PangoScript Commands

New for 2026, a tool has been created with all new fresh documentation of every Pango script Command. The tool linked below allows you to read every command as well as see examples the script commands in use. The tools is searchable with the search bar on the top.

Click the link below to review this tool which will open in a new window.

Click here to view the PangoScript command list.

Information below is being rewritten

```
==== Intro ====


Currently, the main purpose of "scripting" is ability to add custom relation
on various events, like a MIDI, DMX, ArtNet, Channels. The script typically
has a few lines, and a line has one Command. Command may have a parameters.
Typically parameter is a number (constant).


==== Numbers ====

Supported float point , integer and hexadecimal numbers. As example:
100
-20
1.01
0xBF
All hexadecimal numbers much have 0x prefix, similar to C language, but
without ending H

There is not strict separation on integer and float point numbers.

Separator between command parameter could be space (" ") or a comma (",")


==== Special characters (separators) ====


 '.', ';', ',', ':', '!', '', ''', '^', '+', '-', '*', '\', ' ',
 '`', '[', ']', '(', ')', '{', '}', '?', '%','|','&','=':
```

```
==== Predefined constants ====


There are a few constants mostly for readability of the code. Each constant
will be transformed to a number
TRUE - numeric analog 1
FALSE - numeric analog 0
ANY equal to numeric -1. Used in a few WaitFor command
OFF equal to numeric 0
ON equal to numeric 1
TOGGLE equal to numeric 2
AsIs equal to -2.
This is required for commands like "MasterPause". I hope it should be handy to
see ON or OFF instead of 0 or 1.

A few examples of using commands:
  VirtualLJ off
  MasterPause Toggle
  WaitForMidi 0x90, 10, Any

==== Math operations (expressions) ====



Standard: +  -  /   *  %
Inversion: !
bit OR: |
bit AND: &
bit XOR: ^
bit right shift: >>
bit left shift: <<


==== Operators ====



IF operator.

Syntax: if ( expression )  operator

expression - covered with brackets. Must be an expression with
numerical result that gives "0" or not a "0". So, for numerical
variable it's possible to write "if (variable) ...".

Compare operators are: ">" ">=" "<" "<=" "=" "<>". They produces numerical
result "0" or "1".
```

You can combine comparing operations with "&" and "|" bit-wise operators (but be sure that left and right side are "0" or "1"). It's recommended to cover complex operations into brackets - like "if ((1>2) & ((3+2)>1))"

operator will be executed if condition gets non-zero result. If you want to place other operator after "if (condition) operator" - it must be divided with ";" - but it's better to place next operator on the next line ;-)

=== GOTO operator ===

Syntax: goto label

"label" is a name of position for a jump. The label can be placed before any operator and separated by ":" character. As example:

```
mylabel: WaitForBeat 4
... do something...
goto mylabel
```

Operator GOTO can work with the labels inside the string variables. It means that you declare string variable, assign a label name to variable, and then you a variable in GOTO.
Example:

```
var s
s="mylabel"
goto s

DisplayPopup "It does not work"
exit

mylabel:
DisplayPopup "It works"
```

=== VAR operator ===

PangoScript allow define local variables. The lifetime for the local variable defined by lifetime of the Scripter that execute PangoScript. As soon as the Scripter freed, all its local variables are removed as well.

All variables must be declared before the using. No need to specify the type of variable. The variable automatically adjust the type depending on value.

```
Internally supported integer, float and string variables.
The declaration start from VAR operation and the follow one or more variable
names. As example:

var MyVariable
var a,b,c

Before using the variable must be initialized by some value. Otherwise BEYOND
will generate error and stop script execution. Example:
var MyInteger, MyString
MyInteger = 10
MyString = "Hello!"

All variables declared as VAR are local.


GLOBALVAR operator
The variable can be declared as global. In this case, it visible in ALL
scripts of BEYOND.




==== General functions ====


intstr(value:number):string - transform number to string
value - number or float number
result - string

floatstr(value:number):string - transform number to string
value - number
result - string

abs(value:number):number - value by modulus
value - integer or float number
result - same type as argument. result is absolute value

int(value:number):integer - return integer part of float value
value - a float point number

frac(value:number):float - return fractional part of float point value
value - float

round(value:float):integer - return rounded float point value (to integer)
value - a float point number
```

```
sqr(value:number):number - return a square of argument
value - a float point number or integer

sqrt(value:number):float - return a square root of argument
value - a float point number or integer

cos(value:float):float - co-sinus
value - a float point number or integer. angle in radians

sin(value: float):float - sinus
value - a float point number or integer. angle in radians

tan(value:float):float - tangents
value - a float point number or integer. angle in radians

arcsin(value:float)float - arc sinus
value - a float point number or integer. angle in radians

arccos(value: float):float - arc cosinus
value - a float point number or integer. angle in radians

arctan(value: float):float - arc tangents
value - a float point number or integer. angle in radians

arctan2(dx, dy:float):float - arc tangents
dx, dy - ....

min(a,b:number):number - return a minimum of two numeric values
a,b  float or integer

max(a,b:number):number - return a maximum of two numeric values
a,b  float or integer

pi:float - return PI value - 3.1415926...

invert(value:number):integer - invert value. Boolean operation, but can be
used with float point numbers. If value more than 0.5 then function return 0,
otherwise return 1.

==== Date and Time ====


now:float - date&time, calls now() function of Delphi

tickcount: integer - return number of millisecond from start of PC.

hms(hour, minute, second):integer - transform hour, minute and second into
seconds.
```

```
GetYear:integer - function return current year by PC clock. Result

GetMonth:integer - function return current month by PC clock

GetDay:integer - function return current day by PC clock.

timestr(now:float) :string
now - is variable representing time. Function return string with time in short
format such as "11:53", without seconds

timestrlong(now:float):string -
now - is variable representing time. Function return string with hours,
minutes, seconds, such as "11:53:10"

datestr(now:float):string - ********* ****** ****, * *** *** -
now - is variable representing date. Function return string short date format
such as "21.11.2012"

datestrlong(now:float):string - ******* ****** ****, **** -
now - is variable representing date. Function return string long date format
such as "21 ****** 2012 *."

dayofweek(now:float):string - short version day of the week

dayofweeklong(now:float):string - long version day of the week

==== String functions ====


uppercase(string):string  - transform input string to upper case. Result is a
string.

lowercase(string:string):string - transform input string to lower case. Result
is a string.

crlf:string - return a string, line separator (13,10)



==== Clock And Metronome ====


b2s(beats) - transform beats to seconds

b2ms(beats) - transform beats to milliseconds

s2b(seconds) - transform seconds to beats
```

```
b2s(seconds) - transform beats to seconds


==== Functions ====


There are a few functions for access of incoming data

Dmx ( Channel, OutputMin, OutputMax )
Parameters:
Channel - index of DMX channel, acceptable value from 1 to 2048
OutputMin, OutputMax - defines the range resulting value of the function.
Oupout value will start from OutputMin and increase up to OutputMax
result  = OutputMin +  (DmxChannelValue / 255) * (OutputMax-OutputMin)

Example:
  Position Dmx(10, -100, 100), Dmx(11, -100, 100), Dmx(12, -100, 100)

in this example we use Position command that has 3 arguments - X,Y,Z and we
use DMX IN values (three channels, 10,11,12), and map the values range -100 to
+100.

Dmx ( Channel  )
or
Dmx( Channel, OutputMin, OutputMax)
Parameters:
Channel - index of DMX channel, acceptable value from 1 to 2048
Result of the function is value of DMX channel as it is, without anr range
adjustments

OutputMin, OutputMax - defines the range resulting value of the function.
Output value will start from OutputMin and increase up to OutputMax.
Internally channels are normalized to 0...1 range
result  = OutputMin +  (ChannelValue) * (OutputMax-OutputMin)
Example:
  DisplayPopup Dmx(10) // display the value of 10th DMX channel.

Note: version with min/max used for simplification of use of function in
operators where you may need to transform DMX value range (0..255) to some
other range. As example Size operator, you may want to use size range like
0...100, or 10...100, or -100..100.

Channel ( Channel, OutputMin, OutputMax )
or
Channel( Channel )

Parameters:
Channel - index of Channel, acceptable value from 1 to 255
```

```
OutputMin, OutputMax - defines the range resulting value of the function.
Output value will start from OutputMin and increase up to OutputMax.
Internally channels are normalized to 0...1 range
result  = OutputMin +  (ChannelValue) * (OutputMax-OutputMin)

If function has one argument then result is normalized channel value (range
from 0 to 1)

Example:
  Size Channel(10, 5, 100), ChannelIn(10, 5, 100), ChannelIn(10, 5, 100),

in this example we use Size command that has 3 arguments - X, Y,Z and we use
Channel 10 value for all 3 axis.

Channel ( Channel )
Parameters:
Channel - index of Channel, acceptable value from 1 to 255
Result of the function is channel value. Channel value is normalized, the
range from 0 to 1.



CC (Channel, Controller, OutputMin, OutputMax)
Get a value of of MIDI controller. BEYOND memorize all values of incoming MIDI
controllers (Contrl Change Message) and you can get an access to it.
Channel - value 0..15
Controller - value 0..127. Thsi is Data1 in MIDI messages
OutputMin, OutputMax - defines the range resulting value. Result  = OutputMin
+  (ChannelValue/127) * (OutputMax-OutputMin)



ExtValue( OutputMin, OutputMax )
There are a few pretty big tables of code-sources. A tables such as DMX,
ArtNet, ControlChange, PitchBand, NoteOn initiated by channel/message that has
a value. This function allow get access to the channel that intiate the code.
Better explain on example. Lets say I want to connect DMX channel to Master
Live Control, Position X. In this case, code that will do ths is this:
    PositionIndex 1, ExtValue(-100,100)
In this example PositionIndex is a live control command, see below. 1 is index
of axis. And finally ExtValue() is a fuction that take DMX channel value in
map to rangle -100 to 100.

ExtDelta( Delta )
Function equal to ExtValue but made specially for MIDI encoders. There are
many controlled with wheels and knobs that generate Control Change message and
the data2 value is 00 or 7F. That is all, no values in between. For such cases
has a sense to use "delta" version of commands and ExtDelta() for getting
exact value of delta. Function return -Delta or +Delta. Only two values, no
```

```
exceptions. Example of use

PositionDelta 0, ExtDelta(1), 0   // move vertically.

another example:

AngleDelta 0,0, ExtDelta(45) // rotate by Z on 45 degrees.

Random( MaxValue )
Function return random value in range 0..MaxValue. Note, MaxValue included
into the range

RandomIn( MinValue, MaxValue )
Functino return random value in range MinValue..MaxValue

Param( ParamIndex ) - newer name of OscParam() function. It get the value of
parameter supplied into the script. See commend about OscParam()

OscParam( ParamIndex )
Specialy designed for use in "OSC to CODE" table.. When BEYOND receive OSC
message and supply it into interpreter to execution, then we can supply up to
10 parameters with OSC message. BEYOND put them into local array, and you may
use it inside the Code. Note, data is there only during execution of current
script. As soon  we exit from execution of this script, data will be lost. OK,
parameter index is from 1 up to 10. BEYOND check how many parameters the OSC
message has, and if you will try to access non existing paramater then script
will stop with error. So, if OSC message has 3 parameters, then you can use
index from 1 to 3. Accepts f, i, and s type of OSC parameters. You can freely
mix float and integer.

ParamRange( MinValue, MaxValue) function return TRUE (1)  if the first
parameter inside the specified range.

ParamRange(ParamIndex,  MinValue, MaxValue) function return TRUE (1)  if the
specified parameter (pram index) inside the specified range (from MinValue to
MaxValue)

GetTransitionName( index )
function return string, the name of transition. Index is number of transition.
Range 0..23

BeatTime
Function has no arguments. The result of function is float point number.
Integer part of function is number of beat from the start of BEYOND. The
fractional part is progress inside the beat.

GetMidiDeviceIndex
function return currently selected MIDI Device pair, Value range 1..4.
```

