# Logical Switch Tutorial 1
# Introduction

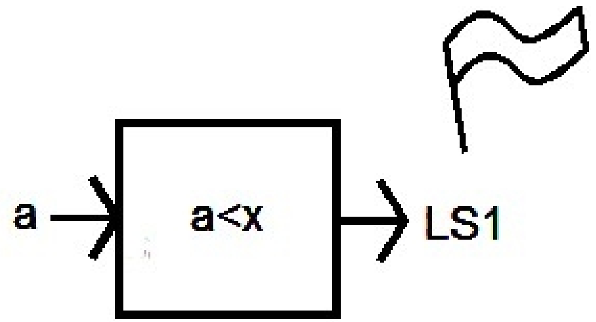
Manually activated Switches are labeled SA to SH: SF is a two position switch, SH is a two position momentary switch and the others are three position switches.  Each of these switches provide signals to the computer. The computer checks each switch position regularly (30 times per second).

Example:

Switch SA can be in one of three positions.  The computer reads +100, 0, and -100. Each position tells the computer something different.  You can use the switch values as input to functions or you can use the values just as “flags” to tell what position the switch is in. The computer decides what to do depending on what you have selected that switch position to do when you set up your model.   Maybe you set SA↑ to select Flight Mode 2. When the computer sees that you have positioned SA to SA↑ it will set Flight Mode 2.

# Logical Switch Tutorial 1 - p2

Logical Switches are like manual switches except that they can be activated automatically, in response to some condition sensed by the OpenTX program.  The computer can be set up to watch for some condition to happen, then “raise a flag” to signal that the condition has been seen.   The computer checks the state of the Logical Switch “flags” along with the manual switch states as it prepares to assemble the next set of commands to send to the receiver in your model.

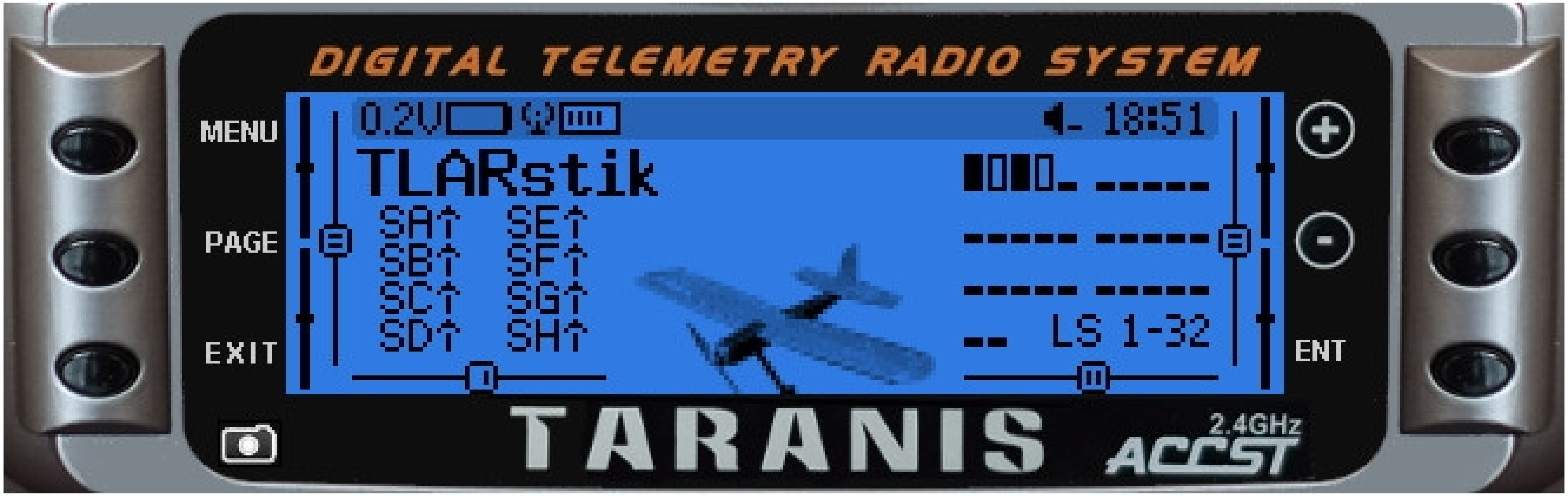



- a = Value being observed
- a<x tells what the “Black Box” function looks for
- LS1 is the output of the Black box if a<x is found

If (a = Throttle value) and (x=-95),
then when the throttle moves to output a value less than -95, LS1 will signal (flag)
Otherwise, LS1 will not signal.

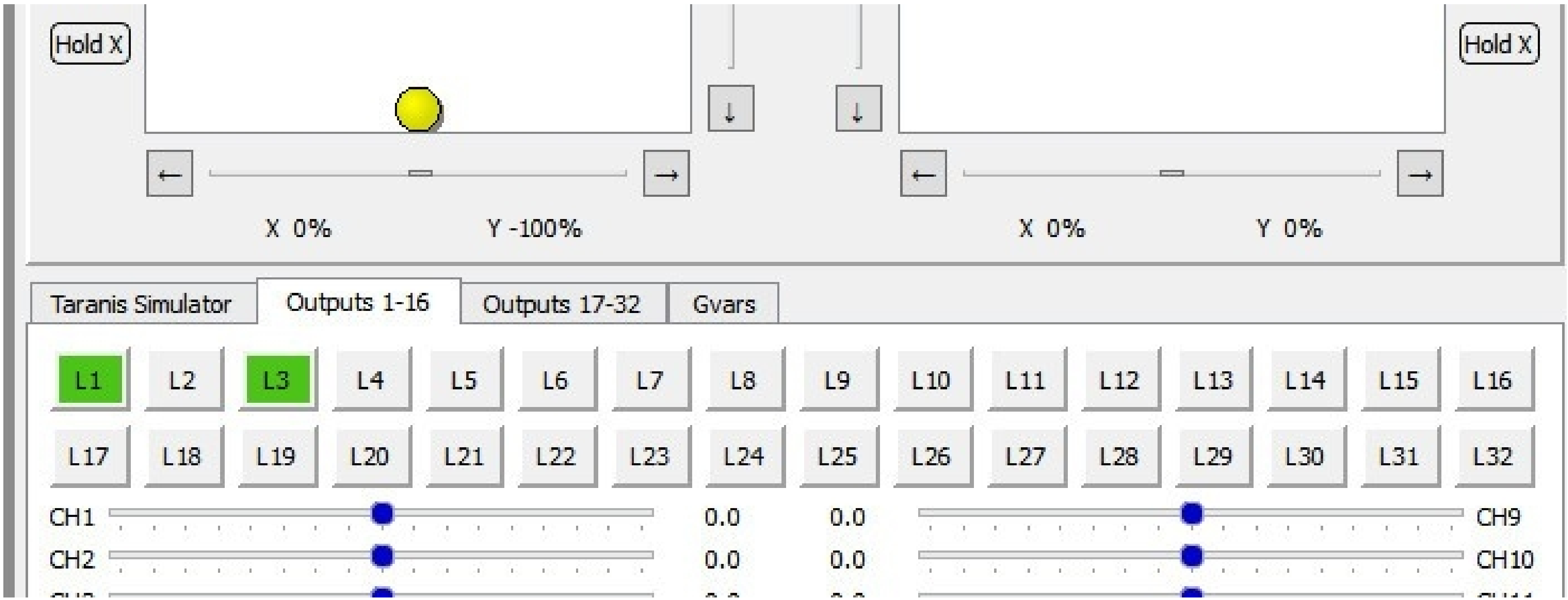
A Logic Switch can be used to sense many conditions, and the resulting signal can command any of a large number of changes to the model setup

# Logical Switch Tutorial 1 – p3
# Taranis Logical Switch Screen



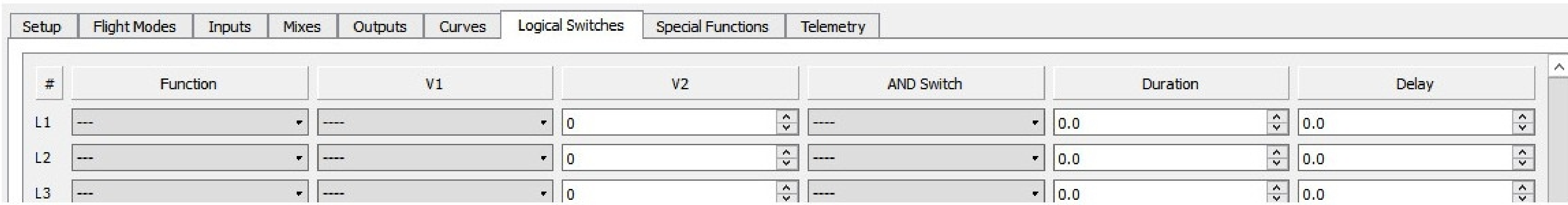

The Logical Switch screen shows the state of the 32 Logical Switches (LS)
The square boxes show which LS has been programmed
Black square boxes show which LS has been set TRUE and so is active.

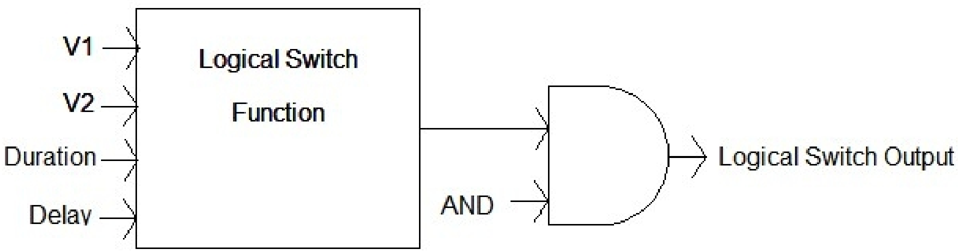
# Logical Switch Tutorial 1 – p4
# Companion Logical Switch Screen




The Logical Switch screen shows the state of the 32 Logical Switches (LS)
The square boxes show all LS states, whether programmed or not
Green square boxes show which LS has been set TRUE and therefore are active

# Logical Switch Tutorial 1 - p5




| | |
|---|---|
| Function | Select Logical Switch (LS) Function |
| V1 | Input Signal Source 1 |
| V2 | Input Signal Source 2 |
| And Switch | Input Signal to the AND Switch |
| Duration | Time the LS Function is TRUE |
| Delay | Time before the LS Function is TRUE after it is signaled |

# Logical Switch Tutorial 1 – p6
## Switch Functions

a = x  When Input V1 is equal to the Value of V2
a ~ x  When Input V1 is almost equal (+/- 0.9) to the Value of V2
a < x  When Input V1 is less than the Value of V2
a > x  When Input V1 is greater than the Value of V2

These four functions look at input value V1 and compare it to a specific value of V2 which you define.  The input V1 is selected from a list of a drop-down menu.  The dropdown list includes discrete and analogue sources of input.

