Satori Circulation 0.763 Beta
Scripting Documentation

last_update_20_feb

Syntax notes for this document

* generally p|pe is used for o|r - various possible cases :)
* italic font is used for *text_string*
* voluntary parameters placed in [this brackets]
* value# is numerical value in certain range if needed. range is described: value# = <min..max>
* value$ is a string value. string cases are described: value$ = <*string1* | *string2* | *string3*>
* value#$ is used for both cases: numerical or string value
* {default_value#} default value for the variable is noted this way
* commands are NOT case sensitive

Values used globally in this document

Position constant defines which source or posteffects will be controlled with your command:

```
position$ = <down, up, right, master>
```

respectively

```
position$ = <S1, S2, S3, MST>
gn#$ = <1..4> | off              // ALL gen(s) default off; off ~ 0
[on | off ] ~ [1 | 0]
```

timeline TIME

* absolute timeline time

`ti:me:co.de#` describes standard timecode form used in Circulation in format:

```
h#:m#:s#.ms#
h# = <0..23>
m# = <0..59>
s# = <0..59>
ms# = <0..999>
```

If you insert command on timeline without assigning a time you insert command to current time (and will be executed immediatelly). When the application is started the initial time is not a real computer time but it's started from 00:00:00.000. To set a real time set on computer to Circulation timeline use command:

goto, *<realtime>*

* relative timeline time

If you need to place command ahead in time you can use following format:

```
+ms#
ms# = <integer>
```

```
example:
log,hello              // hello is logged immediatelly
00:00:10.000,log,hello // hello will be logged 10s after startup of app
+5000,log,hello        // hello will be logged in 5 seconds
```

Macros

Macros are always used inside { ... } and first parameter is always macro command following by parameter(s).

* **FOR** macro

Whenever you type numerical value you can use {**for**} macro to generate more commands on timeline with interpolated value.

{ ***for***, from#, to#, step#, time# **}**

```
from#, to# = <0..1024>
step# = <0..512>   // no step < 0
time# = <0..65536> ms
```

```
example:
00:00:00.000,value,cross_fade,{for,255,0,32,1000}
```

output:
```
00:00:00.000,value,cross_fade,255
00:00:00.125,value,cross_fade,223
00:00:00.250,value,cross_fade,191
00:00:00.376,value,cross_fade,159
00:00:00.501,value,cross_fade,127
00:00:00.627,value,cross_fade,95
00:00:00.752,value,cross_fade,63
00:00:00.878,value,cross_fade,31
00:00:01.000,value,cross_fade,0
```

* **RND** macro
Random value in interval <min#..max#> is generated via {**rnd**} macro.
{ ***rnd***, min#, max# **}**
min# = <-512..512>
max# = <-512..512>

* **STR** macro
You can define strings by **set** command (see below) and insert them later in script by:
{ ***str***, string_name$, **}**

Software control

Apply **end** commands two times to quit Circulation.
end

Log text
log, text$

Image screenshot activation
screen_snap

Preset snap activation
state_snap

Actual time+date
time | **date**

This command gives you a chance to write more commands in one line. Use character ';'
to separate commands. Very usefull for **trigger** to execute more commands in single line
without need to define another file (but for more commands it's recommended to call more
commands from external file with **insert** or **insert_rand** command).
Every command may start with timeline time in relative or absolute form!
cmd, [ti:me:co:de# | +ms#], command1$, command1param(s)$ [; [ti:me:co:de# | +ms#], command2$, command2param(s)$]

example:
```
cmd,00:01:00.000,log,first minute passed;+5000,log,1st minute and 5th second passed
// see trigger command example for more
```

At this moment **set** is used to define string constants:
set, type$, string_name$, data$
type$ = <*str*>

example:
```
set,str,text1,posteffect,down,strobo,on
cmd,{str,text1}
```

Timeline control commands

start
timeline_delete_up, [
goto, [ti:me:co:de# | +/-ms# | <*realtime*>]
restart
insert
insert_rand

command: mixer

syntax:
mixer, variable_name$, number#

list of mixer command variables:

Note: **fmn** = number of fader maps defined

variable name	value min	value max	alternative contants
cross_fade	0	255	
cross_fade_gen	0	1	*on/off*
cross_fade2	0	255	
cross_fade2_gen	0	1	*on/off*
fader_map	0	fmn-1	
fader_map_3rd	0	1	*on/off*
fader2_map	0	fmn-1	
blur	0	127	
blur_gen	0	1	*on/off*
contrast	0	127	
contrast_gen	0	1	
brightness	0	127	
brightness_gen	0	1	*on/off*
colorize	0	127	
colorize_gen	0	1	*on/off*
colorize_color	0	255	
colorize_color_gen	0	1	*on/off*
key_down	0	255	
key_up	0	255	
key_color	0	255	
cross_fade_gen	0	1	*on/off*
cross_fade_gen_rand	0	127	
cross_fade_gen_sin	0	127	
cross_fade_gen_freq	0	127	
cross_fade_gen_hold	0	127	
cross_fade2_gen	0	1	*on,off*
cross_fade2_gen_rand	0	127	
cross_fade2_gen_sin	0	127	
cross_fade2_gen_freq	0	127	
cross_fade2_gen_hold	0	127	
mouse_gen	0	1	*on,off*
mouse_gen_rand	0	127	
mouse_gen_sin	0	127	
mouse_gen_freq	0	127	
mouse_gen_hold	0	127	
gen_on	0	1	*on,off*
gen_rand	0	127	
gen_sin	0	127	
gen_freq	0	127	
gen_hold	0	127	
time_distort_type	0	3	*off,norm,fast,framemap*
time_distort	0	255	
time_distort_gen	0	1	*on,off*
time_distort_map	0	fmn-1	
fader_invert	0	1	*on,off*
fader_outline	0	1	*on,off*
fader_outline_fade	0	1	*on,off*
fader_outline_fade_dir	-1	1	*up,down*
fader_rotate	0	1	*on,off*
fader_rotate_add	0	127	
fader_rotate_dir	-1	1	*up,down*
fader_seq_play	0	1	*on,off*
fader_seq_delay	0	127	
fader2_invert	0	1	*on,off*
fader2_outline	0	1	*on,off*
fader2_outline_fade	0	1	*on,off*
fader2_outline_fade_dir	-1	1	*up,down*
fader2_rotate	0	1	*on,off*
fader2_rotate_add	0	127	
fader2_rotate_dir	-1	1	*up,down*
fader2_seq_play	0	1	*on,off*

```
fader2_seq_delay          |   0   |  127  |
fader_key_level           |   0   |  127  |
fader2_key_level          |   0   |  127  |
blur_seq_play             |   0   |   1   | on,off
fader_outline_r           |   0   |  255  |
fader_outline_g           |   0   |  255  |
fader_outline_b           |   0   |  255  |
fader2_outline_r          |   0   |  255  |
fader2_outline_g          |   0   |  255  |
fader2_outline_b          |   0   |  255  |
stripes                   |   0   |   1   | on,off
gray                      |   0   |   1   | on,off
key_level_clear           |   0   |   1   | on,off
=============================================================
```

other mixer command:

fader, fader2, fader_param, fader2_param, fader_seq, fader2_seq, blur, blur_seq

syntax:
fader, fader_name$
fader_param, fader_param$
fader2, fader_name$
fader2_param, fader_param$
fader_seq, seq#, fader_name$
fader_param_seq, seq#, fader_param$
fader2_seq, seq#, fader_name$
fader2_param_seq, seq#, fader_param$
blur, blur_name#
blur_seq, seq#, blur_name$
fader_name$ = <*cross*,*lumakey*,*lumashade*,*colorize*,
add,*addsatur*,*dif*,*offset*,
map,*mapalfa*,*mapalfarot*,*movebox*,
lumacut,*lumacutinv*,*strobo*,*flash*>
blur_name$ = <*motion*,*lumasatur*,*luma*,*horizontal*,*vertical*,*gausian*>
seq# = <0..7>
Note: fader_param$ are specific for each selected fader and will be described in future version of documentation.

example:
```
fader,add
mixer,cross_fade,64
fader2,colorize
mixer,cross_fade2,128
blur,motion
mixer,blur,64
```

command: posteffect

Command **posteffect** is used to control parameters of posteffects primitives.
syntax:
posteffect, position$, >posteffect_command<

>**posteffect_command**< definitions:

Resets posteffects to defaults.
init ; reset

example:
```
posteffect, S1, init
posteffect, S2, reset
posteffect, right, init
```