ObjectExists(AObjectName) - function get object name a s string and return 1 if object exists, or zero if object is not found.

==== Timecode input ====

GetTimeCode - return latest timecode value as a float point value, value in seconds

GetTimeCodeTick - return the tick when the last timecode value arrived. Use function GetTick for getting the current tick value.

==== Cue related functions ====

CuePlaying(PageIndex, CueIndex) - function return 1 if cue is currently playing in the Grid, otherwise return 0. Indexing of page and cue starts from 1.
Example: CuePlaying(1,1) //return state of first cue on a first page. Note, function is not applicable for ProTracks and DMX server.

CueEmpty(PageIndex, CueIndex) - function return 1 if cue is empty, otherwise return 0. Indexing of page and cue starts from 1.

GetCueCaptionColor(PageIndex, CueIndex) - return cue caption color in GDI format. Format is 24bit RGB, red is LSB. Overall formula Blue*256*256 + Green * 256 + Red. Indexing of page and cue starts from 1.

==== Misc ====

GetBeyondBuild - return integer value, build of BEYOND application.

GetMidiDeviveIndex - return current MIDI device index associated with this script. BEYOND can use 4 MIDI devices. By default script associated with 1st device. There is a command for change device index. This function allows to check current association. Indexing from 1.

GetMidiDeviceLayer - return current layer of MIDI device mapping. Indexing from 1. Layering introduced because BEYOND has big number of functions, and at the same time, MIDI device has limited number of sliders and buttons. MIDI settings allow to organize several layers, and as a result, MIDI controller will do several functions (depending on current layer). GetMidiDeviceLayer return currently active layer of MIDI device associated with this script.

```
==== User Interface ====


GetLcTabMode:integer - function return mode (destination of control) of Live
Control tab. Basically, it tell what button selected - Master, Cue, Zone or
ProTrack
Values are:
1 - Master
2 - Cue
3 - Zone
4 - ProTrack

GetTcTabMode:integer - function return mode of Time Control tab.Constant is
same as in  GetLcTabMode

GetFxTabMode:integer - function return mode of  FX tab.Constant is same as in
GetLcTabMode

GetGrid1Mode:integer - function return currently selected click mode of main
Grid. Result is one of following values
0 - Select mode
1 - Flash mode
2 - FlashSolo mode
3 - Toggle mode
4 - Restart mode
5 - ProTrack mode

GetGrid2Mode:integer - function return currently selected click mode of
secondary Grid. Constants same as for function GetGrid1Mode


==== Timeline mode functions ====


GetTimelinePos:float - function return time position of current timeline.
Value in seconds. Fractional part contain millisecond part.

GetTimelineDuration:float - function return duration of current timeline.
Value in seconds. Fractional part contain millisecond part.

GetTimelineOnline:integer function return 1 when "Enable laser output" is
enabled in Timeline mode, otherwise return value is 0.

GetTImelinePlaying:integer function return 1 when timeline is currently
playing, otherwise return value is 0.

GetTimlineTabIndex:integer. Timeline editor can work with multiple timelines.
Function return index of currently selected timeline.
```

GetTimelineTabName:string - function return tab name of currently selected timeline.

==== General commands ====

StartCue Page, Cell or StartCue "cue name"
Parameters: Page - page index, 1..100,. Cell - cell index, 1..100;
Parameters: "cue name" is a string that define cue name as we see it in the grid. At first BEYOND search for cue name in the current page. If not found than the search starting from the first page of the grid.
Purpose: Start the cue. If cue already started the no action

StopCue Page, Cell or StopCue "cue name"
Parameters: Page - page index, 1..100,. Cell - cell index, 1..100;
Parameters: "cue name" is a string that define cue name as we see it in the grid. At first BEYOND search for cue name in the current page. If not found than the search starting from the first page of the grid.
Purpose: Stop the cue. If cue is not started then no action.

ToggleCue Page, Cell or Toggle "cue name"
Parameters: Page - page index, 1..100,. Cell - cell index, 1..100;
Parameters: "cue name" is a string that define cue name as we see it in the grid. At first BEYOND search for cue name in the current page. If not found than the search starting from the first page of the grid.
Purpose: Start cue if it is not currently playing, or stop a cue if it is currently playing.

StopCueType mask, time
mask:
1-Image
2-Timeline
4-DMX
8-Fixture Sequence
16-Beams
32-Capture
time optional value, this is for soft stop. Zero means "right now".

