameters:
VOICE: The number of the voicde to test
Value returned:
boolean:

Music NUMBER
Play a piece of AMOS Professional format music. May not be implemented in favor of tracker musics.
Parameters:
NUMBER: The number of the music in the music bank

Music Stop
Stop an AMOS Professional music. May not be implemented in favor of tracker musics.

Music Stop
Turn off all musics. May not be implemented in favor of tracker musics.

MVolume
Set the volume of a piece of music. May not be implemented in favor of tracker musics.

MVolume TEMPO

Change the speed of a piece of music. May not be implemented in favor of tracker musics.

Parameters:

TEMPO: The tempo, from 1 (really slow) to 100 (really fast)

Screen Commands

Screen instructions and functions

Screen Open INDEX, WIDTH, HEIGHT, NUMBEROFCOLOURS, PIXELMODE
Open a new screen
Parameters:
INDEX: The index of the screen to open. Any existing screen will be replaced by the new one
WIDTH: The width of the screen in pixels
HEIGHT: The height of the screen in pixels
NUMBEROFCOLOURS: the number of colors of the palette (optional)
PIXELMODE: "Lowres", "Hires", "Laced" or any combination

Screen close INDEX
Destroys the current screen or a given screen
Parameters:
INDEX: The index of the screen to destroy, if ommited will destroy the current screen

Screen Clone INDEX
Create an exact and synchronized copy of the current screen that can be displayed at another position and Z-order. Both screen share the same internal pixel buffers. Graphical operations are fobidden in the cloned screen
Parameters:
INDEX: The index of the screen to create, will replace an existing screen

Screen Hide INDEX
Make a screen disappear from display. The screen will remain active and drawing operation are still possible after this instruction
Parameters:
INDEX: The index of the screen (optional)

Screen Show INDEX
Make a hidden screen reappear on display
Parameters:
INDEX: The index of the screen (optional)

Screen Swap INDEX
Swaps the physical and logical buffers of a screen. (Deprecated in AOZ, legacy instruction, has no real effect)
Parameters:
INDEX: The index of the screen (optional)

Screen Display INDEX, X, Y, WIDTH, HEIGHT
Defines the display position and width of a screen
Parameters:
INDEX: The index of the screen to display
X: The horizontal coordinate of the top-left pixel of the screen on display (hardware coordinate)
Y: The vertical coordinate of the top-left pixel of the screen on display (hardware coordinate)
WIDTH: The number of horizontal pixels to display
HEIGHT: The number of vertical pixels to display

Screen Center INDEX, CENTERX, CENTERY
TOTEST! Enforces the centering of a screen
Parameters:
INDEX: The index of the screen to display
CENTERX: True to center the screen horizontally, False to leave the horizontal position unchanged
CENTERY: True to center the screen vertically, False to leave the vertical position unchanged

Screen Offset INDEX, OFFSETX, OFFSETY
Set the offet in the internal screen buffer of the top-left diplayed pixel, allowing scrollings
Parameters:
INDEX: The index of the screen
OFFSETX: The horizontal offset (optional)
OFFSETY: The vertical offset (optional)

Screen To Front INDEX
Change the display order of the screen, and passes the screen in front of all other screens
Parameters:
INDEX: The index of the screen

Screen To Back INDEX
Change the display order of the screen, and passes the screen behind all other screens
Parameters:
INDEX: The index of the screen (optional)

Screen Hotspot INDEX, X, Y
Set the hot-spot of a screen to a given coordinate. The hot-spot is the position within the screen where the Screen Offset will have effect and around which rotation will be done
Parameters:
INDEX: The index of the screen (optional)
X: The horizontal position of the hot-spot
Y: The vertical position of the hot-spot

Screen Hotspot INDEX, FLAGS
Set the hot-spot of a screen to a given coordinate. The hot-spot is the position within the screen where the Screen Offset will have effect and around which rotation will be done
Parameters:
INDEX: The index of the screen (optional)
FLAGS: Flag of bits indicating the horizontal and vertical position of the hot-spot, 0: top or left, 1: center or middle, 2: right or bottom. Example: $11 centers the hot-spot horizontally and vertically

Screen Rotate INDEX, ANGLE
TOTEST! Rotate a screen around it's hot-spot on display. Warning, on software rendererss this instruction will slow down the application
Parameters:
INDEX: The index of the screen
ANGLE: The angle of the rotation of the screen. Default in radian, and degrees after the "Degree" instruction has been used

Screen Rotate ANGLE
TOTEST! Rotate the current screen around it's hot-spot on display. Does not affect the content of the screen. Warning, on software rendererss this instruction will slow down the application

Parameters:

ANGLE: The angle of the rotation of the screen. Default in radian, and degrees after the "Degree" instruction has been used

Screen Skew INDEX, XSKEW, YSKEW

Applies a horizontal and vertical distortion to a screen during the display process

Parameters:

INDEX: The index of the screen

XSKEW: The number of pixel to shift at each horizontal pixel

YSKEW: The number of pixel to shift at each vertical pixel

Screen Skew XSKEW, YSKEW

Applies a horizontal and vertical distortion to the current screen during the display process. Does not affect the content of the screen

Parameters:

XSKEW: The number of pixel to shift at each horizontal pixel

YSKEW: The number of pixel to shift at each vertical pixel

Screen Scale INDEX, XSCALE, YSCALE

Resize a screen during the display process. Does not affect the content of the screen

Parameters:

INDEX: The index of the screen

XSCALE: The horizontal scale. 1= no effect, 0.5= half the width, 2= twice the width, etc.

YSCALE: The vertical scale. 1= no effect, 0.5= half the height, 2= twice the height, etc.

Screen Scale XSCALE, YSCALE

Resize the current screen during the display process. Does not affect the content of the screen

Parameters:

XSCALE: The horizontal scale. 1= no effect, 0.5= half the width, 2= twice the width, etc.

YSCALE: The vertical scale. 1= no effect, 0.5= half the height, 2= twice the height, etc.

Screen Copy SOURCEINDEX, X1, Y1, X2, Y2, DESTINATIONINDEX, X3, Y3, X4, Y4, MODE

Copy an area from one screen to another or itself, resizing the area

Parameters:

SOURCEINDEX: The index of the source screen

X1: The horizontal coordinate of the top-left corner of the origin area to copy

Y1: The vertical coordinate of the top-left corner of the origin area to copy

X2: The horizontal coordinate of the bottom-right corner of the origin area to copy

Y2: The vertical coordinate of the bottom-right corner of the origin area to copy

DESTINATIONINDEX: The index of the destination screen

X3: The horizontal coordinate of the top-left corner of the destination area

Y3: The vertical coordinate of the top-left corner of the destination area

X4: The horizontal coordinate of the bottom-right corner of the destination area

Y4: The vertical coordinate of the bottom-right corner of the destination area

MODE: TODO! The drawing mode to use while drawing

Screen Copy SOURCEINDEX, X1, Y1, X2, Y2, DESTINATIONINDEX, X3, Y3, MODE

Copy an area from one screen to another or itself, preserving the proportion of the area

Parameters:

SOURCEINDEX: The index of the source screen

X1: The horizontal coordinate of the top-left corner of the origin area to copy

Y1: The vertical coordinate of the top-left corner of the origin area to copy

X2: The horizontal coordinate of the bottom-right corner of the origin area to copy
Y2: The vertical coordinate of the bottom-right corner of the origin area to copy
DESTINATIONINDEX: The index of the destination screen
X3: The horizontal coordinate of the top-left corner of the destination area
Y3: The vertical coordinate of the top-left corner of the destination area
MODE: TODO! The drawing mode to use while drawing

Screen Copy SOURCEINDEX, DESTINATIONINDEX, MODE
Copy one screen to another
Parameters:
SOURCEINDEX: The index of the source screen
DESTINATIONINDEX: The index of the destination screen
MODE: TODO! The drawing mode to use while drawing

Screen Copy SOURCEINDEX, SX, SY, SWIDTH, SHEIGHT, DESTINATIONINDEX, DX, DY, DWIDTH, DHEIGHT, MODE
TOTEST! Copy an area from one screen to another or itself, resizing the area
Parameters:
SOURCEINDEX: The index of the source screen
SX: The horizontal coordinate of the top-left corner of the origin area to copy
SY: The vertical coordinate of the top-left corner of the origin area to copy
SWIDTH: The width of the source area
SHEIGHT: The height of the source area
DESTINATIONINDEX: The index of the destination screen
DX: The horizontal coordinate of the top-left corner of the destination area
DY: The vertical coordinate of the top-left corner of the destination area
DWIDTH: The width of the ddestination area
DHEIGHT: the height of the destination area
MODE: TODO! The drawing mode to use while drawing

Screen Copy SOURCEINDEX, SX, SY, SWIDTH, SHEIGHT, DESTINATIONINDEX, DX, DY, MODE
TOTEST! Copy an area from one screen to another or itself, preserving the size of the area
Parameters:
SOURCEINDEX: The index of the source screen
SX: The horizontal coordinate of the top-left corner of the origin area to copy
SY: The vertical coordinate of the top-left corner of the origin area to copy
SWIDTH: The horizontal coordinate of the bottom-right corner of the origin area to copy
SHEIGHT: The vertical coordinate of the bottom-right corner of the origin area to copy
DESTINATIONINDEX: The index of the destination screen
DX: The horizontal coordinate of the top-left corner of the destination area
DY: The vertical coordinate of the top-left corner of the destination area
MODE: TODO! The drawing mode to use while drawing

Screen Project SOURCE, X1, Y1, X2, Y2, DESTINATION, DX1, DY1, DX2, DY2, DX3, DY3, DX4, DY4
Description of the instruction.
Parameters:
SOURCE: Description of the parameter.
X1: Description of the parameter.
Y1: Description of the parameter.
X2: Description of the parameter.
Y2: Description of the parameter.
DESTINATION: Description of the parameter.

