Teletype v5.0.0 Documentation

Variables

A / A x
get / set the variable A, default 1

B / B x
get / set the variable B, default 2

C / C x
get / set the variable C, default 3

D / D x
get / set the variable D, default 4

FLIP / FLIP x
returns the opposite of its previous state (0 or 1) on each read (also settable)

I / I x
get / set the per-script variable I. *See also L : in control flow*

J / J x
get / set the per-script variable J

K / K x
get / set the per-script variable K

O / O x
auto-increments *after* each access, can be set, starting value 0

O.INC / O.INC x
how much to increment O by on each invocation, default 1

O.MIN / O.MIN x
the lower bound for O, default 0

O.MAX / O.MAX x
the upper bound for O, default 63

O.WRAP / O.WRAP x
should O wrap when it reaches its bounds, default 1

T / T x
get / set the variable T, typically used for time, default 0

TIME / TIME x
timer value, counts up in ms., wraps after 32s, can be set

TIME.ACT / TIME.ACT x
enable or disable timer counting, default 1

LAST x
get value in milliseconds since last script run time

X / X x
get / set the variable X, default 0

Y / Y x
get / set the variable Y, default 0

Z / Z x
get / set the variable Z, default 0

Hardware I/O

CV x / CV x y
CV target value

CV.OFF x / CV.OFF x y
CV offset added to output

CV.SET x y
Set CV value, ignoring slew

CV.GET x
Get current CV value

CV.SLEW x / CV.SLEW x y
Get/set the CV slew time in ms

V x
converts a voltage to a value usable by the CV outputs (x between 0 and 10)

VV x
converts a voltage to a value usable by the CV outputs (x between 0 and 1000, 100 represents 1V)

IN
Get the value of IN jack (0-16383)

IN.SCALE min max
Set static scaling of the IN CV to between min and max.

PARAM PRM
Get the value of PARAM knob (0-16383)

PARAM.SCALE min max
Set static scaling of the PARAM knob to between min and max.

TR x / TR x y
Set trigger output x to y (0-1)

TR.PULSE x TR.P
Pulse trigger output x

TR.TIME x / TR.TIME x y
Set the pulse time of trigger x to y ms

TR.TOG x
Flip the state of trigger output x

TR.POL x / TR.POL x y
Set polarity of trigger output x to y (0-1)

MUTE x / MUTE x y
Disable trigger input x

STATE x
Read the current state of input x

LIVE.OFF LIVE.O
Show the default live mode screen

LIVE.VARS LIVE.V
Show variables in live mode

LIVE.GRID LIVE.G
Show grid visualizer in live mode

LIVE.DASH x LIVE.D
Show the dashboard with index x

PRINT x / PRINT x y PRT
Print a value on a live mode dashboard or get the printed value

Pitch

HZ x
converts 1V/OCT value x to Hz/Volt value, useful for controlling non-euro synths like Korg MS-20

JI x y
just intonation helper, precision ratio divider normalised to 1V

N x
converts an equal temperament note number to a value usable by the CV outputs (x in the range -127 to 127)

N.S r s d
Note Scale operator, r is the root note (0-127), s is the scale (0-8) and d is the degree (1-7), returns a value from the N table.

N.C r c d
Note Chord operator, r is the root note (0-127), c is the chord (0-12) and d is the degree (0-3), returns a value from the N table.

N.CS r s d c
Note Chord Scale operator, r is the root note (0-127), s is the scale (0-8), d is the scale degree (1-7) and c is the chord component (0-3), returns a value from the N table.

N.B d / N.B r s
get degree d of scale/set scale root to r, scale to s, s is either bit mask (s >= 1) or scale preset (s < 1)

N.BX i d / N.BX i r s
multi-index version of N.B, scale at i (index) 0 is shared with N.B

VN x
converts 1V/OCT value x to an equal temperament note number

QT.B x
quantize 1V/OCT signal x to scale defined by N.B

QT.BX i x
quantize 1V/OCT signal x to scale defined by N.BX in scale index i

QT.S x r s
quantize 1V/OCT signal x to scale s (0-8, reference N.S scales) with root 1V/OCT pitch r

QT.CS x r s d c
quantize 1V/OCT signal x to chord c (1-7) from scale s (0-8, reference N.S scales) at degree d (1-7) with root 1V/OCT pitch r

Rhythm

BPM x
milliseconds per beat in BPM x

DR.P b p s
Drum pattern helper, b is the drum bank (0-4), p is the pattern (0-215) and step is the step number (0-15), returns 0 or 1