PauseCue Page, Cell, State
Parameters: Page - page index, 1..100,. Cell - cell index, 1..60; State - recommended to use predefined constants OFF, ON or TOGGLE. Values - 0,1,2

```
Purpose: allow to pause or unpause the cue

RestartCue
Parameters: Page  - page index, 1..100,. Cell - cell index, 1..60;
Purpose: restart the cue if it already playing. Otherwise, no action.

CueDown Page, Cell
Parameters: Page  - page index, 1..100,. Cell - cell index, 1..60;
Purpose: does a "mouse down" on the cue

CueUp Page, Cell
Parameters: Page  - page index, 1..100,. Cell - cell index, 1..60;
Purpose: does a "mouse up" on the cue

StopCueNow Page, Cell
Parameters: - page index, 1..100,. Cell - cell index, 1..60;
Purpose:  Stop cue immediatelly without soft ending what is default option

StopCueSync Page, Cell
Parameters: page index, 1..100,. Cell - cell index, 1..60;
Purpose:  Stop cue softly and wait for completion

HoldClick state
Parameters: state - 0,1,2 or the constants ON, OFF, TOGGLE
Purpose: This is additional modifier for coming "click" command. It allow to
start the Cue in "HOLD" mode. HoldClick  command designed for with MIDI. You
can use a special button on MIDI Controller and inside the button down you
will enable hold mode, and on button release you will disable hold mode.


StopAllNow
Parameters: none
Purpose: Stop all running cues. Equal to click on menu "Run" -> "Stop All
Cues". This version stop all cues immediately
Example: StopAllNow

StopAllSync Time
Parameters: time in seconds.
Purpose: Stop all running cues with specified soft-stop time. Also, wait for
completion
Example: StopAllSync 0.2

StopAllAsync Time
Parameters: time in seconds.
Purpose: Stop all running cues with specified soft-stop time. Do not wait for
command completion
Example: StopAllAsync 0.5
```

```
BlackOut
Parameters: none
Purpose: The same action as click on "Blackout" button on a mail toolbar of
BEYOND
Example: Blackout



EnableLaserOutput
Parameters: none
Purpose: Access to "Enable Laser Output" button on mail toolbar. This command
enable output :-)
Example: EnableLaserOutput

DisableLaserOutput
Parameters: none
Purpose: Access to "Enable Laser Output" button on main toolbar. This command
enable output :-). There is no "toggle" version.
Example: DisableLaserOutput

MasterPause
Parameters: On  or Off or Toggle
Purpose: Control of "Pause" button on main toolbar. Of course it does an
action same as a button.
Example:
MasterPause On
MasterPause Toggle



MasterSpeed Speed
Parameter: Speed, normalized value in range from 0 to 1. 0 represent full
stop, 1 means normal speed
Purpose: It is global speed multiplier for players from Grid



VirtualLJ
Parameters: On  or Off or Toggle
Purpose: Control of "Virtual LJ" button on main toolbar. Of course it does an
action same as a button.
Example: VirtualLJ OFF



==== VLJFX ====



OneCue
Parameters: none
```

```
Purpose: Control of "One Cue" button on main toolbar. This command enable "One
cue" mode
Example: OneCue

OnePer
Parameters: none
Purpose: Control of "One Per Zone" button on main toolbar. This command enable
"One Per Zone" mode. Note, this mode share the button with OneCue mode
Example: OnePer


MultiCue
Parameters: none
Purpose: Control of "Multi Cue" button on main toolbar. This command enable
"Multi cue" mode
Example: MultiCue

Transition
Parameters: On  or Off or Toggle
Purpose: Control of "Transition" button on main toolbar. This command enable
or disable the transition.
Example: Transition On

ClickSelect
Parameters: none
Purpose: Control of "Select" button on main toolbar. This command enable
"Select" mode.
Example: ClickSelect

ClickToggle
Parameters: none
Purpose: Control of "Toggle" button on main toolbar. This command enable
"Toggle" mode.
Example: ClickToggle

ClickRestart
Parameters: none
Purpose: Control of "Restart" button on main toolbar. This command enable
"Restart" mode.
Example: ClickRestart

ClickFlash
Parameters: none
Purpose: Control of "Flash" button on main toolbar. This command enable
"Flash" mode.
Example: ClickFlash

ClickSoloFlash
```

```
Parameters: none
Purpose: Control of "SoloFlash" button on main toolbar. This command enable
"SoloFlash" mode.
Example: ClickSoloFlash

ClickLive
Parameters: none
Purpose: This button is currently invisible, but t active mode of LivePRO -
like tracks.
Example: ClickLive

ClickTrack - enable ProTrack click mode.

StartPrevious - start previously started cue. Command is for button Back of
the main toolbar (Grid mode).

TogglePrevious - stop current cue and start previously stated cue. Command is
for button Swap of the main toolbar (Grid mode).

TimerBeat
Parameters: none
Purpose: Simulate Timer beat (BPM timer on the main toolbar)
Example: TimerBeat

AudioBeat
Parameters: none
Purpose: Simulate Audio beat what usualy comes from AudioIn and FFT.
Example: AudioBeat

ManualBeat
Parameters: none
Purpose: Manual beam comes from keyboard. Now it is possibl to initiate it
from code.
Example: ManualBeat


SetBPM value- set master Beat timer value in BPM (beat per minute). The talk
about BPM (Metronome) panel on the main toolbar in the Grid mode.

SetBpmDelta value  - increase or decrease the BPM. This is relative version of
SetBPM command. It designed for use with MIDI controllers, where MIDI button
can be used for change of tempo.


BeatTap
Parameters: none
Purpose: This is tap-tempo. It work in BPM panel in a main toolbar and allow
```

enter beat-per-minute by means of tapping. Command work same as click on BMP panel. Or "space" key.
Example: BeatTap

BeatResync
Parameters: none
Purpose: Synchronize Beat timer to the moment of this command execution. Same as "backspace" key
Example: BeatResync

==== FX and Live Control commands ====

The Destination

A few words about commands of BEYOND. Each command of BEYOND has information about:
Sender - who initiated the command
Server - who is recipient / server of the command. As example - zone, cue, master, and so on
Server index - index of server, of there are many such servers. As example - Projection Zone
Command - the command itself
Arguments - depends on exact commands.

Lets consider example. We want to set the size of second projection zone to 50%. In this case, command will have such values
Sender - BEYOND set it automatically, no worry about
Server - Zone
Server index - 2
Command - Size
Arguments - 50,50,50

For the script, to make a complex command that include all fields is not practical. Because the process has two parts. The first part - you define the destination. By default it is Master. The second part is a Command itself. BEYOND memorize the setting of destination and all consequent commands will use it.

ControlMaster
Parameters: none
Purpose: Select Master LiveControl as a destination for Live Control commands