DX1: Description of the parameter.
DY1: Description of the parameter.
DX2: Description of the parameter.
DY2: Description of the parameter.
DX3: Description of the parameter.
DY3: Description of the parameter.
DX4: Description of the parameter.
DY4: Description of the parameter.

X Screen INDEX, X

Convert a hardware horizontal coordinate into a screen coordinate. Hardware coordinates are different from screen coordinate only in retro-machine display emulation (Amiga, Atari etc.)

Parameters:

INDEX: The index of the screen
X: The hardware coordinate to convert

Value returned:

integer: The correponding horizontal coordinate in the Screen

X Screen INDEX, X

Convert a hardware horizontal coordinate into a coordinate in the current active screen. Hardware coordinates are different from screen coordinate only in retro-machine display emulation (Amiga, Atari etc.)

Parameters:

INDEX: The index of the screen
X: The hardware coordinate to convert

Value returned:

integer: The correponding horizontal coordinate in the current screen

Y Screen INDEX, Y

Convert a hardware vertical coordinate into a screen coordinate. Hardware coordinates are different from screen coordinate only in retro-machine display emulation (Amiga, Atari etc.)

Parameters:

INDEX: The index of the screen
Y: The hardware coordinate to convert

Value returned:

integer: The correponding vertical coordinate in the given screen

Y Screen INDEX, Y

Convert a hardware vertical coordinate into a coordinate in the current active screen. Hardware coordinates are different from screen coordinate only in retro-machine display emulation (Amiga, Atari etc.)

Parameters:

INDEX: The index of the screen
Y: The hardware coordinate to convert

Value returned:

integer: The correponding vertical coordinate in the current screen

Def Scroll To INDEX, X1, Y1, X2, Y2, DX, DY

Define a new scrolling zone in the current screen. This instruction has no visible effect until a "Scroll" instruction is used

Parameters:

INDEX: The index of the scrolling zone to define
X1: The horizontal coordinate of the top-left pixel of the rectangle to scroll
Y1: The vertical coordinate of the top-left pixel of the rectangle to scroll
X2: The horizontal coordinate of the bottom-right pixel of the rectangle to scroll

Y2: The vertical coordinate of the bottom-right pixel of the rectangle to scroll
DX: Signed horizontal shift to apply. 0= no scroll, -1= one pixel to the left, 1= one pixel to the right, etc
DY: Signed vertical shift to apply. 0= no scroll, -1= one pixel to the top, 1= one pixel to the bottom, etc

Def Scroll INDEX, X, Y, WIDTH, HEIGHT, DX, DY
Define a new scrolling zone in the current screen. This instruction has no visible effect until a "Scroll" instruction is used
Parameters:
INDEX: The index of the scrolling zone to define
X: The horizontal coordinate of the top-left pixel of the rectangle to scroll
Y: The vertical coordinate of the top-left pixel of the rectangle to scroll
WIDTH: The width in pixels of the rectangle to scroll
HEIGHT: The height in pixels of the rectangle to scroll
DX: Signed horizontal shift to apply. 0= no scroll, -1= one pixel to the left, 1= one pixel to the right, etc
DY: Signed vertical shift to apply. 0= no scroll, -1= one pixel to the top, 1= one pixel to the bottom, etc

Scrollt INDEX
Performs the action of scrolling for a pre-defined scrolling area. Moves the pixels in the desired direction. The empty zones at the extremities of the scrollign area, on left, top and/or right and bottom are left unchanged and will have to be cleared
Parameters:
INDEX: The index of the scrolling area as defined with "Def Scroll"

Dual Playfield SCREEN1, SCREEN2
Associate the display of two screens into a parallax display. (Deprecated, use "Set Transparent" and "Screen Offset" to associate more than two screens together)
Parameters:
SCREEN1: The index of first screen, will be on top
SCREEN2: The index of first screen, will be in the back

Dual Priority SCREEN1, SCREEN2
Set the display priority of two screens associate in Dual Playfield. (Deprecated, use "Set Transparent" and "Screen Offset" to associate more than two screens together, and then "Screen To Front" or "Screen To Back" to handle the display priority)
Parameters:
SCREEN1: The index of first screen, will be on top
SCREEN2: The index of first screen, will be in the back

Phybase INDEX
Return the address of the first pixel in the bitmap buffer of the current screen (deprecated, there is no difference between logical and physical screen buffers in AOZ)
Parameters:
INDEX: The index of the screen
Value returned:
integer: A magical number representing the adress of the buffer, to be used later with "Poke" / "Doke" / "Loke" / "Peek" / "Deek" / "Leek" instructions. Not a real address in AOZ

Logbase INDEX
Return the address of the first pixel in the bitmap buffer of the current screen (deprecated, there is no difference between logical and physical screen buffers in AOZ)
Parameters:
INDEX: The index of the screen
Value returned:

integer: A magical number representing the adress of the buffer, to be used later with "Poke" / "Doke" / "Loke" / "Peek" / "Deek" / "Leek" instructions. Not a real address in AOZ

Physic INDEX

TODO! Return a magical number representing the physical buffer of the current screen, to be used in "Screen Copy" and all Screen instructions that necessitate a Screen Index. (deprecated, no such thing as physical or logic al screen in AOZ)

Parameters:

INDEX: The index of the screen

Value returned:

integer: A magical number representing the physical buffer of the screen. To be used wherever you need a screen index

Logic INDEX

TODO! Return a magical number representing the logical buffer of the current screen, to be used in "Screen Copy" and all Screen instructions that necessitate a Screen Index. (deprecated, no such thing as physical or logic al screen in AOZ)

Parameters:

INDEX: The index of the screen

Value returned:

integer: A magical number representing the logical buffer of the screen. To be used wherever you need a screen index

Autoback INDEX, MODE

Set the autoback background preservation system for graphical instruction (Deprecated, has no effect in AOZ)

Parameters:

INDEX: The index of the screen

MODE: A number from 0 to 2 included representing the mode to use

Appear SOURCE, DESTINATION, PIXELS, RANGE

TODO! Progressively draw one screen into another using a fading effect

Parameters:

SOURCE: Index of the source screen

DESTINATION: Index of the destination screen

PIXELS: Value used to perform the effect

RANGE: Range of the apparition in number of pixels (optional)

Zoom SOURCE, X1, Y1, X2, Y2, DESTINATION, X3, Y3, X4, Y4

TODO! Copy and scale a rectangle from one screen to another.

Parameters:

SOURCE: Index of the source screen

X1: Horizontal coordinate of the top-left corner of the rectangle to zoom in the source screen

Y1: Vertical coordinate of the top-left corner of the rectangle to zoom in the source screen

X2: Horizontal coordinate of the bottom-right corner of the rectangle to zoom in the source screen

Y2: Vertical coordinate of the bottom-right corner of the rectangle to zoom in the source screen

DESTINATION: Index of the destination screen

X3: Horizontal coordinate of the top-left corner of the rectangle to zoom in the destination screen

Y3: Vertical coordinate of the top-left corner of the rectangle to zoom in the destination screen

X4: Horizontal coordinate of the bottom-right corner of the rectangle in the destination screen

Y4: Vertical coordinate of the bottom-right corner of the rectangle in the destination screen

Zoom SOURCE, SX, SY, SWIDTH, SHEIGHT, DESTINATION, DX, DY, DWIDTH, DHEIGHT

TODO! Copy and scale a rectangle from one screen to another.