Even comparing a ~ x may be difficult because the value must be about +/- 0.9 from the set value, so approximate is still very close.

# Logical Switch Tutorial 1 – p7
# Switch Functions

a = b  When Input V1 is equal to input V2

a ~ b  When Input V1 is almost equal (+/- 0.9) to input V2

a < b  When Input V1 is less than input V2

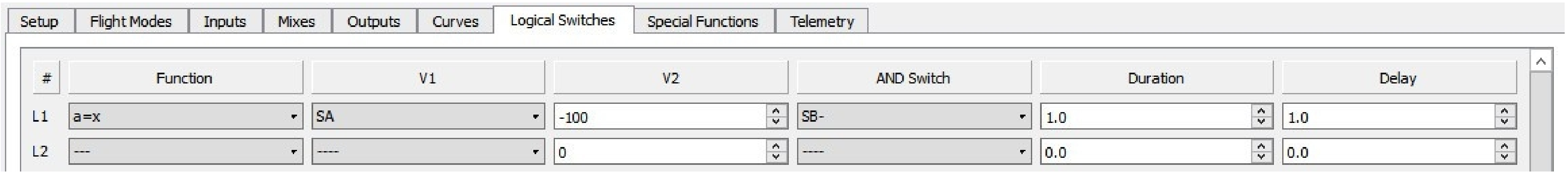
a > b  When Input V1 is greater than input V2

These functions compare the value of an input V1 against the value of a second input V2.  Each input is selected from a list of a drop-down menu.

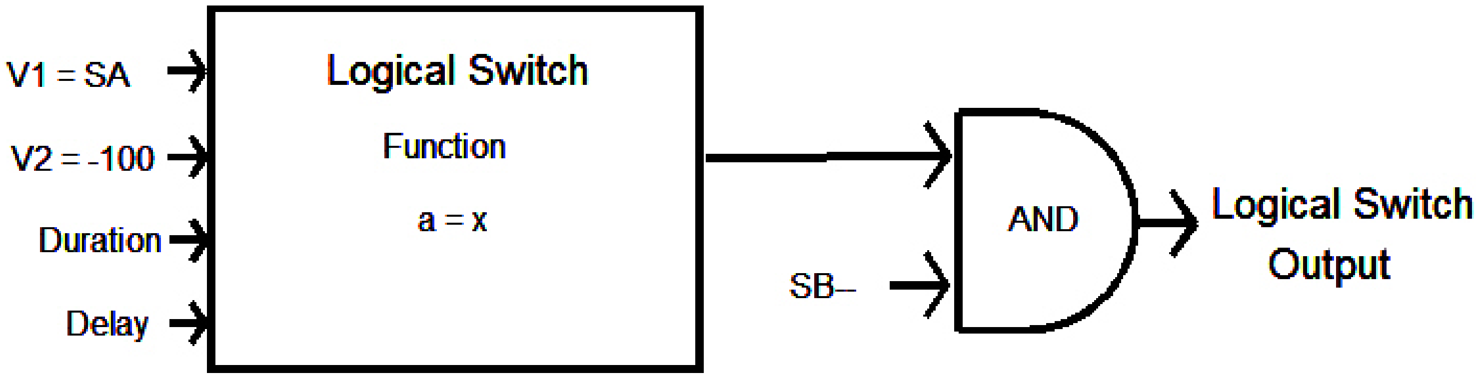
Comparing a = b and  a ~ b may be very difficult when you are comparing two analogue signals such as Left Slider and Potentiometer S1 because it may be difficult to set each one exactly on the same value.

Even comparing a ~ b may be difficult because the value must be about +/- 0.9 from the set value, so approximate is still very close.

# Logical Switch Tutorial 1 - p8


Setup | Flight Modes | Inputs | Mixes | Outputs | Curves | Logical Switches | Special Functions | Telemetry

| # | Function | V1 | V2 | AND Switch | Duration | Delay |
|---|---|---|---|---|---|---|
| L1 | a=x | SA | -100 | SB- | 1.0 | 1.0 |
| L2 | --- | ---- | 0 | ---- | 0.0 | 0.0 |

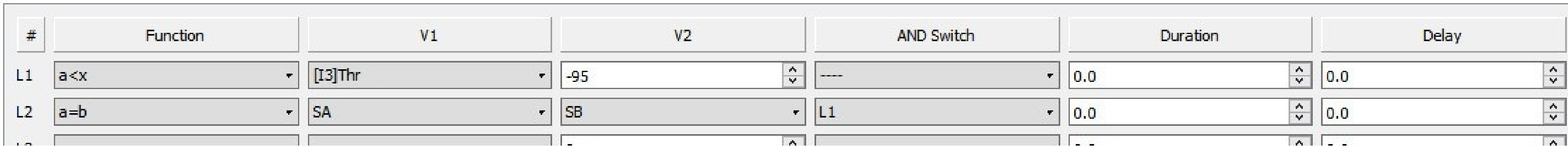



| | |
|---|---|
| Function | a = x |
| V1 | Manual Switch SA |
| V2 | Value = -100 |
| And Switch | Manual Switch SB-- |
| Duration | 1.0 sec |
| Delay | 1.0 sec |

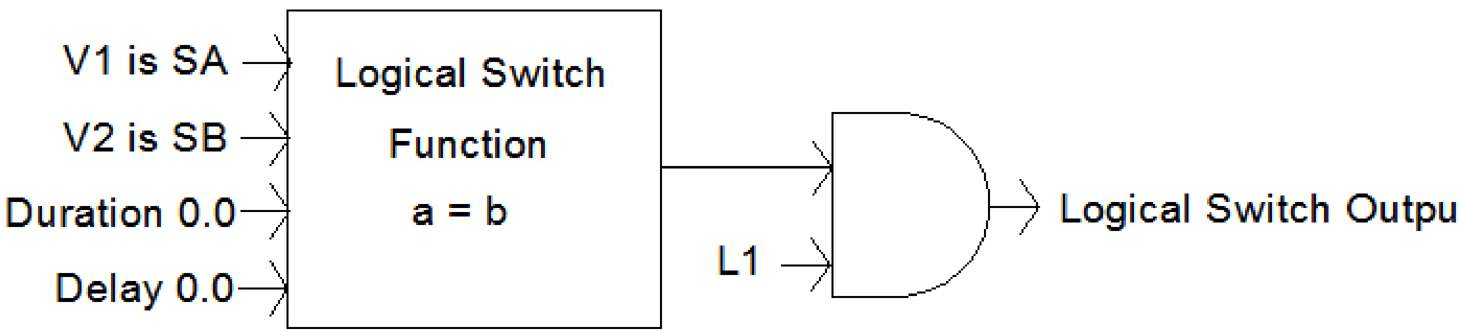
V1 is compared to v2 (-100)
If Switch A reads -100
Then the input to the AND Gate will be TRUE for 1 second after a delay of 1 sec

AND if Switch SB is set in the Middle (SB--) the LS output will be TRUE

# Logical Switch Tutorial 1 - p9



| # | Function | V1 | V2 | AND Switch | Duration | Delay |
|---|---|---|---|---|---|---|
| L1 | a<x | [I3]Thr | -95 | ---- | 0.0 | 0.0 |
| L2 | a=b | SA | SB | L1 | 0.0 | 0.0 |




| | |
|---|---|
| Function | a = b |
| V1 | Manual Switch SA |
| V2 | Manual Switch SB |
| And Switch | Logical Switch L1 |
| Duration | Minimum Duration |
| Delay | No delay |

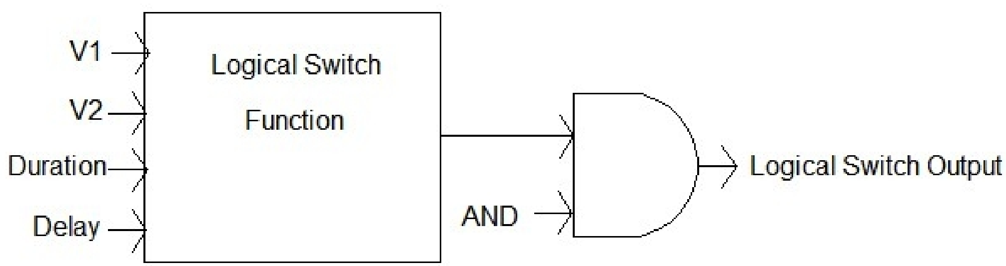
L1 will be true when the Throttle
Is moved to lower than -95

L2 will be TRUE when both SA and SB are set the same AND L1 is TRUE

So L2 cannot be true if the Throttle is active

# Logical Switch Tutorial 1 – p10
# Exercise 1

| # | Function | V1 | V2 | AND Switch | Duration | Delay |
|---|---|---|---|---|---|---|
| L1 | a=x | SA | 100 | ---- | 0.0 | 0.0 |
| L2 | a=x | SA | 100 | ---- | 2.0 | 1.0 |
| L3 | a=x | SA | 100 | ---- | 0.0 | 1.0 |
| L4 | a=x | SA | 100 | L3 | 0.0 | 1.0 |
| L5 | --- | ---- | 0 | ---- | 0.0 | 0.0 |



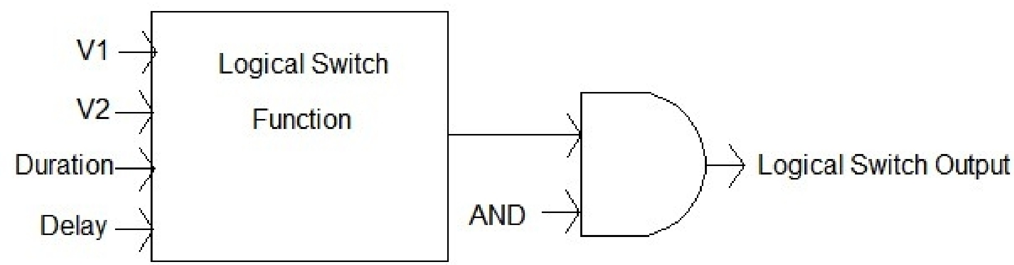

As an exercise, set up logical switches as above.