DR.T b p q l s
Tresillo helper, b is the drum bank (0-4), p is first pattern (0-215), q is the second pattern (0-215), l is length (1-64), and step is the step number (0-length-1), returns 0 or 1

DR.V p s
Velocity helper. p is the pattern (0-19). s is the step number (0-15)

ER f l i
Euclidean rhythm, f is fill (1-32), l is length (1-32) and i is step (any value), returns 0 or 1

NR p m f s
Numeric Repeater, p is prime pattern (0-31), m is & mask (0-3), f is variation factor (0-16) and s is step (0-15), returns 0 or 1

Metronome

M / M x
get/set metronome interval to x (in ms), default 1000, minimum value 25

M! / M! x
get/set metronome to experimental interval x (in ms), minimum value 2

M.ACT / M.ACT x
get/set metronome activation to x (0/1), default 1 (enabled)

M.RESET
hard reset metronome count without triggering

Randomness

RAND x RND
generate a random number between 0 and x inclusive

RRAND x y RRND
generate a random number between x and y inclusive

TOSS
randomly return 0 or 1

R / R x
get a random number/set R.MIN and R.MAX to same value x (effectively allowing R to be used as a global variable)

R.MIN x
set the lower end of the range from -32768 – 32767, default: 0

R.MAX x
set the upper end of the range from -32768 – 32767, default: 16383

CHAOS x
get next value from chaos generator, or set the current value

CHAOS.R x
get or set the R parameter for the CHAOS generator

CHAOS.ALG x
get or set the algorithm for the CHAOS generator. 0 = LOGISTIC, 1 = CUBIC, 2 = HENON, 3 = CELLULAR

DRUNK / DRUNK x
changes by -1, 0, or 1 upon each read saving its state, setting will give it a new value for the next read

DRUNK.MIN / DRUNK.MIN x
set the lower bound for DRUNK, default 0

DRUNK.MAX / DRUNK.MAX x
set the upper bound for DRUNK, default 255

DRUNK.WRAP / DRUNK.WRAP x
should DRUNK wrap around when it reaches it's bounds, default 0

SEED / SEED x
get / set the random number generator seed for all SEED ops

RAND.SEED / RAND.SEED x RAND.SD R.SD
get / set the random number generator seed for R, RRAND, and RAND ops

TOSS.SEED / TOSS.SEED x TOSS.SD
get / set the random number generator seed for the TOSS op

PROB.SEED / PROB.SEED x PROB.SD
get / set the random number generator seed for the PROB mod

DRUNK.SEED / DRUNK.SEED x DRUNK.SD
get / set the random number generator seed for the DRUNK op

P.SEED / P.SEED x P.SD
get / set the random number generator seed for the P.RND and PN.RND ops

Control flow

`IF x: ...`
if x is not zero execute command

`ELIF x: ...`
if all previous `IF` / `ELIF` fail, and x is not zero, execute command

`ELSE: ...`
if all previous `IF` / `ELIF` fail, excute command

`L x y: ...`
run the command sequentially with `I` values from x to y

`W x: ...`
run the command while condition x is true

`EVERY x: ...` `EV`
run the command every x times the command is called

`SKIP x: ...`
run the command every time except the xth time.

`OTHER: ...`
runs the command when the previous EVERY/SKIP did not run its command.

`SYNC x`
synchronizes *all* EVERY and SKIP counters to offset x.

`PROB x: ...`
potentially execute command with probability x (0-100)

`SCRIPT / SCRIPT x` `$`
get current script number, or execute script x (1-10), recursion allowed

`SCRIPT.POL x / SCRIPT.POL x p` `$.POL`
get script x trigger polarity, or set polarity p (1 rising edge, 2 falling, 3 both)

`$F script`
execute script as a function

`$F1 script param`
execute script as a function with 1 parameter

`$F2 script param1 param2`
execute script as a function with 2 parameters

`$L script line`
execute script line

`$L1 script line param`
execute script line as a function with 1 parameter

`$L2 script line param1 param2`
execute script line as a function with 2 parameters

`$S line`
execute script line within the same script as a function

`$S1 line param`
execute script line within the same script as a function with 1 parameter

`$S2 line param1 param2`
execute script line within the same script as a function with 2 parameters

`I1`
get the first parameter when executing a script as a function

`I2`
get the second parameter when executing a script as a function

`FR / FR x`
get/set the return value when a script is called as a function

`SCENE / SCENE x`
get the current scene number, or load scene x (0-31)