Parameters:

SOURCE: Index of the source screen
SX: Horizontal coordinate of the top-left corner of the rectangle to zoom in the source screen
SY: Vertical coordinate of the top-left corner of the rectangle to zoom in the source screen
SWIDTH: Width of the rectangle to zoom in the source screen
SHEIGHT: Height of the rectangle to zoom in the source screen
DESTINATION: Index of the destination screen
DX: Horizontal coordinate of the top-left corner of the rectangle in the destination screen
DY: Vertical coordinate of the top-left corner of the rectangle in the destination screen
DWIDTH: Width of the rectangle to zoom in the destination screen
DHEIGHT: Height of the rectangle to zoom in the destination screen

XGr

Return the current horizontal coordinate of the graphical cursor in the current screen

XGr

Return the current vertical coordinate of the graphical cursor in the current screen

Reserve Zone NUMBER

Reserve memory to store graphical detection zones in the current screen (deprecated: number of zonme is unlimited in AOZ)

Parameters:

NUMBER: (Optional) The number of zones to allocate. If ommited all zones will be erased (not deprecated without parameter)

ScIn INDEX, X, Y

Check if the given hardware coordinates are located above a given screen on display. Hardware coordinate are only different from screen coordinate for retro-machine emulation (Amiga, Atari etc.)

Parameters:

INDEX: The index of the screen to test
X: Horizontal hardware coordinate to test
Y: Vertical hardware coordinate to test

Value returned:

boolean: True if the given coordinate lay over the screen, False if not

ScIn X, Y

Find the top-most screen on display located under the given hardware coordinates. Hardware coordinate are only different from screen coordinate for retro-machine emulation (Amiga, Atari etc.)

Parameters:

X: Horizontal hardware coordinate to test
Y: Vertical hardware coordinate to test

Value returned:

integer: -1 if the coordinate lay outside of all the screen, or the index of the top-most screen if they lay inside of one

Mouse Screen

Return the index of the screen under the mouse

Value returned:

integer: -1 if the coordinate lay outside of all the displayed screens, or the index of the top-most screen if they lay inside of one

Screen Colour INDEX

Return the number of colours in the palette of a given screen
Parameters:
INDEX: The index of the screen to query
Value returned:
integer: The number of colours in the palette

Screen Colour
Return the number of colours in the palette of the current screen
Value returned:
integer: The number of colours in the palette of the curren screen

Screen Base
Not implemented, deprecated, will return 0
Value returned:
integer: Return 0

Screen Width INDEX
Return the width in pixels of a given screen
Parameters:
INDEX: The index of the screen to query
Value returned:
integer: The width of the screen in pixel

Screen Width
Return the width in pixels of the current screen
Value returned:
integer: The width of the current screen in pixel

Screen Height INDEX
Return the height in pixels of a given screen
Parameters:
INDEX: The index of the screen to query
Value returned:
integer: The height of the screen in pixel

Screen Height
Return the height in pixels of the current screen
Value returned:
integer: The height of the current screen in pixel

Screen INDEX
Set the given screen index as the current screen, all graphical operation being directed to this screen after this instruction
Parameters:
INDEX: The index of the screen

Screen
Return the index of the current screen, -1 if no screen is opened when the function is called
Value returned:
integer: The index of the current screen

Hires
Return a magical number to be used in the "Screen Open" instruction, enforce a horizontal compression by half of the pixels
Value returned:
integer: 1

Lowres
Return a magical number to be used in the "Screen Open" instruction, display pixel with their original horizontal ratio
Value returned:
integer: 0

Laced
Return a magical number to be used in the "Screen Open" instruction, enforce a vertical compression by half of the pixels
Value returned:
integer: 2

Halfbright
Return a magical number to be used in the "Screen Open" instruction, and set the screen in Amiga-compatible Halfbright colour mode
Value returned:
integer: 4

X Hard
TOTEST! Converts a horizontal coordinate in a given screen to its equivalent in hardware coordinates taking into account the position, scale and rotation factor of the screen
Value returned:
integer: The hardware equivalent of the horizontal screen coordinate

X Hard
TOTEST! Converts a horizontal coordinate in the current screen to its equivalent in hardware coordinates taking into account the position, scale and rotation factor of the screen
Value returned:
integer: The hardware equivalent of the horizontal current screen coordinate

Screen Alpha NUMBER
Description of the function.
Parameters:
NUMBER: Description of the parameter.
Value returned:
integer: Description of the value returned.

Remap SCOLOR, DCOLOR, X1, Y1, X2, Y2
Transforms the values of all the pixels in a rectangle matching a specific RGB value to another RGBA value. Warning, this operation can take a long time to process and make the browser irresponsive during a while
Parameters:
SCOLOR: The RGB value of the color to look for
DCOLOR: The RGBA value of the color to replace with
X1: The horizontal coordinate of the top-left corner of the origin area to scan
Y1: The vertical coordinate of the top-left corner of the origin area to scan
X2: The horizontal coordinate of the bottom-right corner of the origin area to scan

Y2: The vertical coordinate of the bottom-right corner of the origin area to scan

Remap SCOLOR, DCOLOR, X, Y, WIDTH, HEIGHT

Transforms the values of all the pixels in a rectangle matching a specific RGB value to another RGBA value. Warning, this operation can take a long time to process and make the browser irresponsive during a while

Parameters:

SCOLOR: The RGB value of the color to look for

DCOLOR: The RGBA value of the color to replace with

X: The horizontal coordinate of the top-left corner of the origin area to scan

Y: The vertical coordinate of the top-left corner of the origin area to scan

WIDTH: The horizontal coordinate of the bottom-right corner of the origin area to scan

HEIGHT: The vertical coordinate of the bottom-right corner of the origin area to scan

Remap SCOLOR, DCOLOR

Transforms the values of all the pixels in the current screen matching a specific RGB value to another RGBA value. Warning, this operation can take a long time to process and make the browser irresponsive during a while

Parameters:

SCOLOR: The RGB value of the color to look for

DCOLOR: The RGBA value of the color to replace with

Screen Mode

Return the "mode" parameter of a screen, as used in the "Screen Open" instruction

Value returned:

integer: integer:The magical number generated by the combination of Lowres, Hires, Laced or Halfbright

Screen Hot Spot X, Y

Set the hot-spot of the current screen. The pixel displayed at the coordinates set by "Screen Offset" will be located at these coordinates inside of the screen, and the screen will be shifted on the display

Parameters:

X: The horizontal coordinate of the hot-spot

Y: The vertical coordinate of the hot-spot

Screen Hot Spot INDEX, X, Y

Set the hot-spot of the given screen. The pixel displayed at the coordinates set by "Screen Offset" will be located at these coordinates inside of the screen, and the screen will be shifted on the display

Parameters:

INDEX: The indexd of the screen to modify

X: The horizontal coordinate of the hot-spot

Y: The vertical coordinate of the hot-spot

Screen Hot Spot FLAGS

Calculates the hot-spot of the current screen

Parameters:

FLAGS: Flag of bits indicating the horizontal and vertical position of the hot-spot, 0: top or left, 1: center or middle, 2: right or bottom. Example: $11 centers the hot-spot horizontally and vertically

Double Buffer

Turns the display system into double-buffering. Deprecated: Double Buffering has no effect in AOZ

Simplified Commands

Some commands to simplify the coding.

Set Display WIDTH, HEIGHT
Set the resolution of the display in pixels.
Parameters:
WIDTH: Width of the display in pixel.
HEIGHT: Height of the display in pixel.

Draw Image NUMBER, X, Y
Draw an image loaded with "Load Asset" command, in the current screen
Parameters:
NUMBER: Index of Image.
X: Position X of the canvas in pixel.
Y: Position Y of the canvas in pixel.

Draw Image NUMBER, X, Y, WIDTH, HEIGHT
Draw an image loaded with "Load Asset" command, in the current screen, with a specific size
Parameters:
NUMBER: Index of Image.
X: Position X on the current Screen in pixel.
Y: Position Y on the current Screen in pixel.
WIDTH: Width on the current Screen in pixel.
HEIGHT: Height on the current Screen in pixel.

Draw Image NAME$, X, Y
Description of the instruction.
Parameters:
NAME$: Description of the parameter.
X: Description of the parameter.
Y: Description of the parameter.

Draw Image NAME$, X, Y, WIDTH, HEIGHT
Description of the instruction.
Parameters:
NAME$: Description of the parameter.
X: Description of the parameter.
Y: Description of the parameter.
WIDTH: Description of the parameter.
HEIGHT: Description of the parameter.

Audio Loop On NUMBER
Repeat the playing at the end of audio.
Parameters:
NUMBER: Index of audio.

Audio Loop On NAME$
Description of the instruction.

Parameters:
NAME$: Description of the parameter.

Audio Loop Off NUMBER
Stop the repeat the playing at the end of audio.
Parameters:
NUMBER: Index of audio.

Audio Loop Off NAME$
Description of the instruction.
Parameters:
NAME$: Description of the parameter.

Play Audio NUMBER
Play an audio file loaded with the "Load Asset" command.
Parameters:
NUMBER: Index of audio.

Play Audio NAME$
Description of the instruction.
Parameters:
NAME$: Description of the parameter.

Stop Audio NUMBER
Stop an audio loaded with the "Load Asset" command.
Parameters:
NUMBER: Index of audio.

Stop Audio NAME$
Description of the instruction.
Parameters:
NAME$: Description of the parameter.

Pause Audio NUMBER
Pause an audio loaded with the "Load Asset" command.
Parameters:
NUMBER: Index of audio.