Activate Manual Switch SA.

Observe the results on the Taranis Display and the Companion simulator

# Logical Switch Tutorial 1 – p11
# Exercise 2 – 5 Position Rotary Switch

| # | Function | V1 | V2 | AND Switch | Duration | Delay |
|---|---|---|---|---|---|---|
| L1 | a<x | S1 | -85 | ---- | 0.0 | 0.0 |
| L2 | a<x | S1 | -45 | !L1 | 0.0 | 0.0 |
| L3 | \|a\|<x | S1 | 45 | ---- | 0.0 | 0.0 |
| L4 | a>x | S1 | 45 | !L5 | 0.0 | 0.0 |
| L5 | a>x | S1 | 85 | ---- | 0.0 | 0.0 |
| L6 | --- | ---- | 0 | ---- | 0.0 | 0.0 |




As an exercise, set up logical switches as above.

Turn Rotary Switch S1

Observe the results on the Taranis Display and the Companion simulator

See 5 individual flags as you rotate S1. Could you use this functionality?

Note:
!L1 means NOT L1 or the opposite of L1.
a<x means a less than x
a>x means a greater than x
|a| means V2 is evaluated as positive whether V2 is positive or negative

# Logical Switch Tutorial 1 – p12
# Exercise 3 – 18 Logical Switches from 3 manual switches

| # | Function | V1 | V2 | AND Switch | |
|---|---|---|---|---|---|
| L1 | AND | SA↑ | SB↑ | SC↑ | 0.0 |
| L2 | AND | SA↑ | SB- | SC↑ | 0.0 |
| L3 | AND | SA↑ | SB↓ | SC↑ | 0.0 |
| L4 | AND | SA- | SB↑ | SC↑ | 0.0 |
| L5 | AND | SA- | SB- | SC↑ | 0.0 |
| L6 | AND | SA- | SB↓ | SC↑ | 0.0 |
| L7 | AND | SA↓ | SB↑ | SC↑ | 0.0 |
| L8 | AND | SA↓ | SB- | SC↑ | 0.0 |
| L9 | AND | SA↓ | SB↓ | SC↑ | 0.0 |
| L10 | AND | SA↑ | SB↑ | SC↓ | 0.0 |
| L11 | AND | SA↑ | SB- | SC↓ | 0.0 |
| L12 | AND | SA↑ | SB↓ | SC↓ | 0.0 |
| L13 | AND | SA- | SB↑ | SC↓ | 0.0 |
| L14 | AND | SA- | SB- | SC↓ | 0.0 |
| L15 | AND | SA- | SB↓ | SC↓ | 0.0 |
| L16 | AND | SA↓ | SB↑ | SC↓ | 0.0 |
| L17 | AND | SA↓ | SB- | SC↓ | 0.0 |
| L18 | AND | SA↓ | SB↓ | SC↓ | 0.0 |

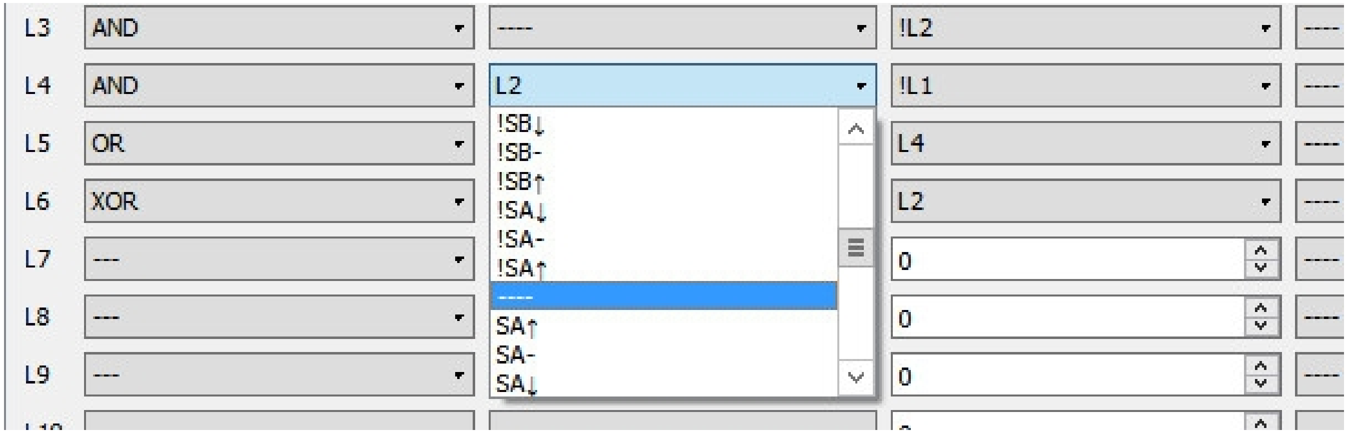
# Logical Switch Tutorial 1 – p13
## Exercise 3 – continued

Two three position switches (SA and SB) are used to select nine unique states.

The third three position switch (SC) is used to activate the Logical Switches formed by the AND of the SA and SB switches.  This will activate 18 Logical Switches

The middle position (SC--) of switch SC provides a neutral state so you can select the Logical Switch you want before you activate it with SC.

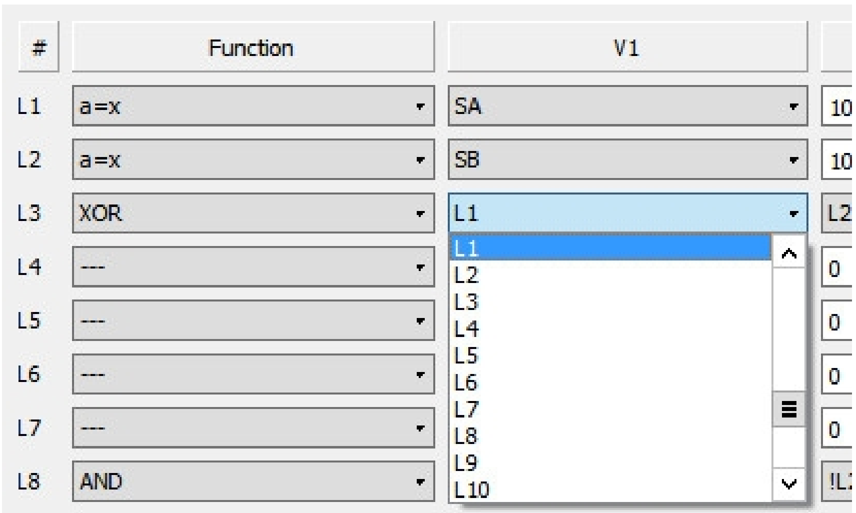
# Logical Switch Tutorial 2 – p1
# Logic Gates – AND, OR, XOR
# Initial Considerations for manual switches

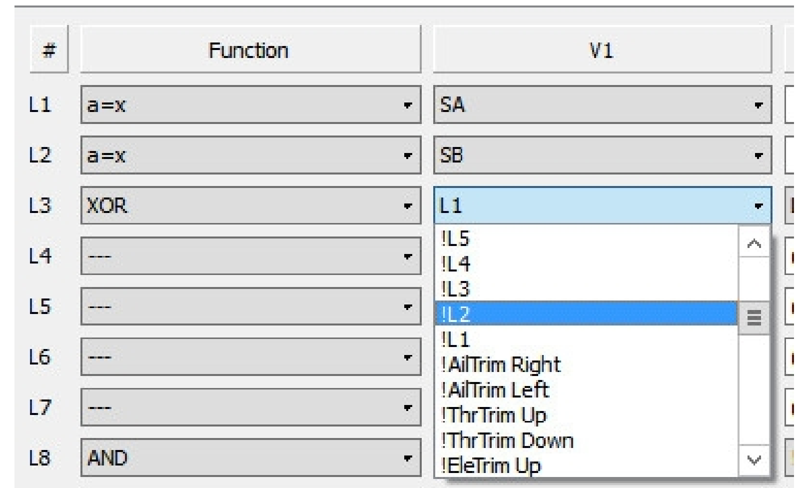



Gate Inputs:

- AND, OR and XOR Gate inputs need to be either TRUE or FALSE. They cannot be analogue (anything from 100 to -100)
- Each manual switch has two or three positions. Each position can be either TRUE (eg. SA↑ can be TRUE or False) or FALSE (!SA↑ can be TRUE or FALSE)
- You can select SA↑ or !SA↑ as an input to a Logic Switch but you must be very aware of what value of SA↑ you are expecting.

# Logical Switch Tutorial 2 – p2
# Logic Gates – AND, OR, XOR
# Initial Considerations for Logical Switches



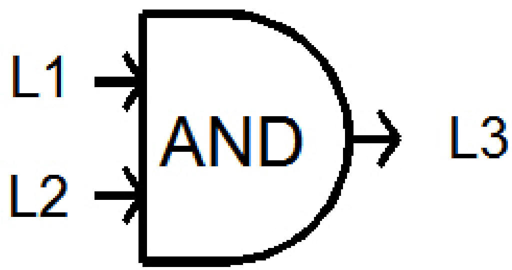

Gate Inputs:

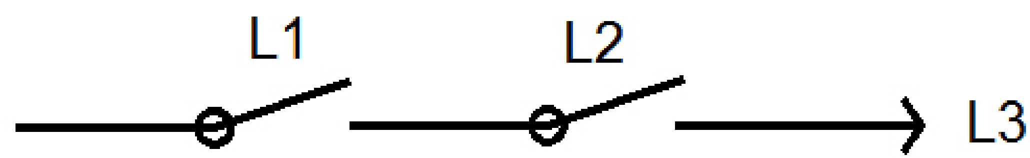
- AND, OR and XOR Gate inputs need to be either TRUE or FALSE. They cannot be analogue (anything from 100 to -100)
- Outputs from Logical Switches can have only two states, either TRUE or FALSE.
- BUT: You have a choice. You can use the TRUE output (L1) or its opposite (!L1) output as the input to another Logical Switch. When L1 is TRUE !L1 will be FALSE.

- The following descriptions will show generic Truth Tables indicating True (T) and FALSE (F). A closed switch will indicate TRUE and an open switch will indicate FALSE.