Make a negation of image.
nega, [*on* | *off*] {*off*}

Stroboscopic change of normal image and it's negative.
strobo_nega, [*on* | *off*] {*off*}; **strobo**, [*on* | *off*] {*off*}
Number of delay frames + 1 to toggle strobo state.
strobo_delay, delay#
You can set a smooth transition between normal image and it's negative.
strobo_xfd, *on* | *off* {*off*}
delay# = <0..255> {0}

Flip/mirror image in horizontal/vertical direction.
flip_hor, [*on* | *off*] {*off*}
flip_ver, [*on* | *off*] {*off*}
mirror_hor, [*on* | *off*] {*off*}
mirror_ver, [*on* | *off*] {*off*}
mirror_hor_cross, [*on* | *off*] {*off*}
mirror_ver_cross, [*on* | *off*] {*off*}

Zoom image into center.
zoom, [*on* | *off* | size#] {*on*}
zoom_size, size#
zoom_size_gen, gn#
size# = <0..320> {0}

200% zoom with mouse controlled position or to given x, y.
zoom2x, [*on* | *off*] {*off*}
zoom2x_mouse, [*on* | *off*] {*on*}
zoom2x_pos_x, pos_x#
zoom2x_pos_y, pos_y#
pos_x# = <0..639> {320}
pos_y# = <0..479> {240}

Pixelate image to given size of block.
mosaic, [*on* | *off* | size#] {*off*}
mosaic_size, size#
mosaic_size_gen, gn#
size# = <0..255> {0}

TODO: mosain8, mosaic16

Image vectorizing.
vectorize, [*off* | *on* | level#], [type$] {*off*}
vectorize_type, [type$] {*flat_luma*}
vectorize_level, level#
vectorize_level_gen, gn#
type$ = <*wire_luma*, *flat_luma*> {*flat_luma*}
level# = <200..1000> {200}

Basic blurring.
blur, [*off* | *on* | size#] {*off*}
blur_size, size#
size# = <1..6> {1}

New position of the image or shift in given direction per processed frame.
pos_x, [pos_x#]
pos_y, [pos_y#]
shift_x, [sht_x#]
shift_y, [sht_y#]
shift_delay, [delay#]
pos_x# = <0..639>
pos_y# = <0..479>
sht_x# = <-64..64>
sht_y# = <-64..64>
delay# = <0..255>

Gray scale conversion.
gray, [*on* | *off*] {*off*}

Front and back colors for duo tone effects (trashhold, dither).
low_color, r#, g#, b# ; **bcgr_color**, r#, g#, b#
high_color, r#, g#, b# ; **frnt_color**, r#, g#, b#
r# = <0..255>
g# = <0..255>
b# = <0..255>

Image is set to back color belov level# and to high color above it.
trashhold, [*on* | *off* | level#], [mix$] {*off*}
trashhold_mix, mix$ {*luma*}
trashhold_level, level#
trashhold_gen, gn#
mix$ = <*luma* | *rgb*>
level# = <0..255> {128}

Lowers continuity of color space.
posterize, [*off* | level#]
posterize_level, level#
posterize_level_gen, gn#
level# = <0..255>

Duo tone dithering with various masks.
dither, [*on* | *off* | level#], [type#] {*off*}
dither_type, type#
dither_level, level#
dither_level_gen, gn#
type# = <0..4> {0}
level# = <0..127> {0}

A lens controlled by mouse position.
lens, [*on* | *off* | level#] {*off*}
lens_level, level#
level# = <0..255> {64}

Twirl image controlled by mouse position.
twirl, [on | off | level#]
twirl_level, level#
level# = <0..255> {64}

Simplified ascii image conversion with selection of various font.
font, [on | off | face#], [mix$] {*off*}
font_face, face#
font_mix, mix$
face# = <0..30>
mix$ = <*luma*, *rgb*> <*luma*>

Rotates the color space of the image.
hue, [*on* | *off* | level#] {*off*}
hue_level, level#
hue_level_gen, gn#
level# = <0..383> {0}

Set the level of RGB image components.
rgb_level, r#, g#, b#
red_level, r#
green_level, g#
blue_level, b#
red_level_gen, gn#
green_level_gen, gn#
blue_level_gen, gn#
r# = <0..255> {255}
g# = <0..255> {255}
b# = <0..255> {255}
// alternatives:
red_off ; red_25% ; red_50% ; red_75% ; red_on
green_off ; green_25% ; green_50% ; green_75% ; green_on
blue_off ; blue_25% ; blue_50% ; blue_75% ; blue_on

