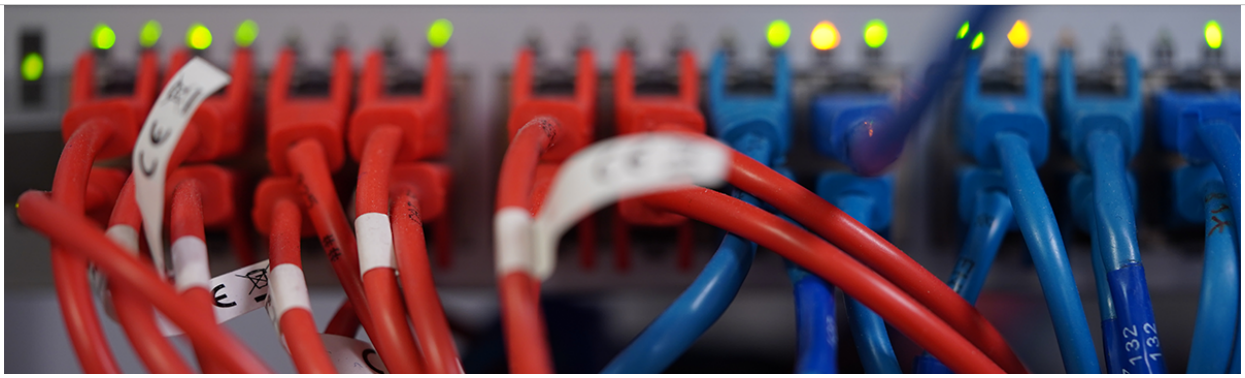
TSL 3.1 and 5.0 Support in UniSketch



Introduction

TSL is a widely used protocol in the broadcast industry for sending tally information and under-monitor labels around on a broadcast network. The protocol is the work of https://tslproducts.com and freely available. In essence one device on the network can send information to turn on or off a red, green or amber tally light by an address number. Included with the message is also a text label field which a multiviewer or under-monitor display (UMD) could pick up.

Please note that to currently access some important features of the TSL configuration you will need to access the Online Configuration by putting the firmware updater into Expert Mode via Option and selecting Online Configuration (Staging). When you have saved the configuration you need to update the firmware via Update Firmware from Master.

UniSketch Implementation

Our implementation allows any SKAARHOJ UniSketch based product to receive (TSL server) and/or send (TSL Client) TSL v3.1 and v5.0 messages over UDP and/or Multicast. The implications are vast: Since we both receive and send messages, we can offer basic protocol conversion to/from TSL and *any* other device core we already offer. Examples:

- Listen for TSL messages in a Tally Box and associate them with SKAARHOJ Tally Lamps for your cameras
- Create Virtual Triggers on any SKAARHOJ UniSketch product to send TSL messages out on the network when certain states change in other device cores; like when an input source is on program on an ATEM switcher, when a given route is set on a video router, when a camera is set to auto focus, when a streaming device is streaming. You can essentially TSL-enable *any* product supported this way!
- Listen for TSL messages and let them trigger other actions: You could start/stop recording on an AJA KiPro when a given TSL message sets a bit on/off. You could flip a relay on a GPI box.
- Listen for a TSL message on a given address to color the tally LED in an RCP and put a label in the ID display.
- Implement button color feedback on a switcher panel using Ross Talk (sending one way) by listening for TSL messages in return.
- Send TSL messages on a button press for whatever reason. Like a GPI input on a ETH-GPI Link

TSL is a bit unusual for SKAARHOJ device cores in that you will only rarely want to assign a TSL send action to a button. You will rather want to automatically issue them along with state changes, hence the talk about Virtual Triggers. On the receiving side there are more classic uses like for tally lamps or indicators like the LED bar and ID display on an RCPv2.

Honestly, SKAARHOJ is by no means experts in TSL and its uses. Our impression is that it has uses far beyond tally signals and UMD labels. Our implementation is designed to bring about the most flexibility we can offer, while trying to keep it simple for the most obvious uses, following the protocol and it's apparent intention as well as possible.

TSL protocol and scope of implementation

Our implementation is based solely on this description:

https://tslproducts.com/media/1959/tsl-umd-protocol.pdf

Version 3.1 implementation:

- Implemented over UDP. Specification only describes an electrical interface. It's assumed that a UDP-Serial converter could be used to implement this for a SKAARHOJ device although it has to be tested.
- Supports all four bits
- Incoming messages are stored in the same memory space as V5.0 messages as if they had Screen ID 0.