# Logical Switch Tutorial 2 – p3
# Logic of AND

| # | Function | V1 | V2 | |
|---|---|---|---|---|
| L1 | a=x | SA | 0 | ---- |
| L2 | a=x | SB | -100 | ---- |
| L3 | AND | L1 | L2 | ---- |



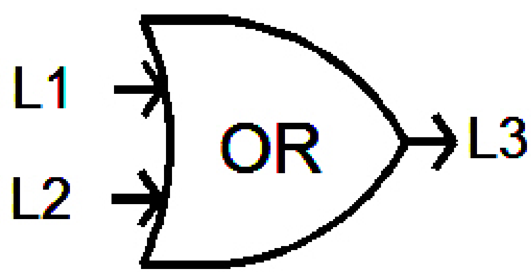

AND

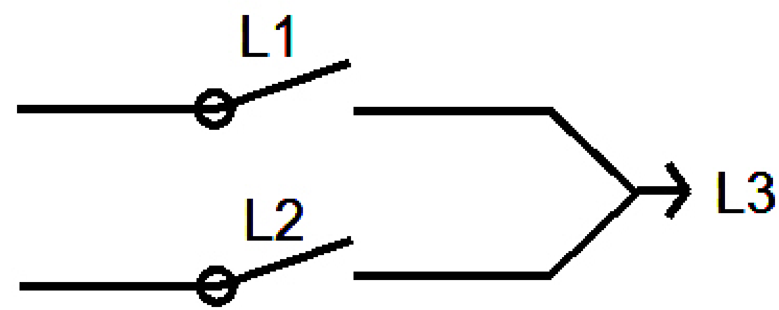
| L1 | L2 | L3 |
|---|---|---|
| F | F | F |
| F | T | F |
| T | F | F |
| T | T | T |

Both inputs must be TRUE for AND to signal the AND Logical Switch Output TRUE

# Logical Switch Tutorial 2 – p4
# Logic of OR

| # | Function | V1 | V2 | AND |
|---|---|---|---|---|
| L1 | a=x | SA | 0 | ---- |
| L2 | a=x | SB | -100 | ---- |
| L3 | OR | L1 | L2 | ---- |



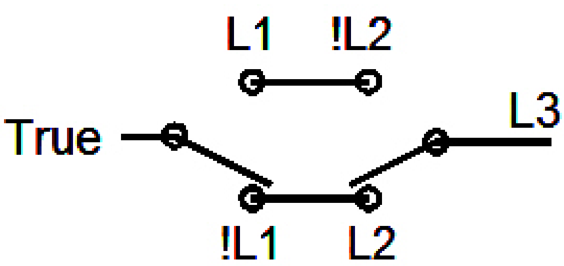

OR

| L1 | L2 | L3 |
|---|---|---|
| F | F | F |
| F | T | T |
| T | F | T |
| T | T | T |

At least one input must be TRUE for OR
to signal the OR Logical Switch Output TRUE

# Logical Switch Tutorial 2 – p5
# Logic of XOR

| # | Function | V1 | V2 | |
|---|---|---|---|---|
| L1 | a=x | SA | 100 | ---- |
| L2 | a=x | SB | 100 | ---- |
| L3 | XOR | L1 | L2 | ---- |



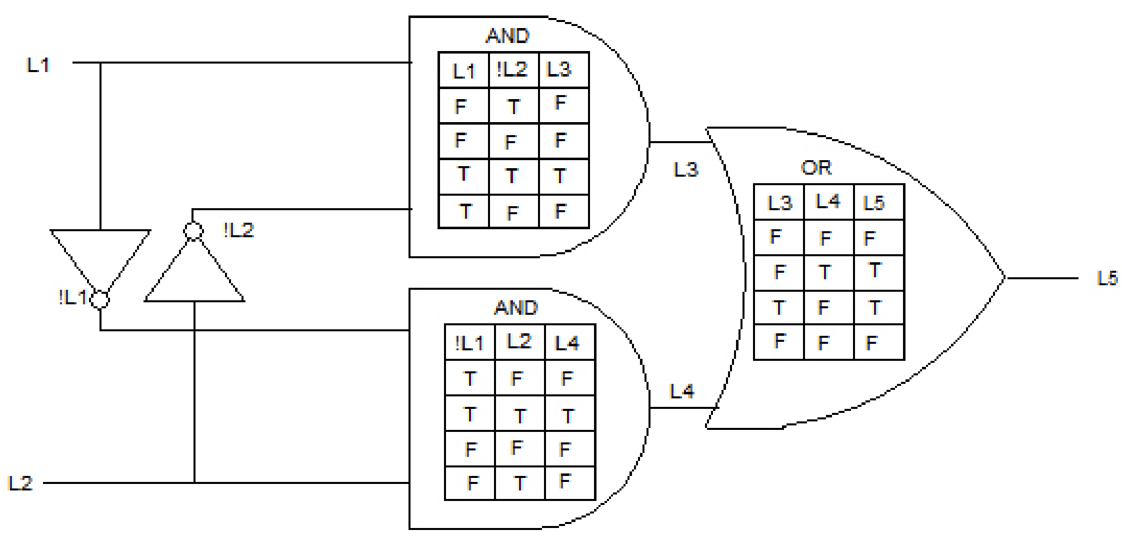

XOR

| L1 | L2 | L3 |
|---|---|---|
| F | F | F |
| F | T | T |
| T | F | T |
| T | T | F |

- The Output of L1 will be TRUE if the Switch A Output Is 100.
- The Output of L2 will be TRUE if the Switch B Output Is 100.
- Either V1 (The output of L1) or V2 (The output of L2) must be TRUE for the output of L3 to be TRUE.
- If the output of L1 and the output of L2 are both TRUE the output of L3 will be FALSE.
- If the output of L1 and the output of L2 are both FALSE the output of L3 will be FALSE.

# Logical Switch Tutorial 2 – p6
# Logic of XOR for the enthusiast

| # | Function | V1 | V2 | |
|---|---|---|---|---|
| L1 | a=x | SA | 100 | ---- |
| L2 | a=x | SB | 100 | ---- |
| L3 | AND | L1 | !L2 | ---- |
| L4 | AND | !L1 | L2 | ---- |
| L5 | OR | L3 | L4 | ---- |
| L6 | XOR | L1 | L2 | ---- |



# Logical Switch Tutorial 2 – p7
# Equivalent AND and OR Functions

This page and the following page is for those who want to investigate the AND and OR Logical Functions some more. You use the appropriate AND or OR function depending on the inputs you put in and the resulting outputs you need.

In Taranis OpenTX you have a choice of using positive (eg. L1) values or negative (eg. !L1) values as inputs and you can choose either positive (eg. L3) or negative (eg. !L3) values as output results. This makes the AND and OR functions very powerful.

See the following tables and compare inputs and outputs to see your options.

# Logical Switch Tutorial 2 – p8
# Equivalent AND and OR Functions

| # | Function | V1 | V2 | AND Switch | |
|---|---|---|---|---|---|
| L1 | a=x | SA | 100 | ---- | 0.0 |
| L2 | a=x | SB | 100 | ---- | 0.0 |
| L3 | AND | L1 | L2 | ---- | 0.0 |
| L4 | AND | !L1 | !L2 | ---- | 0.0 |
| L5 | OR | L1 | L2 | ---- | 0.0 |
| L6 | OR | !L1 | !L2 | ---- | 0.0 |

Table 1 AND

| L1 | L2 | L3 | !L3 |
|---|---|---|---|
| F | F | F | T |
| F | T | F | T |
| T | F | F | T |
| T | T | T | F |

Table 2 AND

| !L1 | !L2 | L4 | !L4 |
|---|---|---|---|
| T | T | T | F |
| T | F | F | T |
| F | T | F | T |
| F | F | F | T |

Table 3 OR

| L1 | L2 | L5 | !L5 |
|---|---|---|---|
| F | F | F | T |
| F | T | T | F |
| T | F | T | F |
| T | T | T | F |

Table 4 OR

| !L1 | !L2 | L6 | !L6 |
|---|---|---|---|
| T | T | T | F |
| T | F | T | F |
| F | T | T | F |
| F | F | F | T |

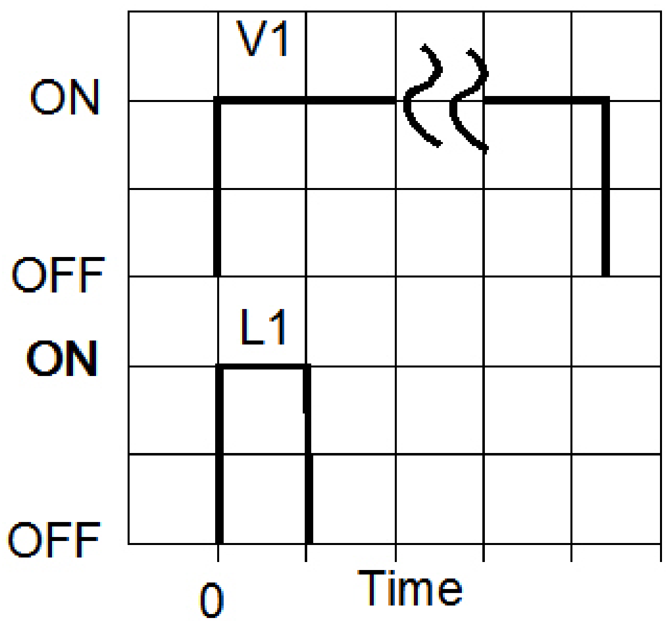
You are able to build logical functions with either positive (eg. L1) values or negative (eg. !L1) values.

See how an AND result can be equivalent to an OR result depending on whether you use negative or positive inputs and use the negative or positive outputs.

# Logical Switch Tutorial 3 - p1
# Edge Function – Trigger with Activating signal Activation

Companion

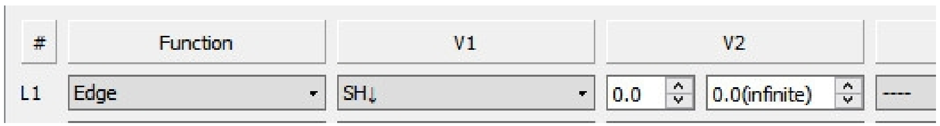

Taranis


L1 Pulse triggered as soon as the Switch SH is active
Example shows switch V1 with 0.0 sec selected and time is Instant.