Pause Audio NAME$
Description of the instruction.
Parameters:
NAME$: Description of the parameter.

Volume Audio NUMBER, VOLUME
Set the volume of an audio loaded with the "Load Asset" command.
Parameters:
NUMBER: Index of audio.
VOLUME: Value of the volume between 0(mute)-100(full)

Volume Audio NAME$, VOLUME
Description of the instruction.

Parameters:
NAME$: Description of the parameter.
VOLUME: Description of the parameter.

Time Audio NUMBER, TIME
Set the position of an audio loaded with the "Load Asset" command.
Parameters:
NUMBER: Index of audio.
TIME: Value of the timer in seconds

Time Audio NAME$, TIME
Description of the instruction.
Parameters:
NAME$: Description of the parameter.
TIME: Description of the parameter.

Time Audio NUMBER
Return the position of audio
Parameters:
NUMBER: Index of audio.
Value returned:
number: The position of audio in seconds.

Time Audio NAME$
Description of the function.
Parameters:
NAME$: Description of the parameter.
Value returned:
integer: Description of the value returned.

Video Loop On NUMBER
Repeat the playing at the end of video loaded with the "Load Asset" command.
Parameters:
NUMBER: Index of video.

Video Loop On NAME$
Description of the instruction.
Parameters:
NAME$: Description of the parameter.

Video Loop Off NUMBER
Stop Repeat the playing at the end of video
Parameters:
NUMBER: Index of video.

Video Loop Off NAME$
Description of the instruction.
Parameters:
NAME$: Description of the parameter.

Play Video NUMBER

Play a video loaded with the "Load Asset" command
Parameters:
NUMBER: Index of video.

Play Video NAME$
Description of the instruction.
Parameters:
NAME$: Description of the parameter.

Stop Video NUMBER
Stop a video loaded by the "Load Asset" command.
Parameters:
NUMBER: Index of video.

Stop Video NAME$
Description of the instruction.
Parameters:
NAME$: Description of the parameter.

Pause Video NUMBER
Pause a video loaded with the "Load Asset" command.
Parameters:
NUMBER: Index of video.

Pause Video NAME$
Description of the instruction.
Parameters:
NAME$: Description of the parameter.

Volume Video NUMBER, VOLUME
Set the video volume
Parameters:
NUMBER: Index of video.
VOLUME: Value of the volume between 0(mute)-100(full)

Volume Video NAME$, VOLUME
Description of the instruction.
Parameters:
NAME$: Description of the parameter.
VOLUME: Description of the parameter.

Time Video NUMBER, TIME
Set the position of video in seconds
Parameters:
NUMBER: Index of video.
TIME: Value of the timer in seconds

Time Video NAME$, TIME
Description of the instruction.
Parameters:
NAME$: Description of the parameter.

TIME: Description of the parameter.

Time Video NUMBER
Return the position of video in seconds
Parameters:
NUMBER: Index of video.
Value returned:
number: The position of video in seconds.

Time Video NAME$, TIME
Description of the function.
Parameters:
NAME$: Description of the parameter.
TIME: Description of the parameter.
Value returned:
integer: Description of the value returned.

Draw Video NUMBER, X, Y
Draw a video loaded with the "Load Asset" command on the current screen
Parameters:
NUMBER: Index of video.
X: Position X of the video in pixel.
Y: Position Y of the video in pixel.

Draw Video NUMBER, X, Y, WIDTH, HEIGHT
Draw a video loaded with the "Load Asset" command on the current screen with a specific size
Parameters:
NUMBER: Index of video.
X: Position X of the video in pixel.
Y: Position Y of the video in pixel.
WIDTH: Width of the video in pixel.
HEIGHT: Height of the video in pixel.

Draw Video NAME$, X, Y
Description of the instruction.
Parameters:
NAME$: Description of the parameter.
X: Description of the parameter.
Y: Description of the parameter.

Draw Video NAME$, X, Y, WIDTH, HEIGHT
Description of the instruction.
Parameters:
NAME$: Description of the parameter.
X: Description of the parameter.
Y: Description of the parameter.
WIDTH: Description of the parameter.
HEIGHT: Description of the parameter.

Video Bob VIDEON, BOBN
Assign a video loaded with the "Load Asset" command as a Bob Image

Parameters:
VIDEON: Index of video.
BOBN: Index of Bob image.

Video Bob VIDEON, BOBN, WIDTH, HEIGHT
Assign a video loaded with the "Load Asset" command as a Bob Image width a specific size
Parameters:
VIDEON: Index of video.
BOBN: Index of Bob image.
WIDTH: Width of the used image.
HEIGHT: Height of the used image.

Video Bob VIDEON, BOBN, WIDTH, HEIGHT, REDVALUE, GREENVALUE, BLUEVALUE
Assign a video loaded with the "Load Asset" command as a Bob Image with Chroma key on a color
Parameters:
VIDEON: Index of video.
BOBN: Index of Bob image.
WIDTH: Width of the used image.
HEIGHT: Height of the used image.
REDVALUE: Value of Red component of the color transparent (0-255)
GREENVALUE: Value of Green component of the color transparent (0-255)
BLUEVALUE: Value of Blue component of the color transparent (0-255)

Video Bob VIDEON, BOBNAME$
Description of the instruction.
Parameters:
VIDEON: Description of the parameter.
BOBNAME$: Description of the parameter.

Video Bob VIDEON, BOBNAME$, WIDTH, HEIGHT
Description of the instruction.
Parameters:
VIDEON: Description of the parameter.
BOBNAME$: Description of the parameter.
WIDTH: Description of the parameter.
HEIGHT: Description of the parameter.

Video Bob VIDEON, BOBNAME$, WIDTH, HEIGHT, REDVALUE, GREENVALUE, BLUEVALUE
Description of the instruction.
Parameters:
VIDEON: Description of the parameter.
BOBNAME$: Description of the parameter.
WIDTH: Description of the parameter.
HEIGHT: Description of the parameter.
REDVALUE: Description of the parameter.
GREENVALUE: Description of the parameter.
BLUEVALUE: Description of the parameter.

Video Bob VIDEO$, BOBN
Description of the instruction.
Parameters:
VIDEO$: Description of the parameter.

BOBN: Description of the parameter.

Video Bob VIDEO$, BOBN, WIDTH, HEIGHT
Description of the instruction.
Parameters:
VIDEO$: Description of the parameter.
BOBN: Description of the parameter.
WIDTH: Description of the parameter.
HEIGHT: Description of the parameter.

Video Bob VIDEO$, BOBN, WIDTH, HEIGHT, REDVALUE, GREENVALUE, BLUEVALUE
Description of the instruction.
Parameters:
VIDEO$: Description of the parameter.
BOBN: Description of the parameter.
WIDTH: Description of the parameter.
HEIGHT: Description of the parameter.
REDVALUE: Description of the parameter.
GREENVALUE: Description of the parameter.
BLUEVALUE: Description of the parameter.

Video Bob VIDEO$, BOBNAME$
Description of the instruction.
Parameters:
VIDEO$: Description of the parameter.
BOBNAME$: Description of the parameter.

Video Bob VIDEO$, BOBNAME$, WIDTH, HEIGHT
Description of the instruction.
Parameters:
VIDEO$: Description of the parameter.
BOBNAME$: Description of the parameter.
WIDTH: Description of the parameter.
HEIGHT: Description of the parameter.

Video Bob VIDEO$, BOBNAME$, WIDTH, HEIGHT, REDVALUE, GREENVALUE, BLUEVALUE
Description of the instruction.
Parameters:
VIDEO$: Description of the parameter.
BOBNAME$: Description of the parameter.
WIDTH: Description of the parameter.
HEIGHT: Description of the parameter.
REDVALUE: Description of the parameter.
GREENVALUE: Description of the parameter.
BLUEVALUE: Description of the parameter.

Show Video VIDEON
Description of the instruction.
Parameters:
VIDEON: Description of the parameter.

Hide Video VIDEON

Description of the instruction.

Parameters:

VIDEON: Description of the parameter.

Speech Recognition System

Speak and intercept your words to interact with your AOZ Program

Speech Recognition Start
Enable the Speech recognition System

Speech Recognition Stop
Disable the Speech recognition System

Speech Recognition Reset
Reset the Speech recognition System

Speech Recognition Value$
Retrieves the word or phrase spoken by the user as a string.
Value returned:
string: the word or phrase spoken by the user.

Speech Recognition Add Word WORD$
Description of the instruction.
Parameters:
WORD$: Description of the parameter.

Speech Voices Count
Description of the function.
Value returned:
integer: Description of the value returned.

Speech Voice INDEX
Description of the function.
Parameters:
INDEX: Description of the parameter.
Value returned:
integer: Description of the value returned.

Speech Lang LANG$
Description of the instruction.
Parameters:
LANG$: Description of the parameter.

Speech Voice VOICE
Description of the instruction.
Parameters:
VOICE: Description of the parameter.

Speech Pitch PITCH#
Description of the instruction.
Parameters:
PITCH#: Description of the parameter.