Version 5.0 implementation:

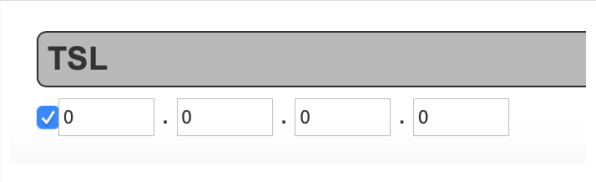
- Implemented over UDP only. (No TCP implementation available).
- Supports max packet lengths of 128 bytes (not 2048 bytes as protocol describes). Support for multiple embedded indexes (displays) per screen ID as long as packet size is not exceeded (to be tested and confirmed).
- Supports Off / Red / Green / Amber colors
- Supports Left, Right and Text locations
- Doesn't support screen control, only screen data (UMD text)
- Supports only Screen IDs and Index IDs from 0-255

General:

- Supports receiving and/or sending messages from/to a Multicast group.
- Supports direct point-to-point UDP messages as well, sending to up to 21 different IP addresses.
- Stores max 64 TSL messages with up to 16 characters ASCII text, prioritized by which ones are being actively used in UniSketch.
- Brightness data is set fixed to full brightness on outgoing messages and ignored for incoming messages.

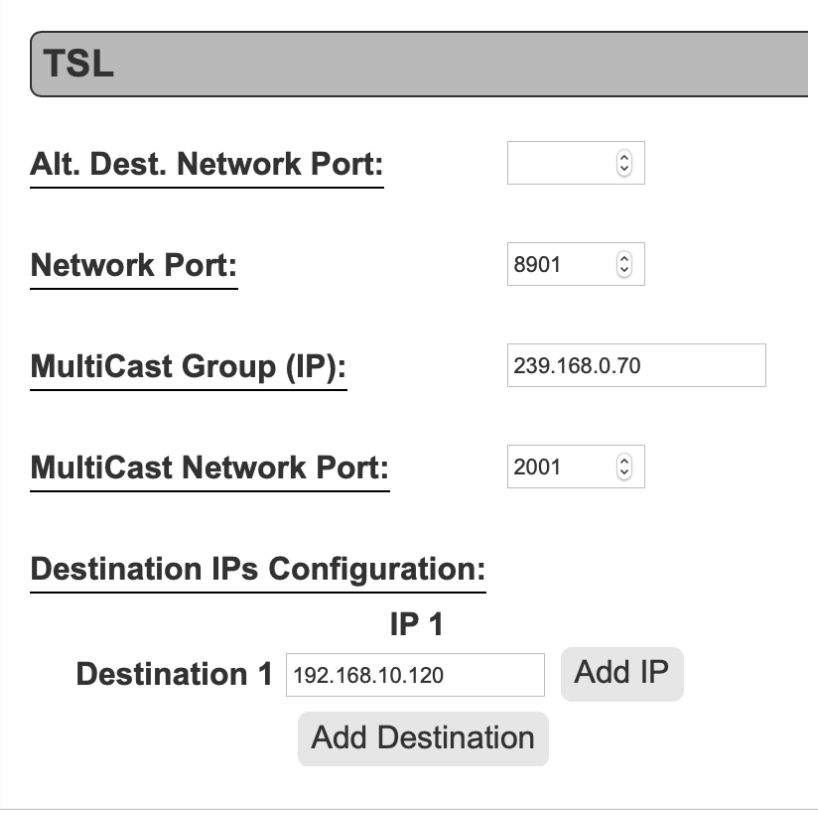
Configuration

As always, you need to *enable* the device core for it to be active.



If you type in an IP address here, it will be the default destination ("Destination 0") for outgoing messages.

In addition to defining the destination IP address you also *need* to set the network port to send to. In fact, before anything will work you will need to set up a number of device core options:



- If you wish to either listen to or send multicast messages you need to set up the multicast group address and a port number (in the example, this is 239.168.0.70 and port 2001). The port is used for both sending and receiving multi cast messages. To use multicast with the TSL core you only need these two things set up and to enable the device core.
- If you wish to send TSL messages to specific IP addresses, you need to set up the Network Port (in the example above, this is set to 8901) and define one or more destination IP addresses. The default "Destination 0" is the main device core IP address (in the example, 192.168.10.110) while up to 20 alternative destinations can be defined as you like.