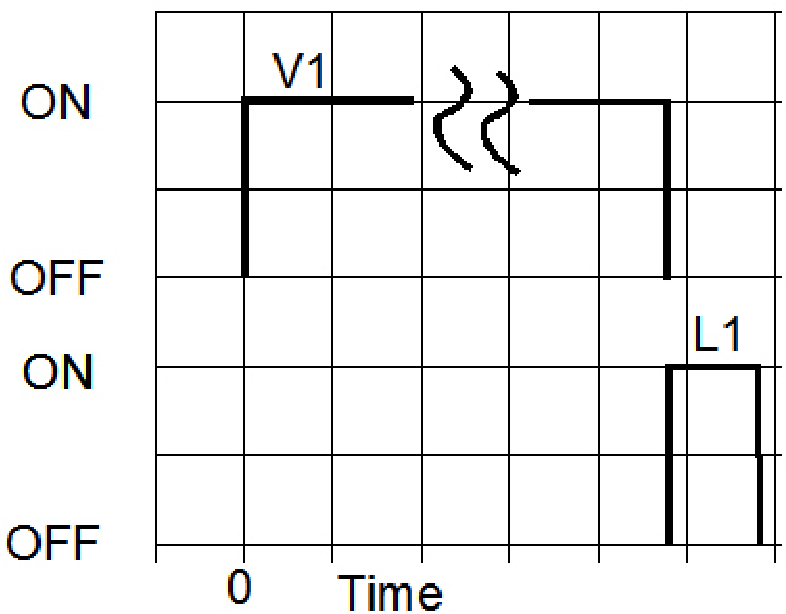
# Logical Switch Tutorial 3 - p2
## Edge Function – Trigger with Activating signal Deactivation

Companion 2.1.7




Taranis

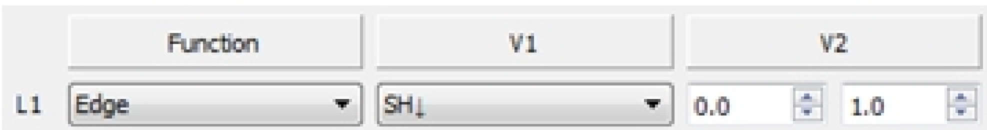

Companion 2.0.19

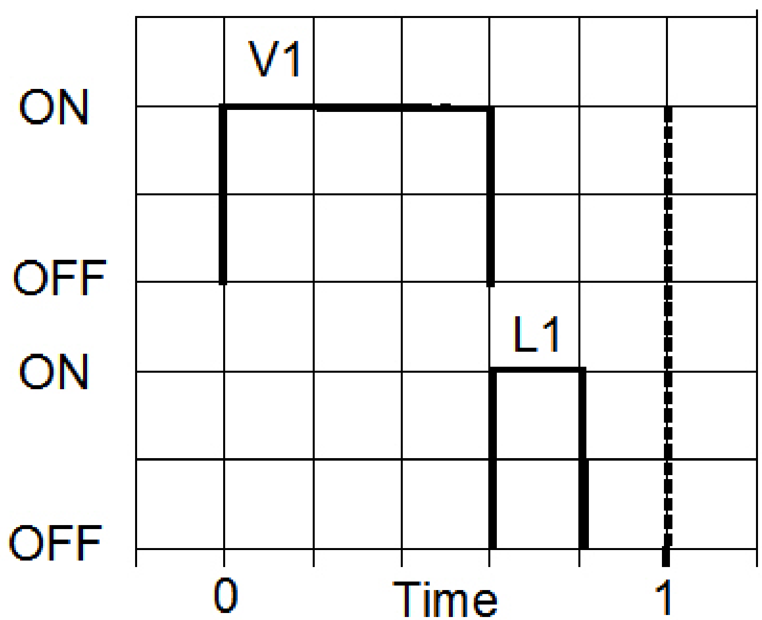

L1 Pulse triggered as soon as the switch SH is Released
Example shows switch V1 with 0.0 sec selected and time is Release.

# Logical Switch Tutorial 3 - p3
# Edge Function – Trigger before the Specified Boundary

Companion



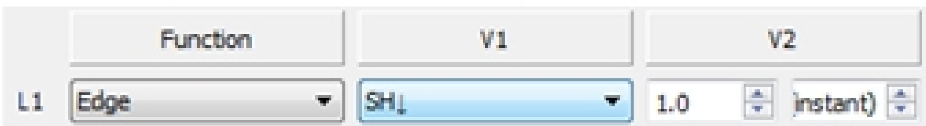

Taranis

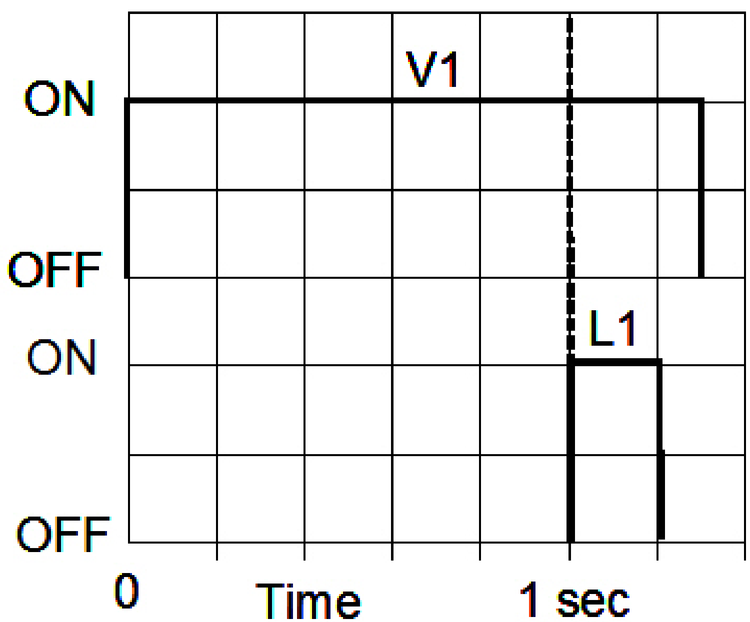




Pulse triggered as soon as Switch SH is Released IF it is released Before 1.0 sec. (The dotted line indicates the time boundary.)
Example shows switch V1 with 0.0 sec selected and Release time is 1.0 sec.

# Logical Switch Tutorial 3 - p4
# Edge Function – Trigger At the Specified Boundary

Companion




Taranis


L1 Pulse triggered as soon as Switch SH is Held until the selected 1.0 sec time is passed. (The dotted line indicates the timeout goal.) SH can be held past the timeout goal but L1 will be triggered at the timeout goal

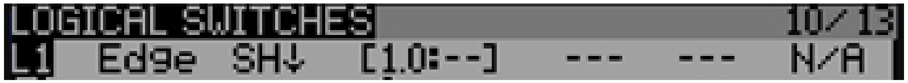
Example shows switch V1 with 1.0 sec selected as the timeout goal and instant as the response when the timeout is reached.

# Logical Switch Tutorial 3 - p5
# Edge Function – Trigger past the Specified Boundary

Companion 2.1.7

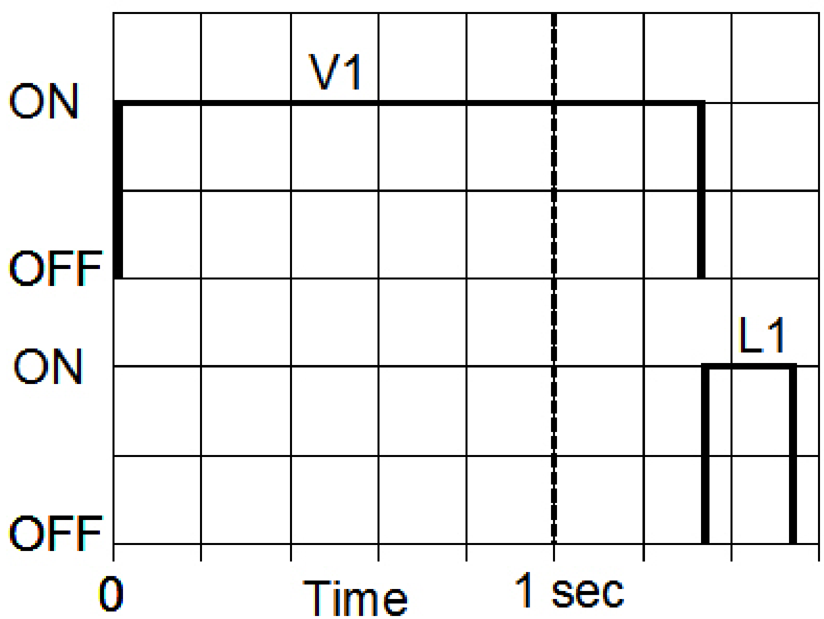
| # | Function | V1 | V2 | | |
|---|---|---|---|---|---|
| L1 | Edge | SH↓ | 1.0 | 1.0(infinite) | ---- |

Taranis




Companion 2.0.19

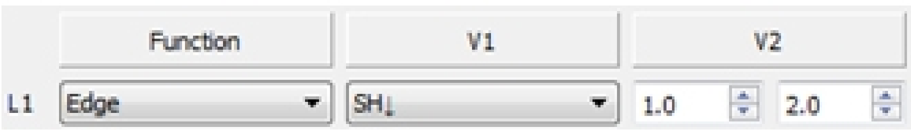
| | Function | V1 | V2 | |
|---|---|---|---|---|
| L1 | Edge | SH↓ | 1.0 | 1.0(release) |



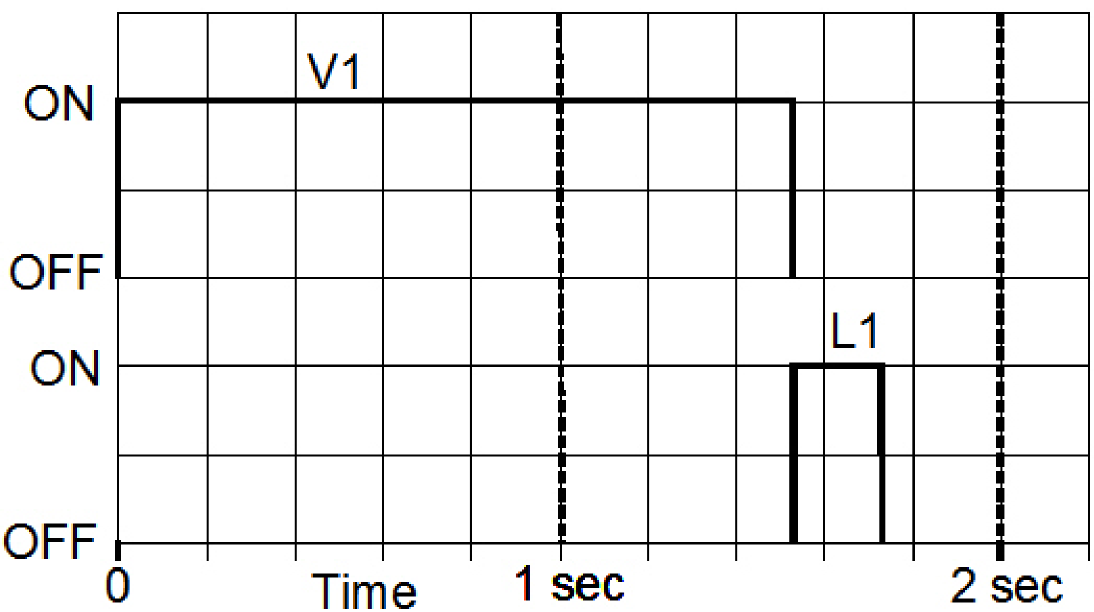

L1 Pulse triggered when Switch SH is Released IF the selected 1.0 sec time is passed. (The dotted line indicates the timeout goal.)
Example shows switch V1 with 1.0 sec selected as the timeout goal and Switch Release as the time when the Logical Switch is activated IF the selected switch is held past the timeout goal.