`SCENE.G x`
load scene x (0-31) without loading grid control states

`SCENE.P x`
load scene x (0-31) without loading pattern state

`KILL`
clears stack, clears delays, cancels pulses, cancels slews, disables metronome

`BREAK` `BRK`
halts execution of the current script

`INIT`
clears all state data

`INIT.CV x`
clears all parameters on CV associated with output x

`INIT.CV.ALL`
clears all parameters on all CV's

`INIT.DATA`
clears all data held in all variables

`INIT.P x`
clears pattern number x

`INIT.P.ALL`
clears all patterns

`INIT.SCENE`
loads a blank scene

`INIT.SCRIPT x`
clear script number x

`INIT.SCRIPT.ALL`
clear all scripts

`INIT.TIME x`
clear time on trigger x

`INIT.TR x`
clear all parameters on trigger x

`INIT.TR.ALL`
clear all triggers

Maths

`ADD x y` `+`
add x and y together

`SUB x y` `-`
subtract y from x

`MUL x y` `*`
multiply x and y together

`DIV x y` `/`
divide x by y

`MOD x y` `%`
find the remainder after division of x by y

`? x y z`
if condition x is true return y, otherwise return z

`MIN x y`
return the minimum of x and y

`MAX x y`
return the maximum of x and y

`LIM x y z`
limit the value x to the range y to z inclusive

`WRAP x y z` `WRP`
limit the value x to the range y to z inclusive, but with wrapping

`QT x y`
round x to the closest multiple of y (quantise)

`AVG x y`
the average of x and y

`EQ x y` `==`
does x equal y

`NE x y` `!= XOR`
x is not equal to y

`LT x y` `<`
x is less than y

`GT x y` `>`
x is greater than y

`LTE x y` `<=`
x is less than or equal to y

`GTE x y` `>=`
x is greater than or equal to y

`INR l x h` `><`
x is greater than `l` and less than h (within range)

`OUTR l x h` `<>`
x is less than `l` or greater than h (out of range)

`INRI l x h` `>=<`
x is greater than or equal to `l` and less than or equal to h (within range, inclusive)

`OUTRI l x h` `<=>`
x is less than or equal to `l` or greater than or equal to h (out of range, inclusive)

`EZ x` `!`
x is `0`, equivalent to logical NOT

`NZ x`
x is not `0`

`LSH x y` `<<`
left shift x by y bits, in effect multiply x by 2 to the power of y

`RSH x y` `>>`
right shift x by y bits, in effect divide x by 2 to the power of y

`LROT x y` `<<<`
circular left shift x by y bits, wrapping around when bits fall off the end

`RROT x y` `>>>`
circular right shift x by y bits, wrapping around when bits fall off the end

`| x y`
bitwise or x | y

`& x y`
bitwise and x & y

`x y`
bitwise xor x ^ y

`~ x`
bitwise not, i.e.: inversion of x

`BSET x y`
set bit y in value x

`BGET x y`
get bit y in value x

`BCLR x y`
clear bit y in value x

`BTOG x y`
toggle bit y in value x

`BREV x`
reverse bit order in value x

`ABS x`
absolute value of x

`AND x y` `&&`
logical AND of x and y

`AND3 x y z` `&&&`
logical AND of x, y and z

`AND4 x y z a` `&&&&`
logical AND of x, y, z and a

`OR x y` `||`
logical OR of x and y

`OR3 x y z` `|||`
logical OR of x, y and z

`OR4 x y z a` `||||`
logical OR of x, y, z and a

`SCALE a b x y i` `SCL`
scale `i` from range a to b to range x to y, i.e. `i * (y - x) / (b - a)`

`SCALE a b i` `SCL0`
scale `i` from range `0` to a to range `0` to b

`EXP x`
exponentiation table lookup. `0-16383` range (V `0-10`)

`SGN x`
sign function: 1 for positive, -1 for negative, 0 for 0

Delay

`DEL x: ...`
Delay command by x ms

`DEL.CLR`
Clear the delay buffer

`DEL.X x delay_time: ...`
Delay x commands at `delay_time` ms intervals

`DEL.R x delay_time: ...`
Trigger the command following the colon once immediately, and delay `x - 1` commands at `delay_time` ms intervals

`DEL.G x delay_time num denom: ...`
Trigger the command once immediately and `x - 1` times at ms intervals of `delay_time * (num/denom)^n` where n ranges from 0 to `x - 1`.