If you define "Alt. Dest. Network Port" this will be the port you send TSL messages to on all destination IP addresses. In that case "Network Port" will only be the sending (and listening) port on the SKAARHOJ device

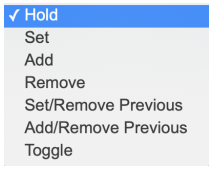
- If you wish to listen to TSL messages you don't need to set up any destination IP addresses, but you need to set up the "Network Port" which is in all cases the UDP port on the SKAARHOJ unit used to listen for TSL messages (except for multicast of course which has its own port).

Sending TSL 5.0 messages

To send TSL 5.0 messages you add this action anywhere you may receive a binary trigger: Buttons, GPI inputs, Virtual Triggers, etc. (no native support for encoders, faders and joysticks). The display and color output will mostly be informative about the selected options.

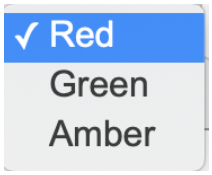
Options:

✓ Destination: 0 Destination: 1 Destination: 2 Destination: 3 Destination: 4 Destination: 5 Destination: 6 Destination: 7 Destination: 8 Destination: 9 Destination: 10 Destination: 11 Destination: 12 Destination: 13 Destination: 14 Destination: 15 Destination: 16 Destination: 17 Destination: 18 Destination: 19 Destination: 20 Multicast	Where to send the TSL message. • Destination 0: The standard IP address of the device core • Destination 1-20: An alternative IP address from the table entered under Device Options. • Multicast: Send to the multicast group / port
✓ Screen ID: 0 Screen ID: 1 Screen ID: 2 Screen ID: 3 Screen ID: 4 Screen ID: 5	This is the screen address in the TSL message
✓ Index ID: 0 Index ID: 1 Index ID: 2 Index ID: 3 Index ID: 4 Index ID: 5	This is the index address in the TSL message



The action modifier determines how the action handles the incoming trigger. A binary trigger consists of a “trigger down” (e.g. button press) and “trigger up” (e.g. button release):

- Hold: On “trigger down” the tally flags (Red / Green / Amber) on the selected positions (Left / Right / Text) will be set. On “trigger up” the set tally flags will be cleared again.
  Notice Red and Green tallies will in this case be set and cleared without interfering with each other. Setting Red while Green is on will yield Amber (by nature of the binary structure of these values)
- Set: On “trigger down” the tally value (Red / Green / Amber) on the selected positions (Left / Right / Text) will be set. Notice that setting red will in this case clear away a green value.
- Add: Like “Hold”, but without clearing the flag on release. Red and Green will not interfere in this case and setting both of them will yield Amber.
- Remove: Like “Hold”s trigger release: It will clear out the selected tally flags. Clearing red will not clear green. If you wish to make sure to clear all flags, choose to clear for Amber.
- Set/Remove Previous + Add/Remove Previous: Like the Set and Add actions, but will also clear the flags sent previously in the last action.
- Toggle: This will toggle on and off the tally flags on repeated triggers.



The tally flags affected by the operation. See previous description of modifiers.

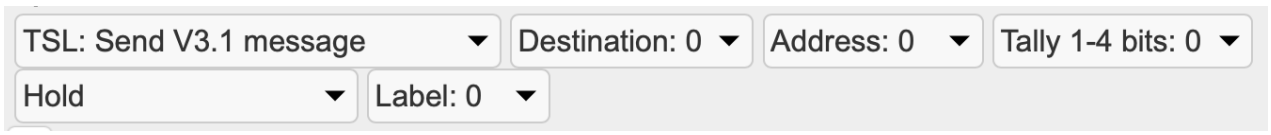
Technically Red is the binary combination “01” while green in the binary combination “10” and Amber is “11”. Therefore, when you set red (01) and green (10) separately with the actions above, it will yield Amber (01 OR 10 = 11)

TSL v5.0 seems to be aimed at Multiviewer products which may have various locations for showing tally. In the protocol locations like “Left”, “Right” and “Text” are specified. This option allows you to select which of these locations to affect with the changes to the tally flags.