ControlCue PageIndex, CellIndex
Parameters: PageIndex (1..PageNumber) and Cell Index 1..100. By default the

```
grid has 60 cells (10x6 mode). But the grid may have another dimension with
100 cells as max.
Purpose: Select Cue LiveControl as a destination for Live Control commands

ControlZone ZoneIndex
Parameters: Zone index, 1..ZoneNumber;
Purpose: Select Zone LiveControl as a destination for Live Control commands

ControlProTrack ProTrackIndex
Parameters: Iindex, 1 to number of ProTracks created in the BEYOND/
Purpose: Select ProTrack LiveControl as a destination for Live Control
commands. Possible to select only one ProTrack

ControlSelZones
Parameters: none
Purpose: In opposite to ConrolSelZones, this command address all selected
zones of BEYOND. BEYOND keep the zones in list and each zone has Selected
property. The command normally goes to only one destination - one zone, one
cue, one track. But specially for external  control, added ability to send the
Live Control command to multiple destinations. Technically, BEYOND will send
many command, by one for each selected zones.
Also, this command address selected ProTracks.

ControlSelCues
Parameters: none
Purpose: Control currently selected Cues. This command allow to address Live
Control to multiple (currently selected) cues. The action of command limited
to one page only. If BEYOND in Grid mode, then will be used the page of Main
grid. If BEYOND in Timeline/PlayList mode, then will be used page from bottom
side grid.

ControlFromUI



Note: commands above are local commands, and modify internal register of
interpreter that execute this code. It means, that the setting work only for
this code, and till




==== FX Control ====



FX Index1, (Index2,...)
Parameters: At least one argument that define  state of FX layer. Number of
arguments should be equal, but can be less than number of layers in FX grid.
Command designed to setting all FX layers at once, by means of one command.
```

```
For setting each Layer separately use SetFX
Purpose: Control of FX register of cues, zones, and master

 0 - stop.
 1..100: Effect indexing start from zero and in current version up to 100
Example:
FX 0,0,0,0 // stop four FX layers
FX 10 // set 1st later to effect number 10.

Example:
 FX 2,3,4,5 // lineary select 2th, 3rd, 4th, 5th FXs
 FX  0,0,0,0 // stop all



FXAction Action1, (Action2,...)
Parameters: At least one argument that define  state of FX layer action.
"Action" define a morph between original frame and frame after effects. Number
of arguments should be equal, but can be less than number of layers in FX
grid. Command designed to setting all FX layers at once, by means of one
command. For setting each Layer separatelly ise SetFXAction
 0 - OFF, no action at all
100 - norma;. full action.
Example:
FXAction 0,0,0,0 // set action for four FX layers

Note: Action parameter is not same as "stop" effect. It relatively slow
operation because of calculation of morph, extra buffers, and relating
calculations.



SetFX  LayerIndex,  EffectIndex,[EffectIndex,EffectIndex...]
A bit more compact version. First parameter index of layer 1 to 4, and Effect
index of 1..100.
Set state of one layer.
If EffectIndex is not defined, then command will stop the layer
If there are multiple Effectindex then command will activate multiple effects
in this layer

SetFXAction LayerIndex, Action
Purpose: set Action parameter of one layer
Layer index - index of FX layer, 1..N
Action - 0..100, in percents



SetFXAction1 ActionValue
SetFXAction2 ActionValue
```

```
SetFXAction3 ActionValue
SetFXAction4 ActionValue
Purpose: set Action parameter of one layer
Action - 0..100, in percents

SetFXActionDelta ActionDelta1, [ActionDelta2,..]
Shift the action value bu delta. For each layer value of delta is personal.
At least one argument required, Starts from 1st layer and up.

FXTimeScale Layer, ClockSale, BeatScale

FXTimeScaleDelta Layer, ClockSale, BeatScale

FXTimeShift Layer, ClockValue, BeatValue

FXTimeShiftDelta Layer, ClockValue, BeatValue

ShiftFX LayerIndex,  Direction
Delta version of control. Allow to shift effect index on N

ToggleFX LayerIndex
Toggle between internal LC register and current state. Layer index is
optional. If Layer not defined that will be swapped all layers

ShiftFX  LayerIndex, IndexDelta
LayerIndex - 1..N
IndexDelta - how to shift current index


DropFX Layer, Effect, DurationMS
Layer - index of effect later in grid
Effect - inded of effec in grid
DurationMS - define how long effect will be executed


FXClick Layer, Index - click on effect of FX grid, with absolute effect index,
1..N

FXCellClick Layer, CellIndex - click on effect of FX grid, but effect index
based on left corner, so, click on visible area

FXScroll BaseIndex
Base index - control scrollbar of FX gird, define left corner. 0 - default
state.
```

```
==== Live Control ====



Size X, Y, Z
Parameters: X, Y and Z are size in %. 100% means a full size. Max value 400%.
Purpose: Access to Size of LiveControl object.
Example: Size 100,100,100

SizeDelta X, Y, Z
Parameters: X, Y and Z are deltas that will be added to Size values. Value in
%.
Purpose: Access to Size of LiveControl object. Good to incrementive control
Example: SizeDelta 0,1,0

Zoom 100
Parameter - zoom value, 0..100
ZoomDelta 0
Parameter - zoom delta,

Position X,Y,Z
Parameters: X, Y and Z are value of Position. Value in %. 0,0,0 is a center :-
), 100% is boundary. Max value 400%.
Purpose: Access to Position of LiveControl object.
Example: Position 100,0,0  // move to teh right corner

PositionDelta X,Y,Z
Parameters: X, Y and Z are deltas for Position. Value in %
Purpose: Access to Positiion of LiveControl object.
Example: PositionDelta -5,0,0  // move to the left on 5%

PositionIndex AxisIndex, Value
Parameters: Axis Index means: 0 is X, 1 is Y, 2 is Z. Value is value of
position, in %, as usual :-)
Purpose: Access to Position of LiveControl object.
Example: PositionDelta  2, 1  // move by Z axis on 1%

AngleX angle
Parameters: Angle in degrees
Purpose: Access to Rotation of LiveControl object.
Example: AngleX 45

AngleY angle
Parameters: Angle in degrees
Purpose: Access to Rotation of LiveControl object.
Example: AngleY 0

AngleZ angle
```

```
Parameters: Angle in degrees
Purpose: Access to Rotation of LiveControl object.
Example: AngleZ 180

Angle X, Y, Z
Parameters: X,Y ans Z are angle in degrees. Directly define a value of
rotation angles
Purpose: Access to Rotation of LiveControl object.
Example: Angle 0, 0, 90

AngleDelta  X, Y, Z
Parameters: X,Y ans Z are deltas of angle in degrees.
Purpose: Access to Rotation of LiveControl object.
Example: AngleDelta 0, 0, 5 // increment Z rotation angle

RotoSpeedX  Value
Parameters: Value - speed. Degrees per second.
Purpose: Access to Rotation of LiveControl object.
Example: RotoSpeedX 0

RotoSpeedY Value
Parameters: Value - speed. Degrees per second.
Purpose: Access to Rotation of LiveControl object.
Example: RotoSpeedY 0

RotoSpeedZ Value
Parameters: Value - speed. Degrees per second.
Purpose: Access to Rotation of LiveControl object.
Example: RotoSpeedX 0

RotoSpeed X, Y, Z
Parameters: X,Y and Z are rotation speed for corresponding axis
Purpose: Access to Rotation of LiveControl object.
Example: RotoSpeed 0, 0, 0   // turn off rotation

RotoSpeedDelta X, Y, Z
Parameters: X,Y and Z are deltas of rotation speed.
Purpose: Access to Rotation of LiveControl object.
Example: RotoSpeedDelta 0,0, -1 // decrement of rotation speed

Brightness Value
Parameters: Brightness, 0..100%
Purpose: Access to Brightness of LiveControl object.
Example: Brightness 50  // half dimmed image

BrightnessDelta Value
Parameters: delta-version of brightnes control
```

```
Purpose: Access to Brightness of LiveControl object.
Example: BrightnessDelta +1 // positive increment

VisiblePoints Value
Parameters: percentage of visible points, 0%..100%.
Purpose: Access to VisiblePoints of LiveControl object.
Example: VisiblePoints 100

VisiblePointsDelta Value
Parameters: delta-version of brightnes control
Purpose: Access to VisiblePoints of LiveControl object.
Example: VisiblePointsDelta 0

ColorSlider Value
Parameters: Value range is 0..255.
Purpose: Access to Color Slider of LiveControl object.
Example: ColorSlider 0

ColorSliderDelta Value
Parameters: Value is delta to be added to ColorSlider
Purpose: Access to Color Slider of LiveControl object.
Example: ColorSliderDelta 5

AnimationSpeed Value
Parameters: Value range is 25..200
Purpose: Access to AnimationSpeed Slider of LiveControl object.
Example: AnimationSpeed 100 // defaul value

AnimationSpeedDelta Value
Parameters: Value is delta to be added to AnimationSpeed
Purpose: Access to AnimationSpeed Slider of LiveControl object.
Example: AnimationSpeedDelta -10

ScanRate Value
Parameters: Value range is 25..200
Purpose: Access to ScanRate Slider of LiveControl object.
Example: ScanRate 100 // defauul state

ScanRateDelta Value
Parameters: Value is delta to be added to ScanRate
Purpose: Access to ScanRate Slider of LiveControl object.
Example: ScanRateDelta 10

RGBA  Red, Green, Blue, Alpha
RGBA Red, Green, Blue
RGBA Index, Value

If 4 parameters: Red, Green, Blue are color channels, values 0..255. Alpha is
```

```
transparency control, range 0..255
If 3 parameters: Red, Green, Blue are color channels, values 0..255. Same as
four parameters, but alpha channel stay unchanged
If 2 parameters then Index define what parameter to change, and Value define
the value of corresponding channel.
Index values are
0 -  Red
1 - Green
2 - Blue
3 - Alpha

Purpose: Direct color control of  of LiveControl object. RGBA is additional
and independent layer to ColorSlider parameter
Example: RGBA 255, 255, 0, 128  // makes image yellow-shaded (on 50%)

RGBADelta Red, Green, Blue, Alpha
RGBADelta Red, Green, Blue
RGBADelta Index, Value

If 4 parameters: Delta (relative change) by Red, Green, Blue and Alpha
channels
If 3 parameters: Delta (relative change) by Red, Green, Blue and Alpha
channels
If 2 parameters then Index define what parameter to change, and Value define
the delta of value of corresponding channel.
Index values are
0 -  Red
1 - Green
2 - Blue
3 - Alpha
Example: RGBADelta 0, 0, 0, -16   decrease alpha channel value

ResetAllRoto
Parameters: none
Purpose: reset all rotation to zero, and clear accumulator
Example: ResetAllRoto

ResetLiveControl
Parameters: none
Purpose: reset all sparameters of LiveControl object to default
Example: ResetLiveControl




==== Live Control - UI surface ====
```