# Logical Switch Tutorial 3 - p6
# Edge Function – Trigger between two Specified Boundaries

Companion




Taranis


L1 Pulse triggered when the Switch SH is Released After the selected 1.0 sec boundary time is passed AND Before the Second boundary time is reached. (The dotted lines indicates the timeout boundaries.) Example shows switch V1 with 1.0 sec selected as the first boundary, 2.0 sec as the second boundary and Switch Release as the time when the Logical Switch is Released some time between the boundaries. If the Switch is released before the first boundary or after the second boundary the Logical Switch will not be triggered.

# Logical Switch Tutorial 3 - p7
# Edge Function – Simple Example
# Throttle Hold

| # | Function | V1 | V2 | | AND Switch | Durat |
|---|---|---|---|---|---|---|
| L1 | a<x | [I3]Thr | -98 | | ---- | 0.0 |
| L2 | Edge | SH↓ | 0.7 | (instant) | L1 | 0.0 |
| L3 | Sticky | L2 | L2 | | ---- | 0.0 |
| L4 | Edge | SH↓ | 0.0 | 0.6 | L3 | 0.0 |

- L1 is TRUE when the throttle is brought to OFF (<-98)
- L2 uses the Edge function which sets L2 TRUE as soon as SH is held for more than 0.7 sec. (but only when L1 is active). This means activating and deactivating the Throttle Hold must be a deliberate action. A bump should not do it. With no duration set the L2 pulse will last for only one cycle of the program.
- L3 uses the Sticky function to create a Toggle which will hold the selection of Throttle Hold until L2 is again activated to release the Throttle Hold. (L1 must be active when release is commanded also.) A mix is added to Throttle to Replace the main mix.
- L4 uses the Edge function which will activate L4 if the Momentary Switch is Released before the specified time boundary. I use this to trigger a Special Function Voice command telling me that Throttle Hold is engaged.

# Logical Switch Tutorial 3 - p8
# Single Channel Compound Escapement Example

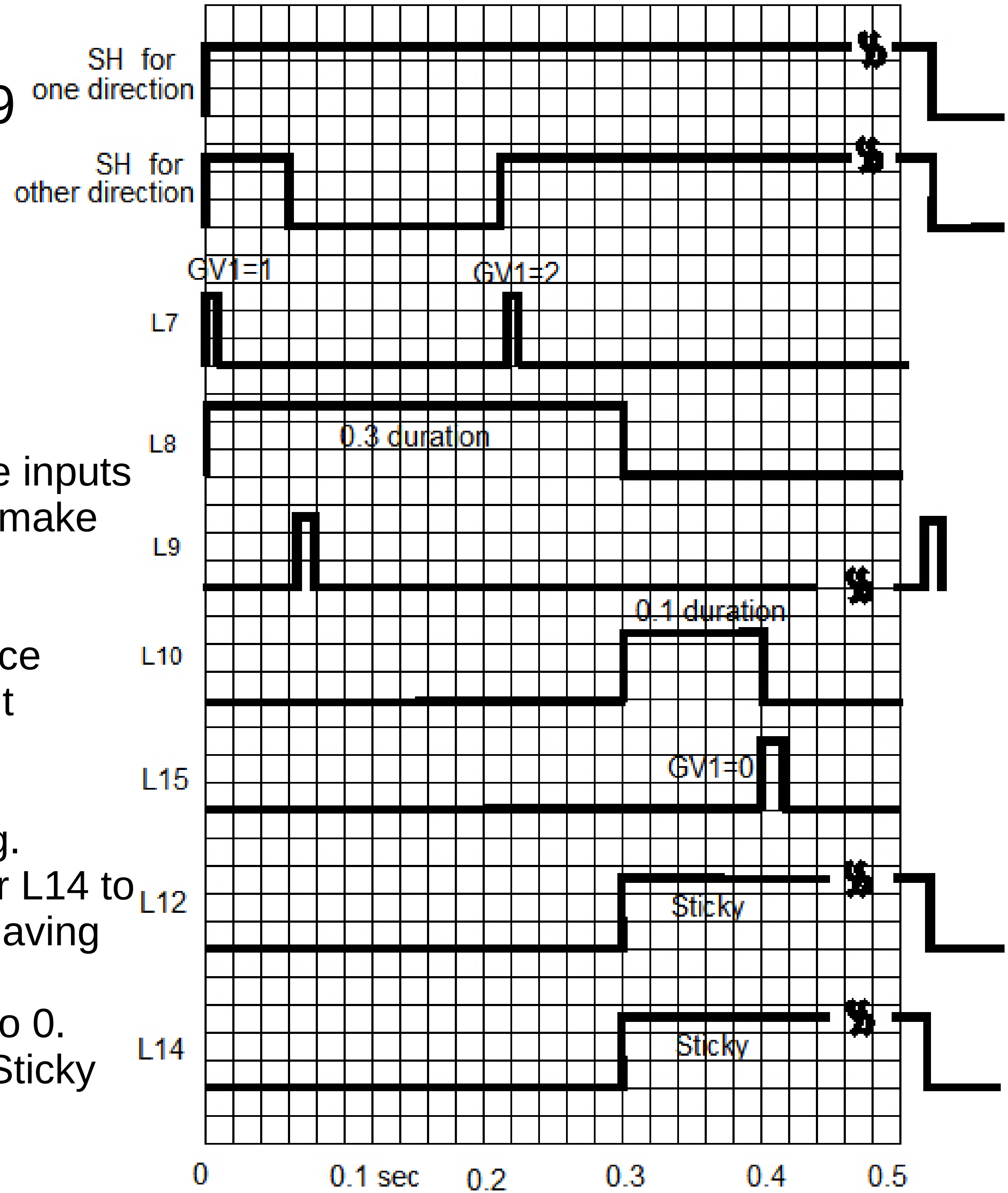
These Logical Functions implement a Single Channel compound Escapement Emulation.
A pull and hold of Momentary Switch SH will move the rudder one way.
Two quick pulls in succession will move the rudder in the opposite direction.

| | | | | | | |
|---|---|---|---|---|---|---|
| L5 | a=x | GV1 | 1 | | ---- | 0.0 |
| L6 | a=x | GV1 | 2 | | ---- | 0.0 |
| L7 | Edge | SH↓ | 0.0 | (instant) | ---- | 0.0 |
| L8 | Edge | SH↓ | 0.0 | (instant) | ---- | 0.3 |
| L9 | Edge | SH↓ | 0.0 | 0.0(infinite) | ---- | 0.0 |
| L10 | Edge | L8 | 0.0 | 0.0(infinite) | ---- | 0.1 |
| L11 | AND | L10 | L5 | | SH↓ | 0.0 |
| L12 | Sticky | L11 | L9 | | ---- | 0.0 |
| L13 | AND | L10 | L6 | | SH↓ | 0.0 |
| L14 | Sticky | L13 | L9 | | ---- | 0.0 |
| L15 | Edge | L10 | 0.0 | 0.0(infinite) | ---- | 0.0 |

# Logical Switch Tutorial 3 – p9
# Edge Function
# Compound Escapement
# Emulation Example
# Timing Diagrams

Notice – Logical Switch outputs can be inputs for other Logical Switches so you can make Complex timing sequences.

Momentary On Switch SH is pulled once or twice to start the Rudder movement
The first pull of SH does two things.
It activates L7 to increment GV1 and starts L8 creating a pulse 0.3 sec. long.
The end of L8 sets either Sticky L12 or L14 to move the rudder, depending on GV1 having a value of 1 or 2.
End of L10 triggers L15 to reset GV1 to 0.
Release of SH pulses L9 to reset the Sticky of L12 or L14 to return the rudder to center position.