Select which label to send along with the message. All TSL messages have the capability to send text labels.

- Label: 0 = No label (empty string)
- Label: 1-100 = Any of the string labels baked into the firmware via cores.skaarhoj.com configuration page
- Grid x,x: A label from the dynamic labels you can enter into the controllers webinterface offline.

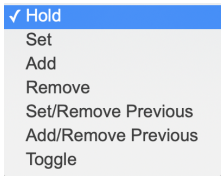
Sending TSL 3.1 messages



To send TSL 3.1 messages you add this action anywhere you may receive a binary trigger: Buttons, GPI inputs, Virtual Triggers, etc. (no native support for encoders, faders and joysticks). The display and color output will mostly be informative about the selected options.

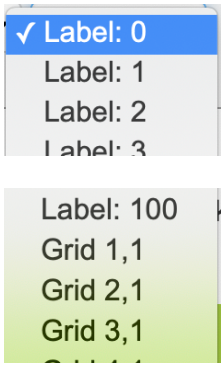
Options:

✓ Destination: 0 Destination: 1 Destination: 2 Destination: 3 Destination: 4 Destination: 5 Destination: 6 Destination: 7 Destination: 8 Destination: 9 Destination: 10 Destination: 11 Destination: 12 Destination: 13 Destination: 14 Destination: 15 Destination: 16 Destination: 17 Destination: 18 Destination: 19 Destination: 20 Multicast	Where to send the TSL message. • Destination 0: The standard IP address of the device core • Destination 1-20: An alternative IP address from the table entered under Device Options. • Multicast: Send to the multicast group / port
✓ Address: 0 Address: 1 Address: 2 Address: 3	This is the address in the TSL message
✓ Tally 1-4 bits: 0 Tally 1-4 bits: 1 Tally 1-4 bits: 2 Tally 1-4 bits: 3 Tally 1-4 bits: 4 Tally 1-4 bits: 5 Tally 1-4 bits: 6 Tally 1-4 bits: 7 Tally 1-4 bits: 8 Tally 1-4 bits: 9 Tally 1-4 bits: 10 Tally 1-4 bits: 11 Tally 1-4 bits: 12 Tally 1-4 bits: 13 Tally 1-4 bits: 14 Tally 1-4 bits: 15	Which bits in the TSL message to set. Unlike TSL v5, the bits are unmarked, so red/green/amber or whatever is apparently up to the user to define. With this option you simply select the bit-pattern you wish to send. Values 0-15 cover 4 bits. In Ross Video documentation this convention is in use: *“Tally 1 is preview and Tally 2 is program.”*



The action modifier determines how the action handles the incoming trigger. A binary trigger consists of an “trigger down” (e.g. button press) and “trigger up” (e.g. button release):

- Hold: On “trigger down” the tally bits 1-4 will be set (OR operation). On “trigger up” only the set tally bits will be cleared again (AND operation with inverted mask).
  Notice that the selected bits will in this case be set and cleared without interfering with each other.
- Set: On “trigger down” the total value of the tally bits 1-4 will be set, overriding any existing value of the four bits.
- Add: Like “Hold”, but without clearing the bits on release.
- Remove: Like “Hold”s trigger release: It will clear out the selected tally flags. If you wish to make sure to clear all flags, choose to clear for value 15.
- Set/Remove Previous + Add/Remove Previous: Like the Set and Add actions, but will also clear the bits sent previously in the last action.
- Toggle: This will toggle on and off the tally bits on repeated triggers.



Select which label to send along with the message. All TSL messages has the capability to send text labels.

- Label: 0 = No label (empty string)
- Label: 1-100 = Any of the string labels baked into the firmware via cores.skaarhoj.com configuration page
- Grid x,x: A label from the dynamic labels you can enter into the controllers webinterface offline.

Listening for TSL 5.0 messages

To listen for TSL 5.0 messages you add this action to any “output” in UniSketch: As sources for Virtual Triggers, for GPI outputs, for LEDs, displays, even buttons that you want to “paint” with the state of the TSL message.

The label following a TSL message will be output in the associated display of the hardware component the listening action is assigned to.

Listening to a TSL message consists of two steps:

- Matching with the incoming message: Do you want a positive match only if green tally is set - or if any flag is set - and on any arbitrary position?
- If there is a match, what do you want to do with it? Shall a LED be colored the same color (Red / Green / Amber) or shall a red tally flag lead to just a highlighted button in any system color?