Live Control tab contain a sliders like a Size, Color, etc. Each such slider has two button that scroll the slider to minimum or to maximum value. What the button does - define a speed of motion. As soon as slider hit the boundary, motion will stop. Speed means a shift of slider during one iteration. Speed can be positive, or negative. Value zero means stop the motion now. In total, we have 11 functions, each

ClickScrollZoom speed.
ClickScrollSize speed.
ClickScrollFade speed
ClickScrollVPoints speed
ClickScrollScanRate speed
ClickScrollColor speed
ClickScrollAniSpeed speed
ClickScrollR speed
ClickScrollG speed
ClickScrollB speed
ClickScrollA speed

Each function control corresponding slider in Live Control tab. Note, that this is control of UI slider. This is not a control of some exact LiveControl object.

ClickLCTabMode Index
Parameters: Index represent mode of Live Control tab. (In UI mode controlled by Master,Cue,Zone, and Protrack buttons on toolbar. Values of Index are:
1 - Master
2 - Cue
3 - Zone
4 - ProTrack
Purpose: Control of Live Control tab mode
Example: ClickLCTabMode 2 // control currently selected cue in the grid

ClickTCTabMode Index
Parameters: Index represent mode. See constants of ClickLcTabMode above.
Purpose: Control of Time Control tab mode
Example: ClickTCTabMode 1 // control master

ClickFXTabMode Index
Parameters: Index represent mode. See constants of ClickLcTabMode above.
Purpose: Control of FX tab mode
Example: ClickFXTabMode 1 // control master