# Logical Switch Tutorial 3 – p10
# Toggle Function using Four different Edge Functions

This function enables the Single Throw Momentary Switch SH
to independently command three Logical Switches ON and OFF

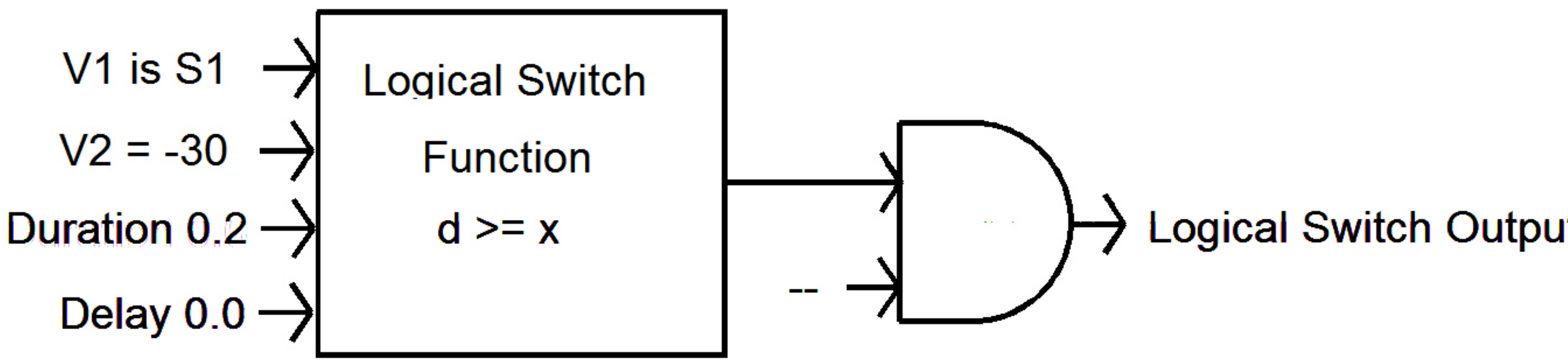
| # | Function | V1 | V2 | | AND Switch |
|---|---|---|---|---|---|
| L1 | Edge | SH↓ | 0.7 | (instant) | -- SH held 0.7 sec for beep signal |
| L2 | Edge | SH↓ | 1.7 | (instant) | -- SH held 1.7 sec for beep signal |
| L3 | Edge | SH↓ | 0.0 | 0.7 | -- SH released before 0.7sec |
| L4 | Sticky | L3 | L3 | | -- Sticky ON-OFF by L3 |
| L5 | Edge | SH↓ | 0.7 | 1.7 | -- SH released between 0.7 and 1.7 sec |
| L6 | Sticky | L5 | L5 | | -- Sticky ON-OFF by L5 |
| L7 | Edge | SH↓ | 1.7 | 1.7(infinite) | -- SH released after 1.7 sec |
| L8 | Sticky | L7 | L7 | | -- Sticky ON-OFF by L7 |

L1 waits for 0.7 sec then is active for one OTX cycle, triggering a beep Special Function
L2 waits for 1.7 sec then is active for one OTX cycle, triggering a beep Special Function
L3 is active before 0.7 sec for one OTX cycle if SH is released before 0.7 sec beep
L4 Sticky is set or reset by L3 pulse
L5 is active after 0.7 sec and before 1.7 sec for one OTX cycle if SH is released after the 0.7 sec beep and before 1.7 sec. beep
L6 Sticky is set or reset by L5 pulse
L7 is active after 1.7 sec for one OTX cycle if SH is released after the 1.7 sec beep
L7 Sticky is set or reset by L6 pulse

# Logical Switch Tutorial 4 – p1
# d >= x

The software checks input VI every 30 ms to see if it has changed by at least V2.  If V2 is positive, it looks for a positive change. If negative it looks for a negative change. If such a change is seen, it signals a TRUE Logical Switch output

| # | Function | V1 | V2 | AND Switch | Duration | Delay |
|---|---|---|---|---|---|---|
| L1 | d>=x | S1 | -30 | ---- | 0.2 | 0.0 |




A negative change of -30 in Rotary Potentiometer S1 setting sets L1 true

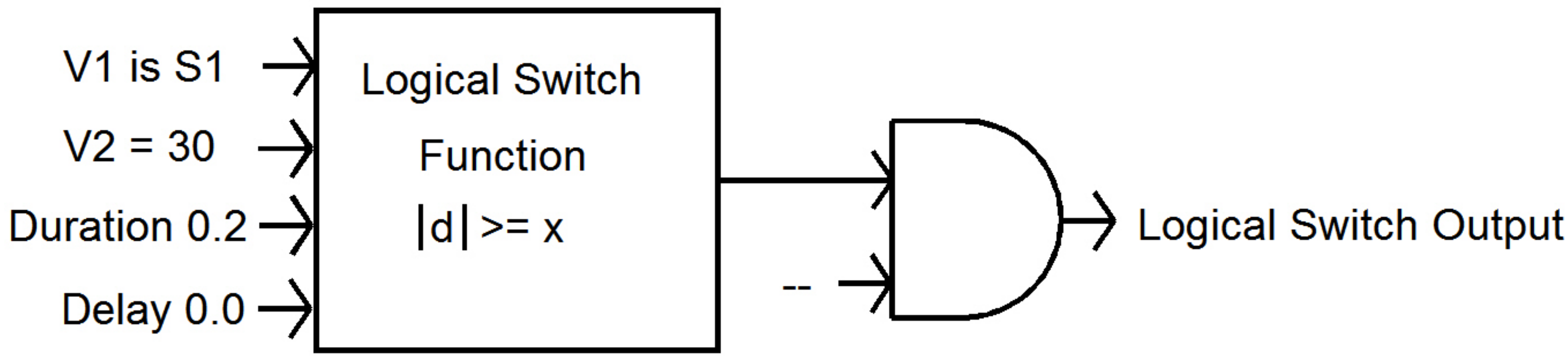
If Duration = 0.0, L1 is true for one cycle of the program

If Duration value is entered (eg 0.2 sec) then L1 will be true for Duration

# Logical Switch Tutorial 4 – p2
# |d| >= x

The software checks input VI every 30 ms to see if it has changed by at least the magnitude of V2, either positive or negative.  If so it signals a TRUE Logical Switch output

| # | Function | V1 | V2 | AND Switch | D |
|---|---|---|---|---|---|
| L1 | \|d\|>=x | S1 | 30 | ---- | 0.2 |



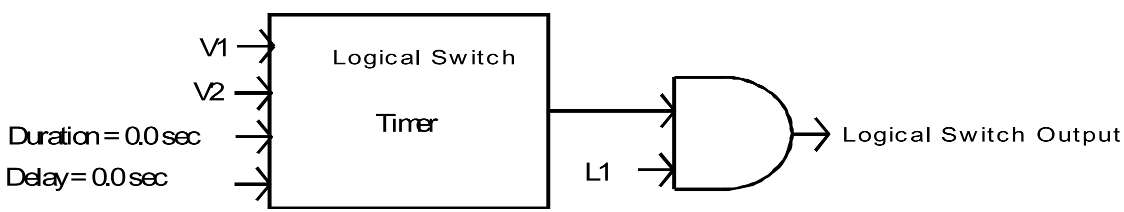

A change of 30 in either direction of Rotary Potentiometer S1 sets L1 TRUE

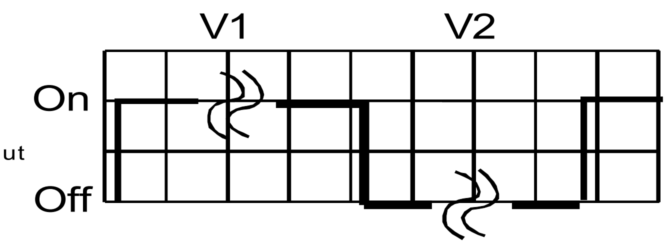
If Duration = 0.0, L1 is true for one cycle of the program

If Duration value is entered (eg 0.2 sec) then L1 will be true for Duration

# Logical Switch Tutorial 5 – p1
# Timer

| # | Function | V1 | V2 | AND Switch |
|---|---|---|---|---|
| L1 | a<b | S1 | S2 | ---- |
| L2 | Timer | 1.0 | 2.0 | L1 |



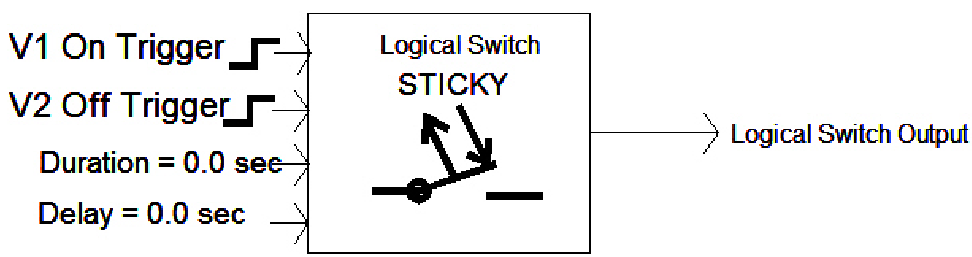

V1 sets the ON time of the timer

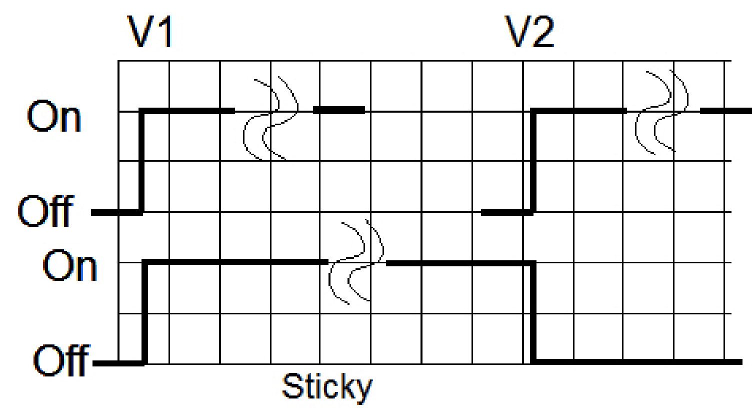
V2 sets the OFF time of the timer

L1 is used as a gate to start and stop the timer output

# Logical Switch Tutorial 6 – p1
# Sticky

| | | | | |
|---|---|---|---|---|
| L7 | a<x | [I3]Thr | -90 | ---- |
| L8 | AND | !L7 | SA↑ | ---- |
| L9 | AND | L7 | !SA↑ | ---- |
| L10 | Sticky | L8 | L9 | ---- |



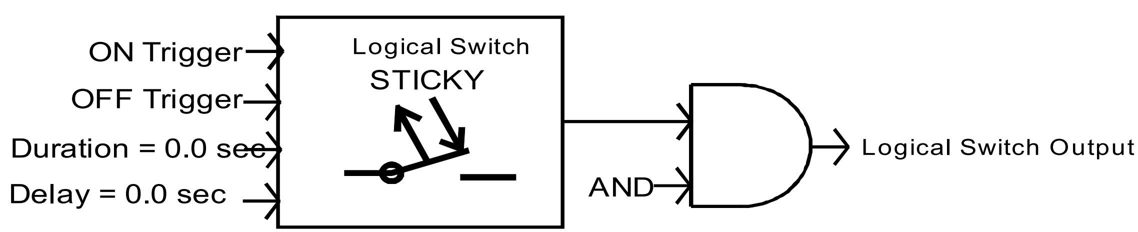




The Sticky Function is like a toggle switch but two different commands can control it
It is set ON with leading edge of V1 TRUE,
It is canceled OFF with leading edge of V2 TRUE
It ignores any changes to V1 when the sticky is TRUE
It ignores any changes to V2 when the Sticky is FALSE