This is what the extensive options define:

Options:

✓ Screen ID: 0 Screen ID: 1 Screen ID: 2 Screen ID: 3 Screen ID: 4 Screen ID: 5	This is the screen address in the TSL message
✓ Index ID: 0 Index ID: 1 Index ID: 2 Index ID: 3 Index ID: 4 Index ID: 5	This is the display address in the TSL message
✓ Match Red Only Match Red Bit Match Green Only Match Green Bit Match Amber Match any color	How to match the TSL message and determine if it should activate a trigger: • Match Red Only: Returns true only if Red is the color (binary 01) • Match Red Bit: Returns true if the red bit is set regardless of green (binary x1) • Match Green Only: Returns true only if Green is the color (binary 10) • Match Green Bit: Returns true if the green bit is set regardless of red (binary 1x) • Match Amber: Returns true only if flags are Amber (binary 11) • Match any color: Returns true as long as any color / bit is set (binary xx different from zero)
✓ Match Any position Match with Left Match with Right Match with Text	Defines which positions to match for. Matching for any position will OR the binary values of all positions together before comparison.
✓ Output Red Output Green Output Amber Output same color	If there is a positive match based on the criterias set with the options above, this option will determine the output color. • None: Returns system default color • Output Red: Returns Red on any match • Output Green: Returns Green on any match • Output Amber: Returns Amber on any match • Output same color: Returns a color Red/Green/Amber/Off according to the incoming color The binary output of an action (the state that defines if a relay in a GPI box is closed) solely depends on the matched state and doesn't care about the actual color.

Listening for TSL 3.1 messages

To listen for TSL 3.1 messages you add this action to any "output" in UniSketch: As sources for Virtual Triggers, for GPI outputs, for LEDs, displays, even buttons that you want to "paint" with the state of the TSL message.

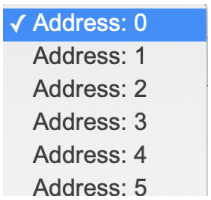
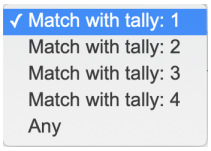
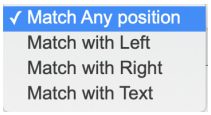
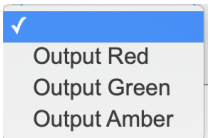
The label following a TSL message will be output in the associated display of the hardware component the listening action is assigned to.

Listening to a TSL message consists of two steps:

- Matching with the incoming message: What combination of the four tally bits shall yield a positive match?
- If there is a match, what do you want to do with it? Shall an LED be colored the same color (Red / Green / Amber) or shall a red tally flag lead to just a highlighted button in any system color?

This is what the extensive options define:

Options:

	This is the address in the TSL message
	How to match the TSL message and determine if it should activate a trigger: • Match with Tally 1-4: Returns true if that particular bit is set (doesn't care about state of the other bits) • Any: Any bit being set yields true.
	Defines which positions to match for. Matching for any position will OR the binary values of all positions together before comparison.
	If there is a positive match based on the criterias set with the options above, this option will determine the output color. • None: Outputs system color • Output Red: Returns Red on any match • Output Green: Returns Green on any match • Output Amber: Returns Amber on any match The binary output of an action (the state that defines if a relay in a GPI box is closed) solely depends on the matched state and doesn't care about the actual color.

Notes on Multicast, Destinations, VPNs and Networks

At SKAARHOJ we are not trying to be experts in network technology in general, so take this only as a description of our experiences and interpretations of them. There may be nuances which are incorrect, and we appreciate any feedback on that to clarify the text.