Elevates vertically pixels according to luma leve in various modesl.
vertical3d, [*on* | *off* | level#], [type$] {*off*}
vertical3d_type, type$
vertical3d_size, size#
vertical3d_size_gen, gn#
type$ = <*line_luma*, *line_rgb*, *wire_luma*, *wire_rgb*, *alfa_luma*> {*line_luma*}
size# = <0..255> {0}

Extrude image in central radial manner according to luma level.
3dsphere, [*off* | level#], [type$] {*off*}
3dsphere_type, type$
3dsphere_size, size#
3dsphere_size_gen, gn#
type$ = <*light*, *dark*> {*light*}
size# = <0..255> {0}

emboss, [*on* | *off* | type$] {*off*}
emboss_type, type$
emboss_level, level#
emboss_x, pos_x#
emboss_y, pos_y#
type$ = <*norm*, *light*, *dark*> {*norm*}
level# = <0..127> {127}
pos_x# = <-32..32> {0}
pos_y# = <-32..32> {0}

Makes difference of last 2 states of rendering.
difference, [*on* | *off* | type$] {*off*}
difference_type, type$
type$ = <*norm*, *light*, *dark*> {*norm*}

Pixelize image according to luma in various modes.
luma_pixel, [*on* | *off* | size#], [type$] {*off*}
luma_pixel_type, type$
luma_pixel_size, size#
luma_pixel_size_gen, gn#
type$ = <*light_luma*, *dark_luma*, *light_rgb*, *dark_rgb*> {*light_luma*}
size# = <0..255> {0}

Blurs image according to luma.
luma_blur, [*on* | *off* | size#], [type$] {*off*}
luma_blur_type, type$
luma_blur_size, size#
luma_blur_size_gen, gn#
type$ = <*light_luma*, *dark_luma*, *light_rgb*, *dark_rgb*> {*light_luma*}
size# = <0..255> {0}

Blur according to luma value of emboss algorythm.
emboss_blur, [*on* | *off* | size#], [type$] {*off*}
emboss_blur_size, size#
emboss_blur_size_gen, gn#
size# = <0..255> {0}

pixelate_interpol, [*on* | *off* | size#] {*off*}
pixelate_interpol_size, size#
pixelate_interpol_size_gen, gn#
size# = <0..255> {0}

Pixel position distorion generated from 2 gray-scale bitmaps.
offset, [*on* | off | level#] {*off*}
offset_antialias, [*on* | *off*] {*on*}
offset_map, map#
offset_level, level#
offset_level_gen, gn#
map# = <0..(num_maps# - 1)> {0}
level# = <0..255> {255}

Gausian blur (unoptimised at the moment).
gausian, [*on* | *off* | size#] {*off*}
gausian_size, size#
gausian_size_gen, gn#
size# = <0..32> {0}

modulo, [*off* | *on* | type#] {*off*}
modulo_type, type#
type# = 0 {0}

Calculates contours at given step.
countour, [*off* | step#], [type$] {*off*}
countour_type, type$
countour_step, step#
countour_step_gen, gn#
type$ = <*cntr_luma*, *light_luma*, *dark_luma*, *cntr_rgb*, *light_rgb*, *dark_rgb*> {*cntr_luma*}
step# = <16..255> {64}

Feedback with selectable pipeline position.
feedback, [*on* | *off* | level#], [fader$] {*off*}
feedback_fader, fader$
feedback_level, level#
feedback_level_dir, dir$
feedback_level_gen, gn#
fader$ = <*cross*, *luma_key*, *lumashade*, *colorize*, *add*, *addsatur*, *dif*, *offset*, *lumacut*, *lumacutinv*> {*cross*}
level# = <0..255> {0}
dir$ = <*up*, *down*> {*up*}

Merge is used for combination of 2 processing states at posteffects pipeline.
merge, [*on* | *off* | level#], [fader$] {*off*}
marge_fader, fader$
merge_level, level#
merge_level_dir, dir$
merge_level_gen, gn#
fader$ = <*cross*, *luma_key*, *lumashade*, *colorize*, *add*, *addsatur*, *dif*, *offset*, *lumacut*, *lumacutinv*> {*cross*}
level# = <0..255> {0}
dir$ = <*up*, *down*> {*up*}

Gamma curve correction.
gamma, [*on* | *off* | level#] {*off*}
gamma_level, level#
gamma_level_gen, gn#
level# = <0..255> {128}