Speech Rate RATE#
Description of the instruction.
Parameters:
RATE#: Description of the parameter.

Speech TEXT$
Description of the instruction.
Parameters:
TEXT$: Description of the parameter.

Sprite Commands

Instructions and functions to display moveable objects on the top of the display (Sprites)

Sprite
Description of the instruction.

Sprite
Description of the instruction.

Set Sprite Buffer NUMBER
Description of the instruction.
Parameters:
NUMBER: Description of the parameter.

Sprite Show
Description of the instruction.

Sprite Hide
Description of the instruction.

Sprite Off
Description of the instruction.

Sprite Priority
Description of the instruction.

Sprite Update Off
Turns off the automatic sprite coordinate update system. After it, all "Sprite" instruction will no longer have a visible effect until an "Sprite Update" instruction is used

Sprite Update On
Turns on the automatic sprite coordinate update system. After it, the effect of all "Sprite" instructions will be visible on display

Sprite Update
Enforce a refresh of all the sprites on screen: all the modification to the coordinates and images of the sprite are reflected immediately on the screen

Sprite Scale
Description of the instruction.

Sprite Rotate
Description of the instruction.

Sprite Skew
Description of the instruction.

X Sprite
Description of the function.
Value returned:
integer: Description of the value returned.

Y Sprite
Description of the function.
Value returned:
integer: Description of the value returned.

I Sprite
Description of the function.
Value returned:
integer: Description of the value returned.

I Sprite$
Description of the function.
Value returned:
string: Description of the value returned.

Sprite Base
Description of the function.
Value returned:
integer: Description of the value returned.

Issprite
Description of the function.
Value returned:
integer: Description of the value returned.

Get Sprite
Description of the instruction.

Get Sprite
Description of the instruction.

Get Sprite
Description of the instruction.

Get Sprite
Description of the instruction.

Get Sprite Palette
Description of the instruction.

Sprite Add
Description of the instruction.

Sprite Move
Description of the instruction.

Sprite Move X
Description of the instruction.

Sprite Move Y
Description of the instruction.

Sprite Move Off
Description of the instruction.

Sprite Move Off
Description of the instruction.

Sprite Move On
Description of the instruction.

Sprite Move On
Description of the instruction.

Sprite Move Freeze
Description of the instruction.

Sprite Move Freeze
Description of the instruction.

Sprite Movon
Description of the function.
Value returned:
integer: Description of the value returned.

Sprite Movon
Description of the function.
Value returned:
integer: Description of the value returned.

Sprite Anim
Description of the instruction.

Sprite Anim Off
Description of the instruction.

Sprite Anim Off
Description of the instruction.

Sprite Anim On
Description of the instruction.

Sprite Anim On
Description of the instruction.

Sprite Anim Freeze
Description of the instruction.

Sprite Anim Freeze
Description of the instruction.

String of characters

Instructions and functions to manipulate the strings of characters.

Asc CHARACTER$
Converts the first character of a string into it's ASCII code
Parameters:
CHARACTER$: The character to convert
Value returned:
integer: The equivalent in ASCII

Flip$ STRING$
Invert a string
Parameters:
STRING$: The string
Value returned:
string: The inverted string

Chr$ CODE
Return the character with a given ASCII code
Parameters:
CODE: The ascii code to convert into a string
Value returned:
string: The string containing the representation of the ASCII code

Space$ LENGTH
Return a string contain the demanded amount of spaces
Parameters:
LENGTH: The number of spaces desired
Value returned:
string: A string with only spaces

String$ STRING$, NUMBER
Create a new string from the first character of an existing string
Parameters:
STRING$: The string to repeat
NUMBER: The number of time to repeat the first character
Value returned:
string: A string with the source first character repeated a number of times

Upper$ STRING$
Convert a string of text to upper-case
Parameters:
STRING$: The string to convert
Value returned:
string: The upper-case version of the string

Lower$ STRING$
Convert a string of text to upper-case

Parameters:
STRING$: The string to convert
Value returned:
string: The lower-case version of the string

Str$ NUMBER
Convert a number into a string
Parameters:
NUMBER: The number to convert
Value returned:
string: The text version of the number, respecting the rules et by teh "Fix" instruction

Val STRING$
Convert a string into a number
Parameters:
STRING$: The string to evaluate
Value returned:
number: The numerical version of the string. Returns zero if the converstion fails (TODO! Report error)

Bin$ NUMBER, DIGITS
Convert a decimal value into a string of binary digits
Parameters:
NUMBER: The number to convert
DIGITS: The number of digits, optional
Value returned:
string: A string in the form of %0000... representing the binary version of the number

Hex$ NUMBER, DIGITS
Convert a decimal value into a string of hexadecimal digits
Parameters:
NUMBER: The number to convert
DIGITS: The number of digits, optional
Value returned:
string: A string in the form of %0000... representing the binary version of the number

Len STRING
Return the length in characters of a string
Parameters:
STRING: The number to convert
Value returned:
integer: The number oof characters in the string

Instr HOST$, GUEST$, START
Search for occurrences of one sub-string within another string
Parameters:
HOST$: The string to search into
GUEST$: The sub-string to search
START: Option first character position to start the search
Value returned:
integer: The position of the sub-string if found, starting at 1, zero if not found (TODO! make a all-indexes-at-zero tag!!!)

Tab$

Return a string containing the Tab character (ASCII: 9) to be used in a "Print" statement

Value returned:

string: A string containing the Tab character

Repeat$ TEXT$, NUMBER

Repeat a string

Parameters:

TEXT$: The string to be repeated

NUMBER: The number of repetitions

Value returned:

string: A string contaning the original string repeated NUMBER of times

System Commands

Instructions and functions to access the system resources. Will be implemented in the future if

Dev Open
Open a device. May be implemented in AOZ, has no effect in current version

Dev Close
Close a device. May be implemented in AOZ, has no effect in current version

Dev Base
Get base address of an IO structure. May be implemented in AOZ later
Value returned:
integer: 0 in this version

Dev Do
Call a command using DoI0. May be implemented in AOZ, has no effect in current version

Dev Send
Call a command using SendIO. May be implemented in AOZ, has no effect in current version

Dev Abort
Abort an IO operation. May be implemented in AOZ, has no effect in current version

Dev Check
Check status of a device with a CheckIO. May be implemented in AOZ, has no effect in current version
Value returned:
integer: 0 in this version

Dev First$
Get the first device from the current device list. May be implemented in AOZ, has no effect in current version
Value returned:
string: "" in this version

Dev Next$
Get the next device from the current device list. May be implemented in AOZ, has no effect in current version
Value returned:
string: "" in this version

Lib Open
Open a library. May be implemented in AOZ (for example for Windows DLLs), has no effect in current version

Lib Close
Close a library. May be implemented in AOZ (for example for Windows DLLs), has no effect in current version

Lib Call
Call a function from the libraryEnd Function. May be implemented in AOZ (for example for Windows DLLs), has no effect in current version

Lib Base

Get the base address of the library. May be implemented in AOZ, has no effect in current version

Value returned:

integer: 0 in this version

Text Window commands

Instructions and functions to print and manage fixed character text output in AOZ screens

Wind Open INDEX, X, Y, WIDTH, HEIGHT, BORDER, TAG$
Open a new text window in the current screen
Parameters:
INDEX: The index of text window to open
X: The horizontal coordinate of the left border of the window in the screen
Y: The vertical coordinate of the left border of the window in the screen
WIDTH: The width in characters of the new text window
HEIGHT: The width in characters of the new text window
BORDER: The number of the border character to use (optional)
TAG$: Unused for the moment

Wind Close
Close the current text window

Wind Save
Save the character content of the the currenttext window so that it can be restored later

Wind Move DX, DY
Move the current text window horizontally and vertically
Parameters:
DX: Signed horizontal displacement
DY: Signed vertical displacement

Wind Size NEWWIDTH, NEWHEIGHT
Resize the current text window horizontally and vertically
Parameters:
NEWWIDTH: New width in number of characters
NEWHEIGHT: New height in number of characters

Window INDEX
Activate the given window in the current screen
Parameters:
INDEX: Index of the window to activate

Windon
Return the index of the current window in the current screen
Value returned:
integer: The index of the current window in the current screen

Locate X, Y
Set the position of the text5 cursor of the current windowin the current screen
Parameters:
X: The new horizontal position of the cursor in text coordinates
Y: The new vertical position of the cursor in text coordinates

Clw PAPER
Description of the instruction.
Parameters:
PAPER: Description of the parameter.