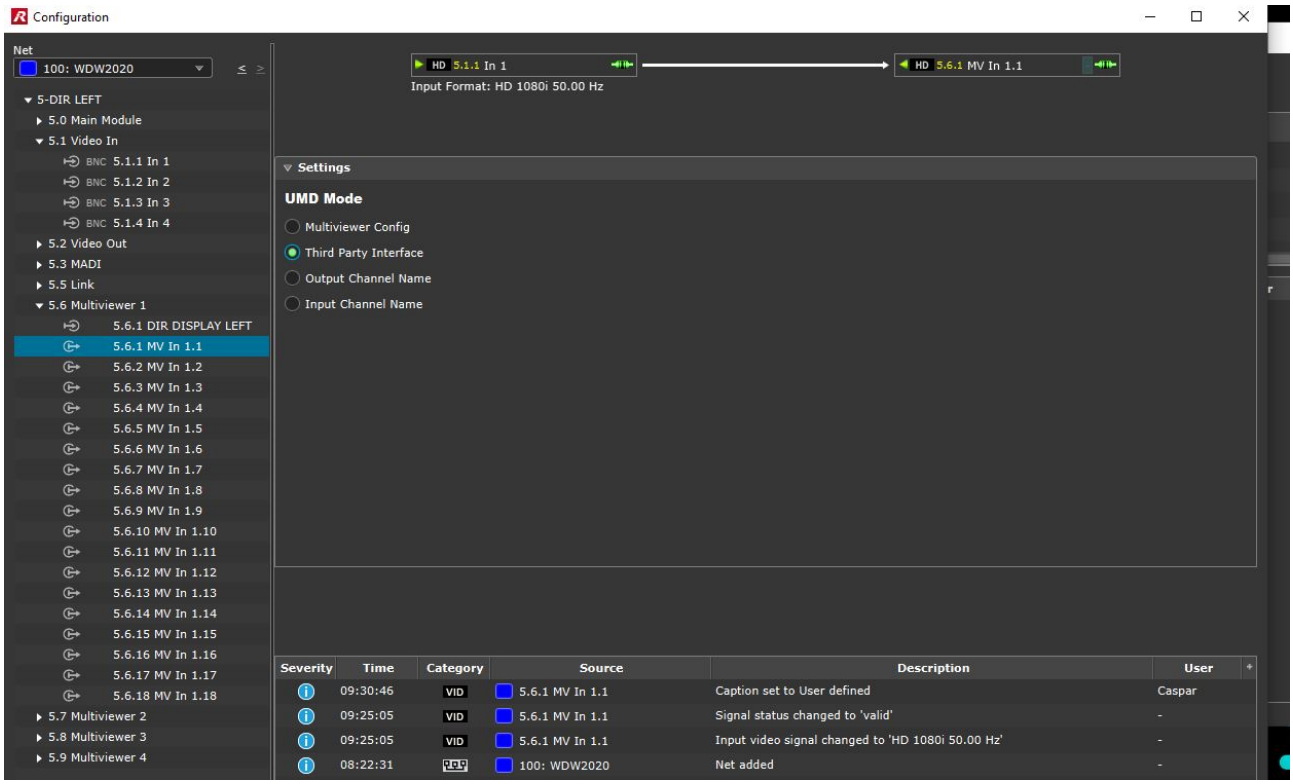
- We have experienced that multicast messages are only distributed on the same subnet. For example, a TSL client sending out messages to multicast group 239.168.0.70 could have an IP address like 192.168.10.99 and a TSL server on a SKAARHOJ controller could have an IP address like 192.168.10.100 (both on same subnet with subnet mask 255.255.255.0) - and messages sent out on this network should reach their destination
- The above case seems to also apply when a subnet is distributed over a VPN - which feels logical since the VPN should be transparent to the units on it.
- We have also experienced that multicast messages will NOT reach their destination if they are sent from one subnet to another, possibly via a router or between VLANs. This may depend on router setup, but we believe in the default case a router will not forward multicast messages to other subnets. For example, if a SKAARHOJ TSL client is on IP address 10.11.0.45 and sending out multicast messages they won't reach a SKAARHOJ TSL server (receiving device) on 192.168.10.100.
- However, if instead of using multicast you send a TSL message directly to a specific destination (in our implementation you can have up to 20 destinations plus the main IP address of the device core) from say 10.11.0.45 to 192.168.10.100 it will reach its destination assuming that they are connected in a way that facilitates connections from the one subnet into the other.
- Using multicast it may be possible that network equipment should be turned on first and then SKAARHOJ units. The concern is that a SKAARHOJ unit with multicast enabled will announce during boot up that it desires to receive multicast traffic sent to its multicast group. In case the router boots up after the SKAARHOJ controller this message would be lost and its not periodically retransmitted. It seems that this could create a situation where the multicast messages would not reach the destination correctly. However, we have actually not experienced that situation, so maybe it's taken care of automatically.