Blending input for selected number of frames and holding output for the same time with previous result.
gapper, [*on* | *off* | level#] {*off*}
gapper_frames, level#
gapper_xfd, [*on* | *off*] {*on*}
level# = <2..64> {8}

Control ammount of color in image.
saturation, [*on* | *off* | level#] {*off*}
saturation_level, level#
saturation_level_gen, gn#
level# = <0..127> {64}

3dplane, [*on* | *off*] {*off*}
3dplane_angle_x, angle# {256}
3dplane_angle_y, angle# {256}
3dplane_angle_z, angle# {256}
3dplane_angle_gen, gn#
3dplane_angle_control, a$
3dplane_scale, scale#
3dplane_pos_x, pos# {0}
3dplane_pos_y, pos# {0}
angle# = <0..511>
a$ = <*x*, *y*, *z*> {*x*}

scale# = <0..255> {64}
pos# = <-64..64>
num# = <0..2> {2}

raster, [*on* | *off* | type#] {*off*}
raster_type, type#
type# = <0..7> {0}

stretch, [*on* | *off* | level#], [type$] {*off*}
stretch_type, type$
stretch_level, level#
stretch_level_gen, gn#
type$ = <*hor*, *ver*, *skew_hor*, *skew_ver*> {*hor*}
level# = <0..255> {128}

histodistort, [*on* | *off* | level#]
histodistort_level, level#
histodistort_level_gen, gn#
level# = <0..320> {0}

tile, [*on* | *off* | size#] {*off*}
tile_size, size#
tile_size_gen, gn#
size# = <0..255>

kaleido, [*on* | *off* | size#], [type$] *{off}*
kaleido_type, type$
kaleido_size, size#
kaleido_size_gen, gn#
kaleido_rot, rot#
kaleido_pos_x, pos# {0}
kaleido_pos_y, pos# {0}
type$ = <*v1*, *v2*> {*v1*}
size# = <32, 480> {256}
rot# = <-64, 64>
pos# = <-64, 64>

anaglyph, [*on* | *off* | level#], [type$] {*off*}
anaglyph_type, type$
anaglyph_level, level#
anaglyph_level_gen, gn#
level# = <0..320> {0}
type$ = <*r-b*, *r-g*> {*r-b*}

copper, [*on* | *off* | level#], [type$] {*off*}
copper_type, type$
copper_level, level#
copper_level_gen, gn#
copper_shade, [*on* | *off*] {*off*}
type$ = <*left*, *right*, *top*, *bottom*, *all*, *left_right*, *top_bottom*> {*left*}
level# = <0..255> {0}

To configure generators:
gen, gn#, **rand**, val#
gen, gn#, **sin**, val#
gen, gn#, **freq**, val#
gen, gn#, **hold**, val#
gn# = <1..4>
val# = <0..127>

pipeline, effect$, pos_x#, pos_y#
effect$ = < *screeshot*, *merge_in*, *feedback_out*, *raster*, *lens_twirl*, *flip*,
mirror, *mirror_smooth*, *zoom2x*, *zoom*, *mosaic*, *mosaic_8/16*, *vectorize*,
blur, *gausian*, *gray*, *emboss*, *nega*, *strobo*, *merge_swap*, *threshold*,
posterize, *dither*, *font*, *modulo*, *vertical3d*, *luma_pixel*, *luma_blur*,
emboss_blur, *pixelate_interpol*, *offset*, *tile*, *shift*, *difference*,
hue, *saturation*, *levels*, *3dsphere*, *3dplane*, *stretch*, *countour*, *histo_dist*,
feedback_in, *gapper*, *anaglyph*, *gamma*, *kaleido*, *copper*, *fader_snap*, *merge_out* >

```
pos_x# = <0..360>
pos_y# = <0..360>

example:
posteffect,down,init
posteffect,down,vertical3d,on
posteffect,down,vertical3d_size,32
posteffect,down,gen,1,rand,32
posteffect,down,vertical3d_size_gen,1
posteffect,down,luma_blur,32,light_rgb
posteffect,S2,zoom_size,{0,320,1,5000}
```

command: combo

Command **combo** is used to control combo section of posteffects.

syntax:

combo, position$, >combo_command<

>**combo_command**< definitions:

select, [num# | *all*], [*on* | *off*]
num# = [0..49]