Home
Move the text cursor at position 0, 0. The next text output with a "Print" instructions will be locate on the top-left of the windows

Curs Pen COLORINDEX
Move the text cursor at position 0, 0. The next text output with a "Print" instructions will be locate on the top-left of the windows
Parameters:
COLORINDEX: Index of the colour in teh palette of the screen the text window belongs to

Pen$ COLORINDEX
Return a magical string to use in a "Print" statement and change the colour of the pen used to draw the following texts
Parameters:
COLORINDEX: Index of the colour to use in the palette of the screen
Value returned:
string: A magical string that will be understood by the "Print" command

Paper$ COLORINDEX
Return a magical string to use in a "Print" statement and change the colour of the paper used to draw the following texts
Parameters:
COLORINDEX: Index of the colour to use in the palette of the screen
Value returned:
string: A magical string that will be understood by the "Print" command

At X, Y
Return a magical string to use in a "Print" statement that changes the location of the cursor for the next characters
Parameters:
X: Horizontal text coordinates where to print
Y: Vertical text coordinates where to print
Value returned:
string: A magical string that will be understood by the "Print" command

Pen COLOURINDEX
Change the colour of the pen used to draw characters during a "Print" statement
Parameters:
COLOURINDEX: The index of the colour in the current screen palette

Paper COLOURINDEX
Change the colour of the paper used to draw the background space of the characters during a "Print" statement
Parameters:
COLOURINDEX: The index of the colour in the current screen palette

Pen
Function to return the current text Pen color index.
Value returned:
:

Paper
Function to return the current text Paper color index.
Value returned:
:

Centre TEXT$
Display and centre a string in the middle of the current text window at the current vertical position of the text cursor. This instruction will have unpredicatable result if the string is larger than the width of the text window
Parameters:
TEXT$: The text to centre in the text window

Border BORDERNUMBER
Change teh txet window character border to one of the pre-defined ones
Parameters:
BORDERNUMBER: The number of the pre-defined text window borders to use

Writing STYLE1, STYLE2
TOTEST! Define the logical operation done during the display of characters by the "Print" statement
Parameters:
STYLE1: A Mask containing bits as flags. Bit 0: REPLACE, bit 1: OR, bit 2: XOR, bit 3: AND, bit 4 IGNORE
STYLE2: A Mask containing bits as flags. Bit 0: Normal Print text and background together, bit 1: paper Only the background to be drawn on screen, bit 2: pen Ignore paper colour and print text on background colour zero

Title Top TITLE$
Display a title in the top of the current text window with a border. Will have no effect if the window has no border
Parameters:
TITLE$: The text of the title to display

Title Bottom TITLE$
Display a title in the bottom of the current text window with a border. Will have no effect if the window has no border
Parameters:
TITLE$: The text of the title to display

Curs Off
Hide the text cursor and stop all associated colour animation. Use this instruction before graphical output to speed up the display

Curs On
Show the text cursor and restarts all associated colour animation

Inverse On
Switch On inverse printing in the current text window: Pen will be used instead of Paper and vice-versa

Inverse Off
Switch Off inverse printing in the current text window

Under On
Switch On underlining

Under Off
Switch Off underlining

Shade On
Switch On the display of a shadow associated with text output

Shade Off
Switch Off the display of a shadow associated with text output

Scroll On
Turn ON the automatic scrolling of teh content of the current text window when the cursor pass the bottom line

Scroll Off
Turn OFF the automatic scrolling of teh content of the current text window when the cursor pass the bottom line, enforcing all further printing to be done in the top of the window

CUp$
Return a magical string to be used during a "Print" statement, moving the cursor one line up
Value returned:
string: A magical string understood by the "Print" statement

CDown$
Return a magical string to be used during a "Print" statement, moving the cursor one line down
Value returned:
string: A magical string understood by the "Print" statement

CLeft$
Return a magical string to be used during a "Print" statement, moving the cursor one character left
Value returned:
string: A magical string understood by the "Print" statement

CRight$
Return a magical string to be used during a "Print" statement, moving the cursor one character right
Value returned:
string: A magical string understood by the "Print" statement

CUp
Move the cursor of the current text window one line up. Will force a scrolling of all text down if the cursor is already on the first line and scrolling is enabled

CDown
Move the cursor of the current text window one line down. Will force a scrolling of all text up if the cursor is already on the last line and scrolling is enabled

CDown
Move the cursor of the current text window one character left. Cursor will wrap to the right and up if it was located at position zero, and eventual generate a scrolling down of the content of the window if scrolling is enabled

CRight
Move the cursor of the current text window one character right. Cursor will wrap to the left and down if it was located at

the rightmost position, and eventual generate a scrolling up of the content of the window if scrolling is enabled

Memorize X
Store the horizontal position of the cursor in internal memory so that you can retreive it later

Memorize Y
Store the vertical position of the cursor in internal memory so that you can retreive it later

Remember X
Recall the horizontal position of the cursor from internal memory and move the cursor to the position

Remember Y
Recall the vertical position of the cursor from internal memory and move the cursor to the position

CMove$ DX, DY
Return a magical string to be used during a "Print" statement, moving the cursor by the given character displacements
Parameters:
DX: Signed horizontal displacement
DY: Signed vertical displacement
Value returned:
string: A magical string understood by the "Print" statement

CMove DX, DY
Move the text cursor by a horizontal and/or vertical displacement
Parameters:
DX: Signed horizontal displacement
DY: Signed vertical displacement

CLine LENGTH
Clear the line (paint with the Paper colour) located at teh current vertical position of the cursor. Does not move the cursor
Parameters:
LENGTH: Eventual number of characters to clear (optional)

HScroll TYPE
Scroll the content of the current window or line by one character horizontally
Parameters:
TYPE: Specifies what should scroll: 1 Scroll current line to the left, 2 Scroll entire screen to the left, 3 Scroll current line to the right, 4 Scroll entire screen to the right

VScroll TYPE
Scroll the content of the current window or line by one character vertically
Parameters:
TYPE: Specifies what should scroll: 1 Scroll down text on and below current cursor Line, 2 Scroll down text from top of screen to current cursor line only, 3 Scroll up text from top of screen to current cursor line only, 4 Scroll up text on or below current cursor line

Set Tab WIDTH
Set the width in characters used when printing the TAB character. Default is 4
Parameters:
WIDTH: The number of characters to use for further printing of the TAB characters

Set Curs L1, L2, L3, L4, L5, L6, L7, L8
Define the shape of the text cursor
Parameters:
L1: Byte mask representing the first line
L2: Byte mask representing the second line
L3: Byte mask representing the third line
L4: Byte mask representing the fourth line
L5: Byte mask representing the fifth line
L6: Byte mask representing the sixth line
L7: Byte mask representing the seventh line
L8: Byte mask representing the nineth line

X Curs
Return the horizontal position of the text cursor, in text coordinates
Value returned:
integer: The horizontal coordinate of the text cursor

Y Curs
Return the vertical position of the text cursor, in text coordinates
Value returned:
integer: The vertical coordinate of the text cursor

X Graphic X
Return the graphical horizontal position of a coordinate in the text window coordinate space
Parameters:
X: The horizontal coordinate to convert. (optional, if ommited will return the coordinate of the text cursor)
Value returned:
integer: The horizontal coordinate of the given text coordinate in graphical screen space

Y Graphic X
Return the graphical vertical position of a coordinate in the text window coordinate space
Parameters:
X: The vertical coordinate to convert. (optional, if ommited will return the coordinate of the text cursor)
Value returned:
integer: The vertical coordinate of the given text coordinate in graphical screen space

Border$ TEXT$, BORDER$
Return a magical string understood by the "Print" statement, enforcing the drawing of a border around the given text
Parameters:
TEXT$: The text to display
BORDER$: The number of the border to use
Value returned:
string: A magical string understood by the "Print" statement

X Text X
Convert a horizontal coordinate from the graphical coordinate space of the screen hosting the text window in a text coordinate within the window
Parameters:
X: The horizontal coordinate to convert
Value returned:

integer: The converted coordinate, or -1 if the given coordinate does not lay inside of the text window

Y Text Y

Convert a vertical coordinate from the graphical coordinate space of the screen hosting the text window in a text coordinate within the window

Parameters:

Y: The vertical coordinate to convert

Value returned:

integer: The converted coordinate, or -1 if the given coordinate does not lay inside of the text window

Set Text STYLE

Change the style of a font by selecting one of eight different styles

Parameters:

STYLE: A bit-map indicating the style: Bit 0 Underline, Bit 1 Bold, Bit 2 Italic

Text Styles

Return the index reference of the text style you last selected using "Set Text"

Value returned:

integer: A bit-map in the same format as the one used in the "Set Text" command

HSlider X1, Y1, X2, Y2, UNITS, POSITION, LENGTH

TODO! Draw a horizontal slider bar in the current screen

Parameters:

X1: Horizontal coordinate of the top-left corner of the slider

Y1: Vertical coordinate of the top-left corner of the slider

X2: Horizontal coordinate of the bottom-right corner of the slider

Y2: Vertical coordinate of the bottom-right corner of the slider

UNITS: Number of individual units that the slider is divided into

POSITION: Position of theactive slider box or control button from the left-hand end of the slider,measured in the same units as UNITS

LENGTH: Length of the slider control box in these units in UNITS

VSlider X1, Y1, X2, Y2, UNITS, POSITION, LENGTH

TODO! Draw a vertical slider bar in the current screen

Parameters:

X1: Horizontal coordinate of the top-left corner of the slider

Y1: Vertical coordinate of the top-left corner of the slider

X2: Horizontal coordinate of the bottom-right corner of the slider

Y2: Vertical coordinate of the bottom-right corner of the slider

UNITS: Number of individual units that the slider is divided into

POSITION: Position of theactive slider box or control button from the left-hand end of the slider,measured in the same units as UNITS

LENGTH: Length of the slider control box in these units in UNITS

HSlider X, Y, WIDTH, HEIGHT, UNITS, POSITION, LENGTH

TODO! Draw a horizontal slider bar in the current screen

Parameters:

X: Horizontal coordinate of the top-left corner of the slider

Y: Vertical coordinate of the top-left corner of the slider

WIDTH: Width of the slider

HEIGHT: Height of the slider

UNITS: Number of individual units that the slider is divided into

POSITION: Position of theactive slider box or control button from the left-hand end of the slider,measured in the same units as UNITS

LENGTH: Length of the slider control box in these units in UNITS

VSlider X, Y, WIDTH, HEIGHT, UNITS, POSITION, LENGTH

TODO! Draw a vertical slider bar in the current screen

Parameters:

X: Horizontal coordinate of the top-left corner of the slider

Y: Vertical coordinate of the top-left corner of the slider

WIDTH: Width of the slider

HEIGHT: Height of the slider

UNITS: Number of individual units that the slider is divided into

POSITION: Position of theactive slider box or control button from the left-hand end of the slider,measured in the same units as UNITS

LENGTH: Length of the slider control box in these units in UNITS

Set Slider INK1, PAPER1, OUTLINE1, PATTERN1, INK1, PAPER1, OUTLINE1, PATTERN1

TODO! Set the fill pattern for the slider bar

Parameters:

INK1: Slider bar ink

PAPER1: Slider bar paper

OUTLINE1: Slider bar color of outline

PATTERN1: Slider bar fill pattern

INK1: Control box ink

PAPER1: Control box paper

OUTLINE1: Control box color of outline

PATTERN1: Control box fill pattern

Track and Med musics modules

You can play music and sound effects inside your AOZ programs, using the old MOD and XM file formats.
The main advantage of this type of file is its small size. You can store music on several voices as well as sound effects. The AOZ instructions allow you to load and play your files.
AOZ uses the BassoonTracker Javascript library from Steffest, available under the MIT license.
Github: https://github.com/steffest/BassoonTracker

Track Loop On
Enable the loop playing

Track Loop Off
Disable the loop playing

Track Load FILENAME, BANK
Load a module music file into a bank
Parameters:
FILENAME: The filename of the module to load
BANK: The bank to store the module loaded

Track Play BANK, PATTERN
Play a music module loaded with Track Load instruction
Parameters:
BANK: The bank number containing the loaded module
PATTERN: The starting pattern

Track Stop
Stop the current playing of the music module after a call of Track Play instruction

Track Pause
Pause the current playing of the music module after a call of Track Play instruction

Track Resume
Continue the current playing of the music module after a call of Track Stop instruction

Track Title
Return the title of the current music module.
Value returned:
:

Track Signature
Return the name of the format of the current music module (Protracker or FastTracker).
Value returned:
:

Track Length
Return the length of the current music module in number of patterns.

Value returned:
:

Track Channels
Return the number of channels of the current music module.
Value returned:
:

Track Position
Return the number of the pattern in playing of the current music module.
Value returned:
:

Track Position
Return the position into the pattern in playing of the current music module.
Value returned:
:

Track Volume VOLUME
Set the global volume of the current music module.
Parameters:
VOLUME: Integer value between 0 and 100

Track Pattern PATTERN
Jump to a pattern of the current music module.
Parameters:
PATTERN: Number of pattern

Track Sam Init BANK
Initialize the samples from a bank number. Note: This instruction replace the current music module by one loaded into the bank.
Parameters:
BANK: The bank number containing the loaded module

Track Sam Play SAMPLE
Play a sample stored into the current module.
Parameters:
SAMPLE: The number of sample to play

Track Sam Play SAMPLE, RATE
Play a sample stored into the current module.
Parameters:
SAMPLE: The number of sample to play
RATE: The sample rate to play

Track Sam Play SAMPLE, RATE, VOLUME
Play a sample stored into the current module.
Parameters:
SAMPLE: The number of sample to play
RATE: The sample rate to play
VOLUME: Integer value between 0 and 100

Med Load FILENAME, BANK
Load a module music file into a bank
Parameters:
FILENAME: The filename of the module to load
BANK: The bank to store the module loaded

Med Play BANK
Play a music module loaded with Med Load instruction
Parameters:
BANK: The bank number containing the loaded module

Med Play BANK, MUSIC
Play a music module loaded with Med Load instruction
Parameters:
BANK: The bank number containing the loaded module
MUSIC: The starting music

Med Stop
Stop the current playing of the music module after a call of Med Play instruction

Med Cont
Continue the current playing of the music module after a call of Med Stop instruction

Midi On
This instruction is not implemented. It's just for the compatibility with AMOS

Track Volume VOLUME
Set the global volume of the current music module.
Parameters:
VOLUME: Integer value between 0 and 100

Using CSV Model Data

Instructions and functions to manage the CSV files

Csv Open INDEX, WIDTH, HEIGHT
Open a CSV channel, enabling you to load, create and later save CSV files
Parameters:
INDEX: Index of the channel
WIDTH: Number of columns in the CSV file, will be overwritten when loading a file
HEIGHT: Number of lines in the CSV file, will be overwritten when loading a file

Csv Load INDEX, PATH$, TAG$
Open a CSV file iinto the channel
Parameters:
INDEX: Index of the channel
PATH$: Path to the file. For the moment, the file must be located in the "filesystem" of your application
TAG$: unuse4d, for future expansion

Csv Save INDEX, PATH$, TAG$
TODO! Save the content of the channel into a CSV file
Parameters:
INDEX: Index of the channel
PATH$: Path to the file to create. For the moment, the file will be saved in the local data area of your browser, and will appear in the directory of the application filesystem next time you run it in the browser with the same URL
TAG$: unuse4d, for future expansion

Csv Close INDEX
Destroys a channel and all data associated
Parameters:
INDEX: Index of the channel

Csv Column INDEX, COLUMN, TITLE$
Set the title of a column
Parameters:
INDEX: Index of the channel
COLUMN: Number of the column
TITLE$: New string for title

Csv Set$ INDEX, COLUMN, LINE, TEXT$
Set the content of a cell from a string
Parameters:
INDEX: Index of the channel
COLUMN: Number of the column
LINE: Number of the line
TEXT$: Text to set

Csv Set INDEX, COLUMN, LINE, TEXT$
Set the content of a cell from a number
Parameters:

INDEX: Index of the channel
COLUMN: Number of the column
LINE: Number of the line
TEXT$: Value to set

Csv Erase INDEX
Reset a CSV channel, all data will be erased
Parameters:
INDEX: Index of the channel

Using JSON Model Data

Instructions and functions to manage the JSON files

JSON Open INDEX, TAGS
Open a JSON channel, enabling you to load or create JSON files
Parameters:
INDEX: Index of the channel
TAGS: Unused, for future expansion

JSON Close INDEX
Destroys a JSON channel and all data associated
Parameters:
INDEX: Index of the channel

JSON Load INDEX, PATH$, TAG$
Load a json file into the channel
Parameters:
INDEX: Index of the channel
PATH$: Path to the file
TAG$: Unused, for future expansion

JSON Save INDEX, PATH$, TAG$
Save the resulting stringifies JSON string as a file
Parameters:
INDEX: Index of the channel
PATH$: Path to the file
TAG$: Unused, for future expansion

JSON New INDEX
Reset the JSON channel
Parameters:
INDEX: Index of the channel

JSON Parse INDEX, STRING$
parse a string into JSON
Parameters:
INDEX: Index of the channel
STRING$: Path to the file

JSON Stringify$ INDEX, STRING$
parse a string into JSON
Parameters:
INDEX: Index of the channel
STRING$: Path to the file

JSON Property INDEX, PATH$, DEFAULTVALUE
Get a numeric value from the JSON file
Parameters:

INDEX: Index of the channel
PATH$: Formatted string indicating the path to the property
DEFAULTVALUE: Value to return if the property is not defined. In this case, AOZ will not generate an error.
Value returned:
number: The value from the property

JSON Property$ INDEX, PATH$, DEFAULTVALUE$
Get a string value from the JSON file
Parameters:
INDEX: Index of the channel
PATH$: Formatted string indicating the path to the property
DEFAULTVALUE$: Value to return if the property is not defined. In this case, AOZ will not generate an error.
Value returned:
number: The string from the property

JSON Set Property$ INDEX, PATH$, VALUE$
Set a string value in the JSON file
Parameters:
INDEX: Index of the channel
PATH$: Formatted string indicating the path to the property
VALUE$: Value to set

JSON Set Property INDEX, PATH$, VALUE
Set a numerical value in the JSON file
Parameters:
INDEX: Index of the channel
PATH$: Formatted string indicating the path to the property
VALUE: Value to set

Using of QR Code in an AOZ Program

Create and read QR Code with AOZ.