`DEL.B delay_time bitmask: ...`
Trigger the command up to 16 times at intervals of `delay_time` ms. Active intervals set in 16-bit `bitmask`, LSB = immediate.

Stack

`S: ...`
Place a command onto the stack

`S.CLR`
Clear all entries in the stack

`S.ALL`
Execute all entries in the stack

`S.POP`
Execute the most recent entry

`S.L`
Get the length of the stack

Patterns

`P.N / P.N x`
get/set the pattern number for the working pattern, default `0`

`P x / P x y`
get/set the value of the working pattern at index x

`PN x y / PN x y z`
get/set the value of pattern x at index y

`P.L / P.L x`
get/set pattern length of the working pattern, non-destructive to data

`PN.L x / PN.L x y`
get/set pattern length of pattern x. non-destructive to data

`P.WRAP / P.WRAP x`
when the working pattern reaches its bounds does it wrap (`0/1`), default 1 (enabled)

`PN.WRAP x / PN.WRAP x y`
when pattern x reaches its bounds does it wrap (`0/1`), default 1 (enabled)

`P.START / P.START x`
get/set the start location of the working pattern, default `0`

`PN.START x / PN.START x y`
get/set the start location of pattern x, default `0`

`P.END / P.END x`
get/set the end location of the working pattern, default 63

`PN.END x / PN.END x y`
get/set the end location of the pattern x, default 63

`P.I / P.I x`
get/set index position for the working pattern.

`PN.I x / PN.I x y`
get/set index position for pattern x

`P.HERE / P.HERE x`
get/set value at current index of working pattern

`PN.HERE x / PN.HERE x y`
get/set value at current index of pattern x

`P.NEXT / P.NEXT x`
increment index of working pattern then get/set value

`PN.NEXT x / PN.NEXT x y`
increment index of pattern x then get/set value

`P.PREV / P.PREV x`
decrement index of working pattern then get/set value

`PN.PREV x / PN.PREV x y`
decrement index of pattern x then get/set value

`P.INS x y`
insert value y at index x of working pattern, shift later values down, destructive to loop length

`PN.INS x y z`
insert value z at index y of pattern x, shift later values down, destructive to loop length

`P.RM x`
delete index x of working pattern, shift later values up, destructive to loop length

`PN.RM x y`
delete index y of pattern x, shift later values up, destructive to loop length

`P.PUSH x`
insert value x to the end of the working pattern (like a stack), destructive to loop length

`PN.PUSH x y`
insert value y to the end of pattern x (like a stack), destructive to loop length

`P.POP`
return and remove the value from the end of the working pattern (like a stack), destructive to loop length

`PN.POP x`
return and remove the value from the end of pattern x (like a stack), destructive to loop length

`P.MIN`
find the first minimum value in the pattern between the START and END for the working pattern and return its index

`PN.MIN x`
find the first minimum value in the pattern between the START and END for pattern x and return its index

`P.MAX`
find the first maximum value in the pattern between the START and END for the working pattern and return its index

`PN.MAX x`
find the first maximum value in the pattern between the START and END for pattern x and return its index

`P.SHUF`
shuffle the values in active pattern (between its START and END)

`PN.SHUF x`
shuffle the values in pattern x (between its START and END)

`P.ROT n`
rotate the values in the active pattern n steps (between its START and END, negative rotates backward)

`PN.ROT x n`
rotate the values in pattern x (between its START and END, negative rotates backward)

`P.REV`
reverse the values in the active pattern (between its START and END)

`PN.REV x`
reverse the values in pattern x

`P.RND`
return a value randomly selected between the start and the end position

`PN.RND x`
return a value randomly selected between the start and the end position of pattern x

`P.+ x y`
increase the value of the working pattern at index x by y

`PN.+ x y z`
increase the value of pattern x at index y by z

`P.- x y`
decrease the value of the working pattern at index x by y

`PN.- x y z`
decrease the value of pattern x at index y by z

`P.+W x y a b`
increase the value of the working pattern at index x by y and wrap it to a..b range

`PN.+W x y z a b`
increase the value of pattern x at index y by z and wrap it to a..b range

`P.-W x y a b`
decrease the value of the working pattern at index x by y and wrap it to a..b range

`PN.-W x y z a b`
decrease the value of pattern x at index y by z and wrap it to a..b range

`P.MAP: ...`
apply the 'function' to each value in the active pattern, I takes each pattern value

`PN.MAP x: ...`
apply the 'function' to each value in pattern x, `I` takes each pattern value