L8 will Set L10 when Throttle Greater than -90 and Switch SA↑
L9 will Cancel L10 when Throttle Less than -90 and Switch !SA↑

# Logical Switch Tutorial 6 – p2
# Warning concerning Sticky Use with AND

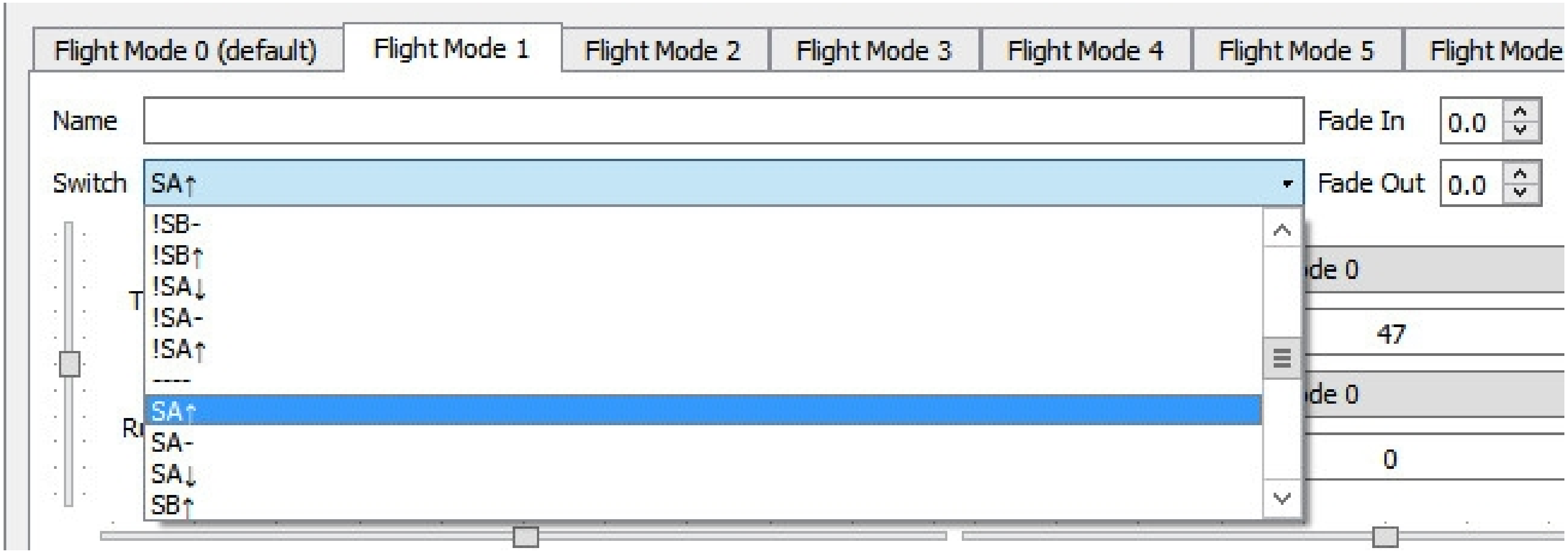
| | | | | |
|---|---|---|---|---|
| L7 | a<x | [I3]Thr | -90 | ---- |
| L8 | AND | !L7 | SA↑ | ---- |
| L9 | AND | L7 | !SA↑ | ---- |
| L10 | Sticky | L8 | L9 | ---- |




The state of the STICKY function can be hidden by the AND gate.
If you have an input on the AND gate and set that input OFF, the output will not represent the state of the STICKY. This could be dangerous if you are using the STICKY to lock a condition for some reason.

# Logical Switch Tutorial 7 – p1
# How to use the Logical Switches

- Both physical switches (SA etc) and Logical Switches (L1 etc) are used to create “Flags” for the computer to notice regularly. When it sees a Flag (a switch position, or state) it checks to see what it should do when that Flag is encountered and does the specified action.
- For example, a simple task would be to select a Flight Mode. Simply select physical switch SA↑ to activate FM1 and when you move switch SA to SA↑ the system will switch to Flight Mode 1.



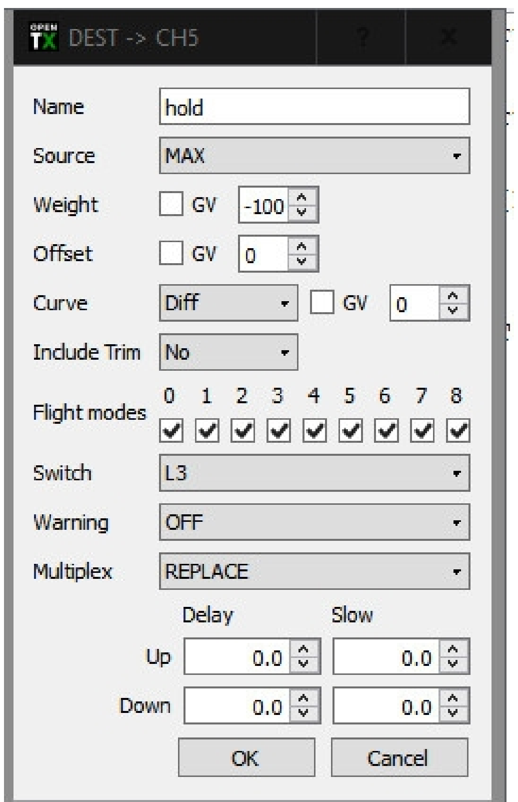
# Logical Switch Tutorial 7 – p2
# Throttle Lock-out Example

- A Logical Switch can also be used to select a change in system configuration. Consider the Throttle Lock-Out below.

Mix

```
CH05            [I3]Thr Weight(+100%)
             := MAX Weight(-100%) Switch(L3) NoTrim [hold]
```

Mix Definition Table




Logical Switch Definition Table

| # | Function | V1 | V2 | |
|---|---|---|---|---|
| L1 | a<x | [I3]Thr | -98 | ---- |
| L2 | Edge | SH↓ | 0.8 (instant) | L1 |
| L3 | Sticky | L2 | L2 | ---- |
| L4 | Edge | SH↓ | 0.0 0.7 | L3 |

Special Function Definition Table

| # | Switch | Action | Parameters | |
|---|---|---|---|---|
| SF1 | L2 | Play Sound | Beep 3 | No repeat |
| SF2 | L4 | Play Track | thrhold ▶ | No repeat |

# Logical Switch Tutorial 7 – p3
# Throttle Lock-out Example -continued

- A Replacement Mix is defined for the Throttle. This is activated when the L3 Logical Switch is seen TRUE.
- Logical Switch L1 is TRUE when Throttle has a value less than -95 and stays TRUE until the Throttle value becomes greater than -96.
- Logical Switch L2 is set by an EDGE function. When the momentary switch SH is pulled (SH↓) it waits for 0.8 seconds before it sets L2 True for one cycle of the processor time. (approx. 30 msec). If SH is released before 0.8 sec. L2 is not set. L2 is only set TRUE if L1 is TRUE
- Logical Switch L3 is a Sticky, set to TRUE when it sees L2 set TRUE. L3 then waits for the next time L2 is set TRUE to set L3 back to FALSE.
- Logical Switch L4 is an edge function that is set to TRUE for once cycle if SH↓ is released before
  0.7 sec if L3 is TRUE. This causes a Special Function Voice warning when TRUE.
- Logical Switch L2 causes a Special Function audible beep if L2 becomes TRUE.
- Logical Switch L3 replaces the continuous Throttle Mix with a MAX = -100 Mix
- Logical Switch L4 causes a Special Function Voice warning if L4 becomes TRUE.

- The Logical Switches are used in quite different ways.
  L1 just watches for a particular state of the Throttle and signals when it is seen.
  L2 watches the state of SH and signals at a specified time.
  L3 locks a particular state of its signal until it is deliberately told to unlock it. It commands a change of state of the Throttle
  L4 signals to activate a Special Function voice warning.

- Notice
  The Logical switches are very versatile. Here they work together with physical switches, Logical Switches and analogue values in very complicated ways to create a simple but quite sophisticated Throttle Lock Out function.

# Logical Switch Tutorial 7 – p4
# Another Throttle Lock-out Example

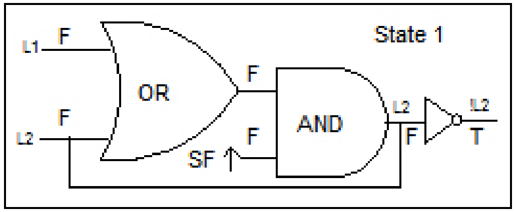
Here is another very interesting Throttle Lock-Out.  It uses the L2 output as feedback to an L2 input.  Note: !L2 controls the Mix replacement Function

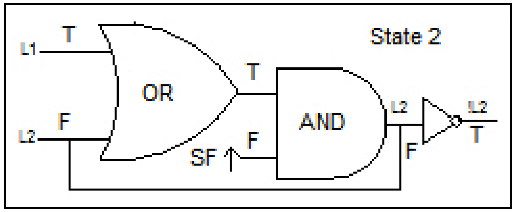
Mix

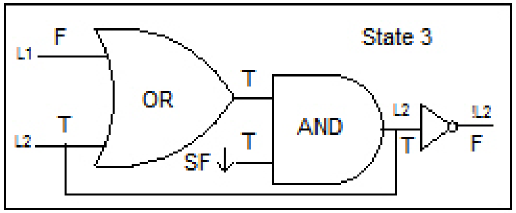
```
CH03            [I3]Thr Weight(+100%)
             := MAX Weight(-100%) Switch(!L2) [cutoff]
```

Logical Switch Definition Table

| # | Function | V1 | V2 | AND Switch | |
|---|---|---|---|---|---|
| L1 | a<x | [I3]Thr | -98 | ---- | 0 |
| L2 | OR | L2 | L1 | SF↓ | 0 |




- State 1: Throttle > -98 so L1 is FALSE, SF↓ is FALSE so L2 is FALSE and !L2 is TRUE holding the Throttle at -100
- State 2: Throttle < 98 making L1 TRUE but SF↓ is FALSE so !L2 is TRUE holding Throttle still at -100
- State 3: SF↓ is TRUE so OR TRUE output can pass through the AND making !L2 FALSE.  The cutoff Mix is disabled allowing the Throttle changes to be active
- Notice the use of the !L2 Logic output.  See Page 15