QRCode Create TEXT$, IMAGE, WIDTH, HEIGHT
Create an QRCode image from a text and assign it to an image number.
Parameters:
TEXT$: Text associated to the QRCode.
IMAGE: Number of the image to use with the BOB instructions
WIDTH: Width of the QRCode Image
HEIGHT: Height of the QRCode Image

QRCode Scan CAMERAID$
Setting up the QR Code reader.
Parameters:
CAMERAID$: One of the following IDs: "user" for the front camera, "environment" for the front camera

QRCode Scan
Starts the QR Code reader.

QRCode Scan
Stops the QR Code reader.

QRCode Scan
Retrieves the text of the QR Code retrieved by the reader.
Value returned:
string: String containing the text of the QR Code retrieved by the reader.

Using the MarkDown files

nstructions and functions to load and display a MD file.

MD Open INDEX, WIDTH, HEIGHT, TAGS
Open a MD channel, enabling you to load and display MD Files or texts
Parameters:
INDEX: Index of the channel
WIDTH: Width in pixels of the display area
HEIGHT: Heigth in pixels of the display area
TAGS: Various tags. #transparent will set the background transparent

MD Load INDEX, PATH$, TAG$
Load a md file into the channel
Parameters:
INDEX: Index of the channel
PATH$: Path to the file
TAG$: unused, for future expansion

MD Display INDEX, X, Y, WIDTH, HEIGHT
Set the display area of the MD channel
Parameters:
INDEX: Index of the channel
X: Horizontal coordinate of the top-left corner of the display area
Y: Vertical coordinate of the top-left corner of the display area
WIDTH: Width of the display area
HEIGHT: Height of the display area

MD Close INDEX
Destroys a MD channel and all data associated
Parameters:
INDEX: Index of the channel

MD Set Fonts FONT1$, FONT2$, HEIGHT, WEIGHT
Set the fonts of the MD channel
Parameters:
FONT1$: Name of the font used to display texts
FONT2$: Name of the font used to display code
HEIGHT: Base height of the fonts (defaults to 15)
WEIGHT: Base weight of the fonts (default to 400)

Variables Commands

Instructions and functions to manage variables and arrays in AOZ

True
Constant: boolean value TRUE
Value returned:
boolean: TRUE

False
Constant: boolean value FALSE
Value returned:
boolean: FALSE

Is Defined
Test if a variable has been defined. Only valid if automatic definition of variable is switched off when transpiling (TODO! link to tag)
Value returned:
boolean: TRUE if the variable is defined, FALSE if not

Is Defined VARIABLE$
Description of the function.
Parameters:
VARIABLE$: Description of the parameter.
Value returned:
integer: Description of the value returned.

Array ARRAY
Return the memory address of the beginning of an array. May be imploemented in AOZ for education purpose
Parameters:
ARRAY: The array to get the address from
Value returned:
integer: 0 until imjplemented

Array VARIABLE$
Description of the function.
Parameters:
VARIABLE$: Description of the parameter.
Value returned:
integer: Description of the value returned.

Voice Synthetizer Commands

Instructions and functions to make your application speak

Say TEXT$, MODE
TODO!Speak a string using the build in voice synthetizer
Parameters:
TEXT$: Text to say
MODE:

Set Talk SEX, MODE, PITCH, RATE
TODO! Set the style of the speech
Parameters:
SEX:
MODE:
PITCH:
RATE:

Talk Misc VOLUME, FREQUENCY
TODO! Set volume and frequency of speech
Parameters:
VOLUME:
FREQUENCY:

Talk Stop
TODO! Stop Speech

Mouth Read
TODO! Start reading the position of animated mouth
Value returned:
integer: The position of the mouth

Mouth Width
TODO! Start reading the position of animated mouth
Value returned:
integer: The width of the mouth

Mouth Width
TODO! Start reading the position of animated mouth
Value returned:
integer: The height of the mouth

WebSocket instructions for AOZ

AOZ offers several instructions for managing connection interfaces.
A connection interface, called "Socket" allows the conversation between several remote computers.
Like a chat or a multiplayer game.
More informations on Web Socket API here: https://www.w3.org/TR/websockets/

WebSocket Open INDEX, TAGS
Open a Web Socket channel, enabling you to send and receive messages and files
Parameters:
INDEX: Index of the channel
TAGS: Unused, for future expansion

WebSocket Close INDEX
Destroys a WebSocket channel and all data associated
Parameters:
INDEX: Index of the channel

WebSocket Send Message INDEX, MESSAGE$, TAG$
Send a message through a WebSocket channel
Parameters:
INDEX: Index of the channel
MESSAGE$: Message to send, no check on length.
TAG$: Unused, for future expansion

WebSocket Is Message INDEX
Returns TRUE if a new message is available, FALSE if not
Parameters:
INDEX: Index of the channel
Value returned:
boolean: TRUE or FALSE if a new message is available

WebSocket Message$ INDEX
Returns a newly arrived string message
Parameters:
INDEX: Index of the channel
Value returned:
string: The message

WebSocket Is Error INDEX
Returns TRUE if a an error has occurred, FALSE if OK
Parameters:
INDEX: Index of the channel
Value returned:
boolean: TRUE or FALSE if an error has occured.

WebSocket Error$ INDEX
Returns the last error message
Parameters:

INDEX: Index of the channel

Value returned:

string: The message

Zones Commands

Instructions and functions to handle and detect zones

Reset Zone INDEX
Erase the definition of a zone, making it non-detectable
Parameters:
INDEX: (Optional) The index of the zone to reset, if ommited all zones will be reset

Zone$ TEXT$, INDEX
Return a magical string to use in the Print instruction when printing text, allowing you to define a zone around the text included as a parameter
Parameters:
TEXT$: The text to print, the zone will de defined from the top-left corner of the first character to the bottom-right corner of the last character. Warning: impredictible result if the text does not fit in one single line
INDEX: The index of the zone to set

Set Zone INDEX, X1, Y1, X2, Y2
Define a new zone in the current screen
Parameters:
INDEX: The index of the zone to set
X1: The horizontal coordinate of the top-left corner of the zone rectangle
Y1: The vertical coordinate of the top-left corner of the zone rectangle
X2: The horizontal coordinate of the bottom-right corner of the zone rectangle
Y2: The vertical coordinate of the bottom-right corner of the zone rectangle

Set Zone INDEX, X, Y, WIDTH, HEIGHT
Define a new zone in the current screen
Parameters:
INDEX: The index of the zone to set
X: Horizontal coordinate of the top-left corner of the zone rectangle
Y: Vertical coordinate of the the top-left corner of the zone rectangle
WIDTH: Width of the zone rectangle
HEIGHT: Height of the zone rectangle

Zone INDEX, X, Y
Check if a given coordinate lays inside of a pre-defined zone
Parameters:
INDEX: The index of the zone to test
X: Horizontal coordinate to test
Y: Vertical coordinate to test
Value returned:
boolean: True if the given coordinate lay insid eof the zone, False if not

Zone X, Y
Find the zone located at the given coordinates
Parameters:
X: Horizontal coordinate to test
Y: Vertical coordinate to test
Value returned:

integer: -1 if the coordinate lay outside of every zones, index of the zone if they lay inside of one

HZone INDEX, X, Y

Check if a given hardware coordinate lay inside of a zone. Hardware coordinate are only different from screen coordinate for retro-machine emulation (Amiga, Atari etc.)

Parameters:

INDEX: The index of the zone to test

X: Horizontal hardware coordinate to test

Y: Vertical hardware coordinate to test

Value returned:

boolean: True if the given coordinate lay insid eof the zone, False if not

HZone X, Y

Find the zone located at the given hardware coordinates. Hardware coordinate are only different from screen coordinate for retro-machine emulation (Amiga, Atari etc.)

Parameters:

X: Horizontal hardware coordinate to test

Y: Vertical hardware coordinate to test

Value returned:

integer: -1 if the coordinate lay outside of every zones, index of the zone if they lay inside of one

Mouse Zone

Return the index of a predefined zone (with "Set Zone") under the mouse

Value returned:

integer: -1 if the coordinate lay outside of all the zones, or the index of the zone if they lay inside of one

Zone On INDEX

Description of the instruction.

Parameters:

INDEX: Description of the parameter.

Zone Off INDEX

Description of the instruction.

Parameters:

INDEX: Description of the parameter.

Zones Length INDEX

Description of the function.

Parameters:

INDEX: Description of the parameter.

Value returned:

integer: Description of the value returned.

Zone Enabled INDEX

Description of the function.

Parameters:

INDEX: Description of the parameter.

Value returned:

integer: Description of the value returned.