Queue

`Q / Q x`
Modify the queue entries

`Q.N / Q.N x`
The queue length

`Q.AVG / Q.AVG x`
Return the average of the queue

`Q.CLR / Q.CLR x`
Clear queue

`Q.GRW / Q.GRW x`
Get/set grow state

`Q.SUM / Q.SUM x`
Get sum of elements

`Q.MIN / Q.MIN x`
Get/set minimum value

`Q.MAX / Q.MAX x`
Get/set maximum value

`Q.RND / Q.RND x`
Get random element/randomize elements

`Q.SRT / Q.SRT`
Sort all or part of queue

`Q.REV`
Reverse queue

`Q.SH / Q.SH x`
Shift elements in queue

`Q.ADD x / Q.ADD x i`
Perform addition on elements in queue

`Q.SUB x / Q.SUB x i`
Perform subtraction on elements in queue

`Q.MUL x / Q.MUL x i`
Perform multiplication on elements in queue

`Q.DIV x / Q.DIV x i`
Perform division on elements in queue

`Q.MOD x / Q.MOD x i`
Perform module (%) on elements in queue

`Q.I i / Q.I i x`
Get/set value of elements at index

`Q.2P / Q.2P i`
Copy queue to current pattern/copy queue to pattern at index `i`

`Q.P2 / Q.P2 i`
Copy current pattern to queue/copy pattern at index `i` to queue

Turtle

`@ / @ x`
get or set the current pattern value under the turtle

`@X / @X x`
get the turtle X coordinate, or set it to x

`@Y / @Y x`
get the turtle Y coordinate, or set it to x

`@MOVE x y`
move the turtle x cells in the X axis and y cells in the Y axis

`@F x1 y1 x2 y2`
set the turtle's fence to corners x1,y1 and x2,y2

`@FX1 / @FX1 x`
get the left fence line or set it to x

`@FX2 / @FX2 x`
get the right fence line or set it to x

`@FY1 / @FY1 x`
get the top fence line or set it to x

`@FY2 / @FY2 x`
get the bottom fence line or set it to x

`@SPEED / @SPEED x`
get the speed of the turtle's @STEP in cells per step or set it to x

`@DIR / @DIR x`
get the direction of the turtle's @STEP in degrees or set it to x

`@STEP`
move @SPEED/100 cells forward in @DIR, triggering @SCRIPT on cell change

`@BUMP / @BUMP 1`
get whether the turtle fence mode is BUMP, or set it to BUMP with 1

`@WRAP / @WRAP 1`
get whether the turtle fence mode is WRAP, or set it to WRAP with 1

`@BOUNCE / @BOUNCE 1`
get whether the turtle fence mode is BOUNCE, or set it to BOUNCE with 1

`@SCRIPT / @SCRIPT x`
get which script runs when the turtle changes cells, or set it to x

`@SHOW / @SHOW 0/1`
get whether the turtle is displayed on the TRACKER screen, or turn it on or off

Grid

G.RST
full grid reset

G.CLR
clear all LEDs

G.DIM level
set dim level

G.ROTATE x
set grid rotation

G.KEY x y action
emulate grid press

G.GRP / G.GRP id
get/set current group

G.GRP.EN id / G.GRP.EN id x
enable/disable group or check if enabled

G.GRP.RST id
reset all group controls

G.GRP.SW id
switch groups

G.GRP.SC id / G.GRP.SC id script
get/set group script

G.GRPI
get last group

G.LED x y / G.LED x y level
get/set LED

G.LED.C x y
clear LED

G.REC x y w h fill border
draw rectangle

G.RCT x1 y1 x2 y2 fill border
draw rectangle

G.BTN id x y w h type level script
initialize button

G.GBT group id x y w h type level script
initialize button in group

G.BTX id x y w h type level script columns rows
initialize multiple buttons

G.GBX group id x y w h type level script columns rows
initialize multiple buttons in group