You can activate certain combo by it's position number with this command. A special case is to activate randomly (***rnd***) choosen combo (from all or either only selected - ***rnd_sel***) and last case is to activate ***def***ault posteffects state (usefull for fading out combos to default state of posteffects)

apply, [num# | *rnd* | *rnd_sel* | *def*]
num# = [0..49]

autochange, [*on* | *off*]
autochange_time, sec#
autochange_mode, mode$
num# = <0..49>
sec# = <1..255>
mode$ = <*upward* | *downward* | *pingpong* | *rand*>
autochange_fade, [on | off]
autochange_fade_delux, [on | off]

TO-DO: rest of params

```
example:
combo,S1,select,2,on      // selects 3rd combo as active in posteffects1
combo,S2,select,all,off   // unselects all combos in posteffects2
```

command: source

Source command is used to select which source plugin is loaded where

syntax:

source, position$, source_name$
source_name$ = <*aviplay*, *picblur*, *shifter*, *black*, *noise*, *capture*, *capture4x*, *typer*, *slider*, *render*, *swfplay*>

to unload plugin from certain source use this syntax

source, position$, <*none*>

```
example:
source,down,aviplay
source,s2,black
source,right,<none>
```

command: state

State command is used to configure source plugin in desired position.

syntax:

state, position$, >state_command<, >state_param(s)<

>state_command< and >state_param(s)< are always specific according to source activated in certain position.

>**state_command**< definitions:

* AVIPLAY

file, filename$ | name$
frame, num#
dir_forward | **dir_backward**
loop | **noloop** | **pingpong**
deinterlace
rewind_forward, speed#
rewind_backward, speed#
bank, num#
num# = <0..99>

* PICBLUR

pic, filename$
pos_x, posx#
posx# = <0.0..1.0>
pos_y, posy#
post# = <0.0..1.0>
shift_size, shift#
rand_size, rand#
rot_size, rot#
rot_pos, pos#
blur, blur#
blur# = <0..255>

* SHIFTER

pos_x, pos_x#
pos_y, pos_y#
step_x, step_x#
step_y, step_y#
fade_speed, speed#
fade_satur, satur#
wrap
hold
clear_count, frames#

* RENDER

store_point, [*mixer* | *output*]
render_pos, pos#
pos# = <0..7>

* BLACK

red_level, level#
green_level, level#
blur_level, level#
level# = <0..255>

command: trigger

Trigger command is used to register "guard" on certain input action and **trigger_remove** for unregister.
syntax:
trigger, trigger_type$, trigger_param(s)$#, >command(s)<
trigger_type$ = < *time*, *key_down*, *key_up*, *joy_button_down*, *joy_button_up*,
module_instrument, *module_position* >
//TO-DO: midi, OSC, joy_analog

* **TIME** trigger

Time trigger is executed ONCE the configured time is reached. Trigger time can be written in timecode format or
in number of milliseconds.
syntax:
trigger, *time*, [ti:me:co:de# | ms#], >command(s)<

example:
```
set,str,mystring1,log,second minute;+5000,log,hello 02:05
trigger,time,00:01:00.000,log,first minute
trigger,time,120000,log,cmd,{str,mystring1}
```

* **KEY_DOWN / KEY_UP** trigger

Keyboard triggers works only in INTERACTIVE OFF mode (Capslock is turned ON).
Commands will be executed after press or release of certain key letter or number.
syntax:
trigger, key_state$, key$, >command(s)<
key_state$ = <*key_down*, *key_up*>
key$ = <A..Z | 0..9>

If you need to remove certain trigger use **trigger_remove** command and place the same parameters except executings commands as removal parameters. For example for the keyboard trigger.
trigger_remove, key_state$, key$
will remove all triggers with configured attributes

example:
```
trigger,key_down,Q,value,cross_fade,{rnd,0,255}
trigger,key_down,W,cmd,insert_rand,scripts_folder;value,cross_fade2,{for,0,255,2,2000}
trigger,time,00:01:00.000,trigger_remove,key_down,Q
```

* **JOY_BUTTON_DOWN** / **JOY_BUTTON_UP** trigger

Usefull trigger might be with wireless joysticks to have chance and design setup for your joystick.
You can execute action after press or release of certain joystick button.
syntax:
trigger, joy_button_state$, button#, >command(s)<
joy_button_state$ = <*joy_button_down*, *joy_button_up*>
joy_button# = <1..31>

example:
```
trigger,joy_button_down,1,combo,S1,apply,rnd
trigger,joy_button_up,1,combo,S1,apply,def
```