ClickFXStopAll
Parameters: no arguments
Purpose: Same as click on Stop All button on FX tab (stop all effects in

```
currently selected Live Control)
Example: ClickFXStopAll

ClickFxVlj Index, State
Parameters: Index of FX line. Indexing from 1. Up to 8 lines. State is
optional parameter and represent state of VLJ for this FX line (on or off).
State values are
0 - Off, 1 - On, 2 - Toggle.
Purpose: Enable or disable VLJ for FX line.
Example: ClickFxVlj 2, 0 // disable VLJ for second FX line




==== WaitFor command group ====


Sleep Time
Parameters: Time measured in milliseconds
Purpose: Allow to pause execution to defined time
Example 1: Sleep 100 // this will cause pause in execution on 1/10 of a
second.
Example 2:
MidiOut 0x90, 0x40, 0x7F
Sleep(500);
MidiOut 0x90, 0x40, 0x00
Sleep(500);
Restart


WaitForMidi Command, Data1, Data2
Parameters: Command, Data1, Data2. MIDI messages contain 3 bytes. 1st known as
Command, and ha steh range 0x80h, up to 0xFF. Data1 and Data2 are parameters
of command, max value is ox7F (127 decimal). It is possible to use special
value "minus one", (-1), it will means "any value". As example: WaitForMidi
0x90,-1,-1   this command will wait for any NoteOn message, any note (Data1 is
-1), any volume (Data2 is -1).
Purpose: Allow to make reaction on incoming MIDI event.
Example:
WaitForMidi 0x90, 0x01, -1
NextPage
Restart
Note: See SelectMidi command that allow select a MIDI device from what expect
the message. SelectMidi control input and output operations
```

```
WaitForArtNet ChannelIndex
Parameters: ChannelIndex
Purpose: Wait for change of value in incoming ArtNet packet. ChannelIndex
values from 1 to 512. Special value "-1" means any channel index
Example: WaitForArtNet 2

WaitForDmx ChannelIndex
Parameters: ChannelIndex
Purpose: Wait for change of value in incoming DMX packet. ChannelIndex values
from 1 to 512. Special value "-1" means any channel index
Example: WaitForArtDMX 512

WaitForChannel ChannelIndex
Parameters: ChannelIndex
Purpose: Wait for change of internal Channels. BEYOND has 256 channels that
typically used in effects, shapes, abstracts.  ChannelIndex values from 1 to
255. Special value "-1" means any channel index
Example: WaitForChannel 512

WaitForCueStart PageIndex, CellIndex.
Parameters: PageIndex, CellIndex.
Purpose: Wait for start of exact Cue. Cue addres composes of Page and Cell.
BEYOND has up to 64 pages, and 60 cell (cues) per page. So, address is two
numbers - page and cell. Special value "-1" means "any"
Example: WaitForCueStart  -1, 10

WaitForCueStop PageIndex, CellIndex.
Parameters: PageIndex, CellIndex.
Purpose: Wait for stop of exact Cue. Cue addres composes of Page and Cell.
BEYOND has up to 100 pages, and 60 cell (cues) per page. So, address is two
numbers - page and cell. Special value "-1" means "any"
Example: WaitForCueStop  -1, 10

WaitForCellDown CellIndex.
Parameters: CellIndex.
Purpose: Main storage of cues is a Grid. When we click on the grid cell teh we
start a cue, at least this is initial, and default action. But, click itself
is just a click. There is special "input manager" that transform mouse down,
mouse up events into commands to start to stop the cues. Managr takes into
account current mode, current page, state of the cue (playing or no) and after
this generate a command to start (or stop) the cue. OK, this version allow to
wait and react on clicks
Example: WaitForCellDown  1

WaitForCellUp CellIndex.
Parameters: CellIndex.
Purpose:  OK, this version allow to wait and react on "release" the cell.
Example:
```

```
WaitForCellUp 1
MidiOut 0x90, 0x40, 127
Sleep 100
MidiOut 0x80, 0x40, 0
Restart

WaitForTime Hour,Minute, Seconds, Milliseconds
Parameters: Hour - 0..23, Minute - 0..59, Seconds - 0..59 , Milliseconds -
0..999;
Purpose:  Wait for "real" time defined by PC clock
Example:
WaitForTime 21,00,0,0 // wait for 9 PM, and then allow code for go forward

WaitForTimePos Hour,Minute, Seconds, Milliseconds
Parameters: Hour - 0..23, Minute - 0..59, Seconds - 0..59 , Milliseconds -
0..999;
Purpose:  wait for time from begining of the script execition. Time is
relative to the script start. "Restart" command clear local script time
Example:
WaitForTime 0,1,45,0 // wait for 1 minute, 45 seconds from teh script start.


WaitForHotkey no arguments
Parameters:  none
Purpose:  Each script in Code list has a hotkey that can activate it. Waiting
for hotkey what allow create multiple reaction, or reaction loop by means of
restart command
Example:
WaitForHotkey
DmxOut 1, 255 // set first channel to 255
Sleep 500 // wait a half of second
DmxOut 1, 0 // zero first channel


WaitForAudioBeat no arguments
Parameters:  none
Purpose:  what for audio beat, same as does VLJ

WaitForManualBeat no arguments
Parameters:  none
Purpose:  what for manual beat

WaitForTimerBeat no arguments
Parameters:  none
Purpose:  what for main beat controlable from main toolbar.

==== Custom Events ====
```

```
WaitForEvent  Name[, Name, Name]
Parameters - at least one string that define name of event. May be used more
than one event names.

PulseEvent Name[, Name, Name]
Parameter - at least one name of the event. It will reactive the scripts that
suspended by WaitForEvent


==== Workspace Pages, Tabs, Category ====


SelectGrid Index
Index allow to choose what grid will be used for executing commands below.
Zero (0) means "recently selected" grid. 1 means main grid that in center of
main form. 2 means secondary grid that is on bottom tabs.

SelectTabName "Name"
Select the tab of the grid by specified name. Name is a string. Function work
within current category.

SelectTab Index
Select the tab by linear index. Counting starts from 1. Function work within
current category.

SelectNextTab
Select the next tab relatively to current. No arguments. Function work within
current category.

SelectPrevTab
Select the previous tab relatively to current. No arguments. Function work
within current category.


SelectPageName "Name"
Select the page/tab of the grid by specified name. Name is a string. Function
work independently on current category.

SelectPage Index
Select the page/tab by linear index. Counting starts from 1. Function work
independently on current category.

SelectNextPage
Select the next page/tab relatively to current. No arguments. Function work
independently on current category.

SelectPrevPage
Select the previous page/tab relatively to current. No arguments. Function
```

```
work independently on current category.

SelectAllCat
No argumens. Select all categoris of pages, same as click on "All"

SelectCatName CategoryName
CategoryName - string. Select a category by name. Names of categories are
completelly user defined, and may vary from workspace to workspace.

SelectCat Index
Select category by index. Counting from 1. -1 mean all, this is analog of
SelectAllCat.

SelectNextCat
Select next category relatively to current. No arguments.

SelectPrevCat
Select previous category relatively to current. No arguments.


==== Zones ====


UnselectAllZones
Unselect all projection zones. Analog of "use cue defined routing". No.
arguments

StoreZoneSelection
Store information about currently selected zones in local buffer of
interpreter. Lifetime of strong is limited to lifertime of time of executing
this code

ReStoreZoneSelection
Restore previously store zone selection.

SelectZone Index
Select the zone by index. Argument - number. Counting from 1.

UnSelectZone Index
Select the zone by index. Argument - number. Counting from 1.

ToggleSelectZone Index
Toggle zone selection state. Argument - number. Counting from 1.

SelectZoneName ZoneName
Select zone by name. Argument - string.
```

```
UnSelectZoneName ZoneName
Unselect zone by name. Argument - string.

ToggleSelectZoneName ZoneName
Toggle zone selection state. Argument - string.



==== Cell Navigation ====


Command from this section designed to actiavte the cues within the page.
Functionality affected by SelectGrid !!!

FocusCell Col, Row
Set focused cell by Column and Row. Columt range 1..10. Row range 1..6.

FocusCellIndex Index
Set focused cell by index (1..60)

StartCell
No arguments. Start focused cell

StopCell
No arguments. Stop focused cell

ShiftFocus DeltaIndex
Shift focused cell on DeltaIndex cells forward or backward. Positive value -
go forward. Independently on step value the resulting index will be valid
range.

MoveFocus DeltaX, DeltaY
DeltaX - how many cells to go horizontaly
DeltaY - how many cells to go vertically

==== Local Commands ====



Restart
Parameters: none
Purpose:  start exection from a first line. Use it if you need to start from
begining. Typical use - some WaitFor command at begining and Restart at the
end
Example: Restart

Exit
Parameters: none
Purpose:  stop the script execution
```

```
Example: Restart

Autostart
Parameters: none
Purpose: Automatically start the script execution. Suitable for PangoScript
tab and MIDI background scripts.

StopOnBlackout  0,1 or ON, OFF
Purpose: this option define how the scripter should react on blackout. By
default (at least now), the script execution will not be interrupted. If to
will enable this option then the script execution will stop.

CodeName string
Parameters: String. Allow define a program name in the Code list. This string
is also used for identification of Code for StartCode and StopCode functions.
Example: Name "my first code"

CodeShortcut shortcut-string
Paremeter: a string that contain a shortcut in text form. As example:
""Ctrl+A". Delphi has functions ShortcutToText and TextToShortcut. It allow
transform. Main idea - allow code contain information about its name and
shortcut as a text, inside the code. How to know shortcut text? right now -
there is dialog that allow define a shortcut to Code, and you can see how it
looks.

StartCode CodeName
Parameter: string that define code name to start. Could be used only code from
the main pool (listbox in Code tab);
Purpose:  This allow to actiavte one code from another code, for making a
complex reactions. In act, this is analog of subroutine
Example:
// First script
WaitForHotkey "F1"
StartCode "Rotate Z"
WaitForHotekey "F1"
StopCode "Rotate Z"
Restart

// Second script
CodeName "Rotate Z"
WaitForBeat
LCMaster
AngleDelta 0,0, 5
Restart


StopCode CodeName
```

```
Parameter: string that define code name to stop.
Purpose:  Stop the execution of specified code, a pair for StartCode. (see
above)



==== Output ====


SelectMidi DeviceNumber
Parameters: DeviceNumber is from 1 to 4 and correspond to device number of
MIDI settings dialog. Special value "-1" accepted for WaitForMIDI command
only. MidiOut do not accept this value and will ignore it.
Example:
SelectMidi 1
WaitForMidi 0xC0, 00, ANY

See also - GetMidiDeviceIndex. Function return currently selected device
number

SetMidiLayer



MidiOut Message, Data1, Data2
Parameters: 3 parameters, standard for MIDI, Message is value in range of 80h
up to FFh. Data1 and Data2  from zero to 7Fh.

DmxOut Channel, Value
Parameters: Channel is index of DMX channel, value range 1..512. Value is
state of DMX channel, range 0..255
Example: DmxOut 1,255

DmxOutRange



ChannelOut ChannelIndex, Value
Parameters: ChannelIndex is index of internal BEYOND Channel, value range
1..255. Value is state of DMX channel, range 0..1000.
Example: ChannelOut 1,1000

OscOut Address, [Agrument]
The first parameter is osc message address. string.
Second and other agruments are optional. Types - string or number.

==== Unsafe commands ====


RunApp File, Parameters, ShowCmd
```

```
Purpose: This command use ShellExecute() function inside it. Please check
MSDN:
http://msdn.microsoft.com/en-us/library/windows/desktop/bb762153(v=vs.85).aspx
.
Reason of adding this command - ability to run extenal application from
BEYOND. Internally, BEYOND use this code:

ShellExecuteW(0,nil,PWideChar(FFile), PWideChar(FParameters), nil,
round(FShowCmd))

Parameters:
File - string. exe file name including path.
Parameters - string.
ShowCmd - number, see MSDN for details are optional. For more info please
check MSDN. Default value is SW_SHOWNORMAL (1)

Example: RunApp "C:\Windows\Notepad.exe", "", 1

IMPORTANT: This command is disabled by default because of security reasons. If
you going to use this command, enable it in "Configuration" dialog, "Security"
tab.

==== Limiters ====


SetLimiterProfile Index
Parameter - index of profile. There are two profiles, the 1st for "One Cue"
mode, and second for "Multi Cue" mode. You should select 1 or 2 before using
SetLimiter commands. This also will affect the state of BEYOND and change the
active profile. It is recommended to call OneCue, OnePer or MultiCue command
after the profile.

SetLimiterPerZone Value
Value specify max number of cues per Zone. The range is 1..30. The range is
big, and we recommend to keep the values in reasonable limits

SetLimiterPerGrid Value
Value specify max number of cues per Grid. The range is 1..30.

SetLimiterFlash Value
Value specify max number of cues that can be in Flash mode. The range is
1..30.

SetLimiterHold Value
Value specify max number of cues that can be in Hold mode. The range is 1..30.

SetLimiterBeam Value
```

```
Value specify max number of playing Beam cues. The range is 1..30.

SetLimiterDMX Value
Value specify max number of playing Beam cues. The range is 1..30.

SetLimiterShow Value
Value specify max number of playing Show cues. The range is 1..30.


==== Timeline control ====


PlayTimeline
Start playback of current show in timeline editor

StopTimeline
Stop playback of current show in timeline editor

TimelineMarker
Add timeline marker. There are 3 modifications
TimelineMarker (no arguments) - add marker at current position and current
color
TimelineMarker Color - addmarker with specified color at current time position
TimelineMarker Color, Time - add marker with specified color at specified time
Time - time in seconds, float point
Color  - index, 1..10;


==== QuickDMX Editor ====


SetDmxEditorChannel


==== Play List control ====


PlayListStop
Stop playing of current show

PlayListPlay
Start playing of current show

PlayListFirst
Jump to first show in the list. Work in play and edit modes.

PlayListLast
Jump to last show in the list. Work in play and edit modes.
```

```
PlayListNext
Jump to next show in the list. Work in play and edit modes.

PlayListPrev
Jump to previous show in the list. Work in play and edit modes.
PlayListSetPos index
Jump to specified show of the list. Index define a number of show in the list

PlayListSetTime time
Jump to time position of the show, value defined in seconds. Work only in PLAY
mode.


==== Debug messages ====


qLog  string or numeric argument(s)
Purpose: display message in QLog tab. Usefull for debug purposes.

Write string or numeric argument(s)

Purpose: create special output window and display message.

WriteLn string or numeric argument(s)
Purpose: create special output window and display message.


==== Player control ====


InvertPlayersTime
No arguments. Invert time direction of all currently playing cues. Works for
all dynamic players. Used in corresponding button on Master tab

PlayersSetJump (no argument)
No arguments. The first call memorize current player position and second call
restore the time of player to that position. See PlayersResetTime that clear
this time storage. Works for all dynamic players. Used in corresponding button
on Master tab

PlayersResetJump
No arguments. Function is an addition for PlayersSetJump, it clear the saved
time to zero,. Works for all dynamic players. Used in corresponding button on
Master tab

PlayersSetLoop
No arguments.One more function of "time fun" groups. First call memorize time
```