G.BTN.EN id / G.BTN.EN id x
enable/disable button or check if enabled

G.BTN.X id / G.BTN.X id x
get/set button x coordinate

G.BTN.Y id / G.BTN.Y id y
get/set button y coordinate

G.BTN.V id / G.BTN.V id value
get/set button value

G.BTN.L id / G.BTN.L id level
get/set button level

G.BTNI
id of last pressed button

G.BTNX / G.BTNX x
get/set x of last pressed button

G.BTNY / G.BTNY y
get/set y of last pressed button

G.BTNV / G.BTNV value
get/set value of last pressed button

G.BTNL / G.BTNL level
get/set level of last pressed button

G.BTN.SW id
switch button

G.BTN.PR id action
emulate button press/release

G.GBTN.V group value
set value for group buttons

G.GBTN.L group odd_level even_level
set level for group buttons

G.GBTN.C group
get count of currently pressed

G.GBTN.I group index
get id of pressed button

G.GBTN.W group
get button block width

G.GBTN.H group
get button block height

G.GBTN.X1 group
get leftmost pressed x

G.GBTN.X2 group
get rightmost pressed x

G.GBTN.Y1 group
get highest pressed y

G.GBTN.Y2 group
get lowest pressed y

G.FDR id x y w h type level script
initialize fader

G.GFD grp id x y w h type level script
initialize fader in group

G.FDX id x y w h type level script columns rows
initialize multiple faders

G.GFX group id x y w h type level script columns rows
initialize multiple faders in group

G.FDR.EN id / G.FDR.EN id x
enable/disable fader or check if enabled

G.FDR.X id / G.FDR.X id x
get/set fader x coordinate

G.FDR.Y id / G.FDR.Y id y
get/set fader y coordinate

G.FDR.N id / G.FDR.N id value
get/set fader value

G.FDR.V id / G.FDR.V id value
get/set scaled fader value

G.FDR.L id / G.FDR.L id level
get/set fader level

G.FDRI
id of last pressed fader

G.FDRX / G.FDRX x
get/set x of last pressed fader

G.FDRY / G.FDRY y
get/set y of last pressed fader

G.FDRN / G.FDRN value
get/set value of last pressed fader

G.FDRV / G.FDRV value
get/set scaled value of last pressed fader

G.FDRL / G.FDRL level
get/set level of last pressed fader

G.FDR.PR id value
emulate fader press

G.GFDR.N group value
set value for group faders

G.GFDR.V group value
set scaled value for group faders

G.GFDR.L group odd_level even_level
set level for group faders

G.GFDR.RN group min max
set range for group faders

MIDI In

MI.$ x / MI.$ x y
assign MIDI event type x to script y

MI.LE
get the latest event type

MI.LCH
get the latest channel (1..16)

MI.LN
get the latest Note On (0..127)

MI.LNV
get the latest Note On scaled to teletype range (shortcut for `N MI.LN`)

MI.LV
get the latest velocity (0..127)

MI.LVV
get the latest velocity scaled to 0..16383 range (0..+10V)

MI.LO
get the latest Note Off (0..127)

MI.LC
get the latest controller number (0..127)

MI.LCC
get the latest controller value (0..127)

MI.LCCV
get the latest controller value scaled to 0..16383 range (0..+10V)

MI.NL
get the number of Note On events

MI.NCH
get the Note On event channel (1..16) at index specified by variable `I`

MI.N
get the Note On (0..127) at index specified by variable `I`

MI.NV
get the Note On scaled to 0..+10V range at index specified by variable `I`

MI.V
get the velocity (0..127) at index specified by variable `I`

MI.VV
get the velocity scaled to 0..10V range at index specified by variable `I`

MI.OL
get the number of Note Off events

MI.OCH
get the Note Off event channel (1..16) at index specified by variable `I`

MI.O
get the Note Off (0..127) at index specified by variable `I`

MI.CL
get the number of controller events

MI.CCH
get the controller event channel (1..16) at index specified by variable `I`

MI.C
get the controller number (0..127) at index specified by variable `I`

MI.CC
get the controller value (0..127) at index specified by variable `I`

MI.CCV
get the controller value scaled to 0..+10V range at index specified by variable `I`

MI.CLKD / MI.CLKD x
set clock divider to x (1-24) or get the current divider

MI.CLKR
reset clock counter

Calibration

DEVICE.FLIP
Flip the screen/inputs/outputs

IN.CAL.MIN
Reads the input CV and assigns the voltage to the zero point

IN.CAL.MAX
Reads the input CV and assigns the voltage to the max point

IN.CAL.RESET
Resets the input CV calibration

PARAM.CAL.MIN
Reads the Parameter Knob minimum position and assigns a zero value

PARAM.CAL.MAX
Reads the Parameter Knob maximum position and assigns the maximum point

PARAM.CAL.RESET
Resets the Parameter Knob calibration

CV.CAL n mv1v mv3v
Calibrate CV output n

CV.CAL.RESET n
Reset calibration data for CV output n