position A, second call memorize time position B, and after that, Player bounce the time within specified A-B range. Third call of this procedure clear the time varibales, and player continue normal playback. Works for all dynamic players. Used in corresponding button on Master tab

PlayersDisk SpeedMultiplier, PositionDelta
PlayersDisk provide access to "DJ disk" from Master tab of BEYOND. Disk action based on two things - distance from center that define the slowdown of rotation (speed control) and second is position shift - what actually is time shift delta.
SpeedMultipler is normalized value within 0..1 range, where 1 is full speed, 0 a full stop
PositionDelta is a shift of time, backward or forward. Value in radians. Each call add this value to time position of the players. Parameter is optional!, there is a special procedure for this if you plan to use it from MIDI.
Works for all dynamic players. Used in corresponding button on Master tab

PlayersDiskShift PositionDelta
Command does the time shift (on delta) of all currently playing cue. It is simple action like Time=Time+Delta;
Command made for use with MIDI and action is equal to PlayersDisk command.

==== Property Animation ====

Property animation designed for linear change of property from one state to another during some period of time. Technically, this is sort of micro script, but BEYOND does this job automatically. Internally BEYOND has internal list of the tasks for object property animation. The list is dynamic. The only way to create request for property animation is a script.

Note, the execution of property animation performed right after execution of all script by dedicated thread of BEYOND. The time resolution is ~ 40 "FPS". For animation of laser related properties it might be not perfect because laser projector frame rate may be much higher.

AnimateProp PropertyName, StartValue, FinishValue, DurationMS, FinishEvent
PropertyName - string that contain complete name of object and its property. The property must be numeric.
StartValue - number. Specify start value of the property during animation
FinishValue - number. Specify final value of the property during animation.
DurationMS - duration of animation in milliseconds
FinishEvent - optional parameter, string. Specify name of Event that will be activated at the end of animation. Action equal to call of PulseEvent() procedure.

```
Exmaple:
AnimapeProp "Master.Brightness", 100, 0, 1000
// command change value of Master brightness from 100% to zero during 1000 ms
(one second)

Example2 :
AnimapeProp "Master.Brightness", Master.Brightness, 0, 500
// command change value of Master brightness from current value to zero during
500 ms (0.5 second)



AnimatePropDelta PropertyName, TotalDelta, DurationMS, FinishEvent
PropertyName - string that contain complete name of object and its property.
The property must be numeric.
TotalDelta- number. Specify how much will change the value of specified
property
DurationMS - duration of animation in milliseconds
FinishEvent - optional parameter, string. Specify name of Event that will be
activated at the end of animation. Action equal to call of PulseEvent()
procedure.

Example:
AnimatePropDelta "Master.Brightness", -25, 300    // decrease master
brightness on 25% during 0.3 second

Example 2:
AnimatePropDelta "Master.Color", -32, b2ms(1)    // shift color "slider" of
master on 32 during 1 beat

DeletePropAni PropertyName, PropertyName, PropertyName,,,,
Commamd delete existing animation-tasks from the pool. If no one parameter
specified then command delete ALL tasks. Command can contain one or more names
of properties.
PropertyName - string that contain complete name of object and its property.
The property must be numeric.

Example1:
DeletePropAni   // delete all

Example2:
DeletePropAni "Master.Brightness"

Example3:
DeletePropAni "Master.Brightness",  "Master.Color",



==== MIDI Surface Layer (MSL) ====
```

BEYOND offer more functions that can be mapped to MIDI sliders/buttons of overage MIDI console has. The soluion is map a few function to one slider, and enable only one of them. For making this possible BEYOND intoduce conception of "layers". You organize some set of functions grouped into layers, and after that you can change the layer, what give quick access to the groups of functions.

MIDI Sufrace Layering allow connect multiple BEYOND features to one MIDI command (message). Layeing work with Main Grid, Secondary Grid, Sufrace Buttons, Surface Sliders, FX and Zone selection. Currently available 12 layers. Counting starts from 1 and up to 12. The function of BEYOND may belong to more than one layer. In this case this function stay active in all layers where it enabled. If the function of BEYOND is not enabled in current layer, then it will not react on assigned MIDI message, same as will not generate a feedback messages.

Simplified Layer control.

Each MIDI Mapping object has preperty Layer. When you write to this propetry then it change Layer of all tables. Such operation equal to using of 12 commands descrived below.

Example:
Midi1.Layer=1
Midi2.Layer=10
Midi3.Layer=5

Detailed control.

Each table has own Layer property that you can control independently.

==== Triggers ====

"Trigger" is a special mode for the scripter. The trigger as two part - Definition and Action. The the Definition part you define the type of the trigger, range of values and labels for corresponding sections of code. In the Action part has one or more code sections that will be activated - depending on the definition. Lets talk a bit more about when trigger may be in help and why it done this way.

The simplest and classic example is when DMX come into some range of values, and the fact that the value is in range now it create some reaction. If the is not the same as "if value is in range then we do something". No., It is like - we do it every time as value come into the range. The next time it will happen

when the value will go out of range and back to the range.

No doubt it is possible to make a trigger by means for "standard" PangoScrtipt commands, but it require much more complex script code, more lines, and at the end it will be more slow. The Trigger use a hybrid method. You define what to check and the ranges, and BEYOND software do the idle job for you - BEYOND read the state and compare with ranges, take care about the state and other things. All it done in native code. If you will do it all in PangoScript then it will require much more CPU time. So, this part can be more optimal. What say in PangoScript is a reaction (action). You need to define a section(s) of code that will be activated.

Trigger definition commands

DefineDmxTrigger ChannelIndex - this command set the scripter into trigger mode, and define that trigger react on DMX channel number "ChannelIndex "

DefineMidiTrigger Message, Data1 - this command set the scripter into trigger mode, and define that trigger react on MIDI message. You need to specify the message n

DefineTrigger String-Expression, Caption - this command set the scripter into trigger mode, and define that trigger depends on expression. The expression is math formula. it has a syntax of string. The dumb exampe: DefineTrigger "2+2". In this example 2+2 is expression. But, according to syntax of this command we put expression inside "..". The Caption is just text string for PangoScript tab

The top level logic of these command is this. The most possible, the triggers will work with MIDI and DMX consoles. The trigger must be fix, and effective, because it will work on a high speed, because we have a trigger specially made for MIDI and DMX. BEYOND precalculate values and do it all in native code. But, not doubt will appear a need in some universal method, and in this case will help universal command DefineTrigger. This command work with expression, it is more slow because BEYOND need to calculate expression all the time, but it is very flexible, and work for all types of input data. You can use objects, variables, functions, expression and so on. It can work with Audio, Kinect or DMX, Universe or GamePad and any mix of this. So, for DefineTrigger we need to supply text of expression to trigger engine, and it is a string.

Trigger range commands

Here a formal description of commands, examples and logic after that.

InRangeTrigger MinValue, MaxValue, LabelName

```
The action will be activated when values comes in range between MinValue and
MaxValue. When it happen the scripter does goto to LabelName
MinValue - number, a minimum value of the range
MaxValue - number, a maximum value of the range
LabelName - string that contain label name

InRangeTriggerCmd MinValue, MaxValue, Command
The action will be activated when values comes in range between MinValue and
MaxValue. When it happen the scripter execute Command.
MinValue - number, a minimum value of the range
MaxValue - number, a maximum value of the range
Command- string that contain a PangoScript command


OutOfRangeTrigger MinValue, MaxValue, LabelName
The action will be activated when values comes out of range of MinValue and
MaxValue. When it happen the scripter does goto to LabelName.
MinValue - number, a minimum value of the range
MaxValue - number, a maximum value of the range
LabelName - string that contain label name

OutOfRangeTriggerCmd MinValue, MaxValue, Command
The action will be activated when values goes out of range of MinValue and
MaxValue. When it happen the scripter execute Command.
MinValue - number, a minimum value of the range
MaxValue - number, a maximum value of the range
Command- string that contain a PangoScript command


IncreaseTrigger MinValue, MaxValue, LabelName
The action will be activated when value increase and stay in range of MinValue
and MaxValue. When it happen the scripter does goto to LabelName.
MinValue - number, a minimum value of the range
MaxValue - number, a maximum value of the range
LabelName - string that contain label name


DecreaseTrigger MinValue, MaxValue, LabelName
The action will be activated when value decrease and stay in range of MinValue
and MaxValue. When it happen the scripter does goto to LabelName.
MinValue - number, a minimum value of the range
MaxValue - number, a maximum value of the range
LabelName - string that contain label name

About the logic

The most simple is InRangeTrigger command. When values comes in range, then
something happen. We can define a few ranges, and when value comes into the
```

range then something will happen. What will happen? We considered a few options, and appeared that in simplest case one simple command will be enough. I mean, Blackout, or EnableLaserOutput, or StopAllCue , who knows. In many cases one command will be enough. But, what is not enough? I this case we should do goto to a second of code. In fact, you simply put a name of label, and BEYOND will do a goto (this process is optimized too). So, we have two options - Command or Label for Goto.

There is inverted version of InRangeTrigger - OutOfRange trigger. The only different is goes to be activated when value goes out of range. All the rest is equal. There in Min and Max value, and there version of command for Command and for Label.

The third variation of trigger allow define reaction on the increase or decrease of the value. it work like this: scripter memorize current value, and if the new value is bigger (or smaller) than current state then it active the trigger. Command IncreaseTrigger create a reaction when new value is more than current. Command DecreaseTrigger create a reaction when new value is less than current.

No doubt, we can add more commands, add more functionality to trigger. The range oriented commands is a classic. The increase/decrease has a practical use too.

Rule of 3 words: Type, Range, Interaction
T - type of the trigger - DMX, or MIDI, or universal
R - range, when value in range the we do something
I - interaction, the trigger is only a way to do some action, we need define what it will do.

Example 1: Enable/Disable laser output from MIDI

```
DefineMidiTrigger 0xB0, 0x00, "Enable Laser test"
InRangeTriggerCmd 0,63,"DisableLaserOutput"
InRangeTriggerCmd 64,127,"EnableLaserOutput"
```

This trigger react in the MDI slider 0xB0, 0x00, and has values in range of 0..127. When slider in lower half the laser output will be disables, and when in higher half, then output will be disables. Now command bby comamnd:

```
DefineMidiTrigger 0xB0, 0x00, "Enable Laser test"
DefineMidiTrigger  is  command
0xB0, 0x00 - this is slider parameter
"Enable Laser test"  - this is optional parameter - caption of this scripter


InRangeTriggerCmd 0,63,"DisableLaserOutput"
```

```
InRangeTriggerCmd - command, define the range and command
0,63 - the range
"DisableLaserOutput" - command as string.


InRangeTriggerCmd 64,127,"EnableLaserOutput"
InRangeTriggerCmd - command that define second range
64,127 - second range
"EnableLaserOutput" - command that will be executed.



Example 2: Same as example 1, but with goto

DefineMidiTrigger 0xB0, 0x00, "Enable Laser test 2"
InRangeTrigger 0,63, "D1"  // D1 is a name of  label, see below
InRangeTrigger 64,127, "D2" // D2 is a name of  label, see below
exit // we need to exit because we should stop script execution after the
declaration

D1:  // this is label used 1st range
DisplayPreview "Disable" // visual indication in Preview panel
DisableLaserOutput // Commamd
exit // we should stop script execution, otherwise it will do below

D2: // this is label used 2nd range
DisplayPreview "Ensable" // visual indication in Preview panel
EnableLaserOutput
exit // we should stop script execution. This is good practice to have it.

Commentaries for this example. The script itself is bigger, and there are
sections for definition and for corresponding scripts. There are two typical
mistakes
1. Do not forger to put exit instruction
2. Ensure that label name in code and in definition is the same. Otherwise, it
will not work. Label is case sensitive.

And general advice - use DisplayPreview, DisplayPopup or qlog for debug
purpose. It might be good idea to create a trigger definition with all jumps,
and put inside commands like DipsplayPreview only, without real functionality.
Once you see that the logic works fine, the put there the rest of code.



Example 3: Slider at first time do Enable Laser, and and second time Disable
Laser

This code based on previous, but it use variable. Variable organize a
```

```
DefineMidiTrigger 0xB0, 0x00, "Enable Laser test 3"
InRangeTrigger 64,127, "Trick1"
var counter // declare variable
counter=0 // initialize variable
exit


Trick1:
DisplayPreview Counter  // debug action
if (Counter=0) EnableLaserOutput  // action when Count is 0
if (Counter=1) DisableLaserOutput // action when Count is 1
Counter=(Counter+1) % 2 // increment counter and divide by modulus 2, so, it
will be 0,1,0,1,0,1 and so on
exit // stop this section


Example 4: The same as example 4, but with Goto

DefineMidiTrigger 0xB0, 0x00, "Enable Laser test 3"
InRangeTrigger 64,127, "Trick1"
var counter
counter=1
exit

Trick1:
DisplayPreview Counter
Counter=(Counter+1) % 2
if (Counter=0) goto When0
if (Counter=1) goto When1
exit

When0:
EnableLaserOutput
exit

When1:
DisableLaserOutput
exit

Well, example, is similar, and show that you can use additional Goto
instructions.

Example 5

What I want to demonstrate by this example - BEYOND does a goto when it
execute a trigger. It means this. You can do goto by means of script command.
But trigger engine use the same goto. Look a the code, comments below
```

```
DefineMidiTrigger 0xB0, 0x00, "Enable Laser test 3"
InRangeTrigger 64,127, "Trick1"
InRangeTrigger 0,63, "StopIt"
exit


Trick1:
DisplayPopup "Yes"
Sleep 1000
DisplayPopup "No"
Sleep 1000
goto Trick1

StopIt:
DisplayPopup "I dont care"
exit
```

Two ranges, and main action in range of 64..127. The main action is after label Trick1. It has a dead loop. At least, it looks like dead loop. You see message panel with Yes and No and it will work until you move the slider down. It will break this endless loop.

Example 6: Lets try to use expressions... BPM notification

```
DefineTrigger "Master.BPM" // expression...
InRangeTrigger 0,50,"WhenSlow"
InRangeTrigger 250,500,"WhenFast"
exit

WhenSlow:
DisplayPopup "Hey, dont sleep"
exit

WhenFast:
DisplayPopup "Hey, too fast!"
exit
```

This example use simplest expression - read the value of object property - Master.BPM, and depending on the BPM show the message.

Example 7: Out of range.

This example based on previous. The only change is OutOfRangeTrigger. This section will be activate when BPM is too slow or too fast. This is only example without big sense, but it it easy to test, because you can see BPM value on toolbar and have the slider that control it.

```
DefineTrigger "Master.BPM" // expression...
OutOfRangeTrigger 60,250,"WhenOutOf"
exit

WhenOutOf:
DisplayPopup "Hmm, this is not normal"
exit

Example 8: Starts/Stop timeline

DefineMidiTrigger 0xB0, 0x00, "Timeline control"
InRangeTriggerCmd  0, 63, "StopTimeline"
InRangeTriggerCmd 64,127, "PlayTimeline"
```