Resources

Ross Video details on TSL 3.1 and 5.0 implementations

- http://help.rossvideo.com/carbonite-device/Topics/Devices/UMD/TSL.html#topic.tsl
- http://help.rossvideo.com/acuity-device/Topics/Devices/UMD/TSL.html#topic.tsl

Riedel Mediornet, typically using port 8901 for TSL 5.0. A customer comment to enable UMD labels in a MicroN multiviewer, you need to set a special mode in the configuration:

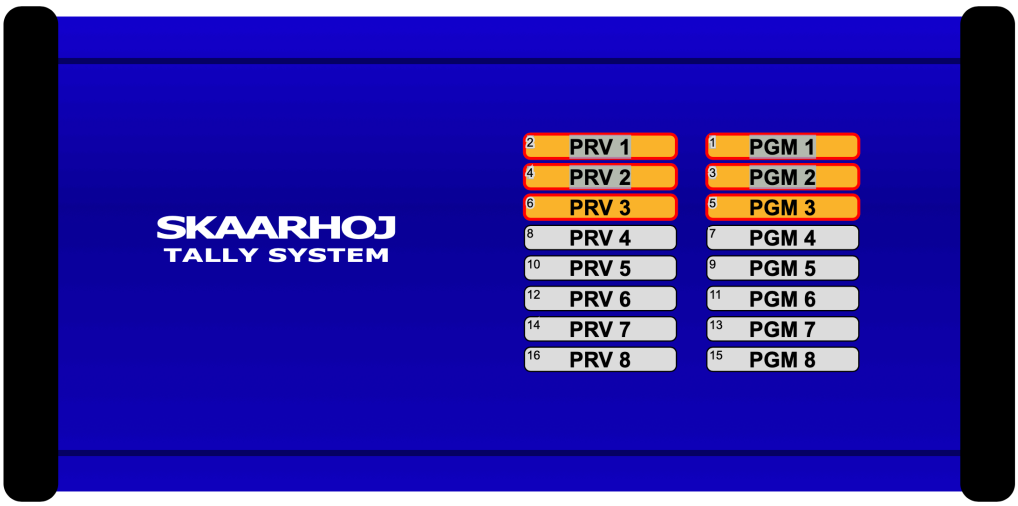


Information about MediorNet TSL implementation can be found in the MediorNet Handbook 3rd Party Interfaces Rev 4.0

Examples

Here are a number of helpful examples of how to use TSL with SKAARHOJ products

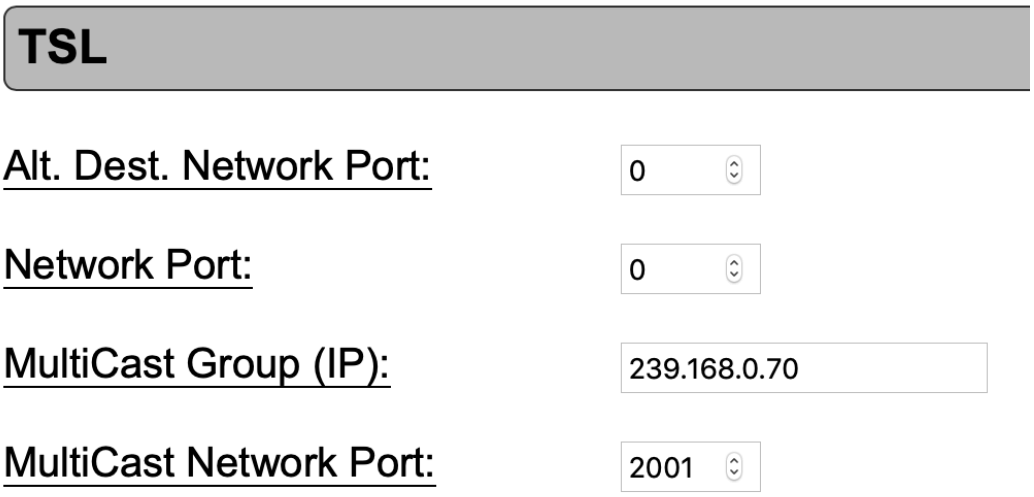
Listening for TSL messages and mapping them to outputs on a SKAARHOJ Tally Box or ETH-GPI Link



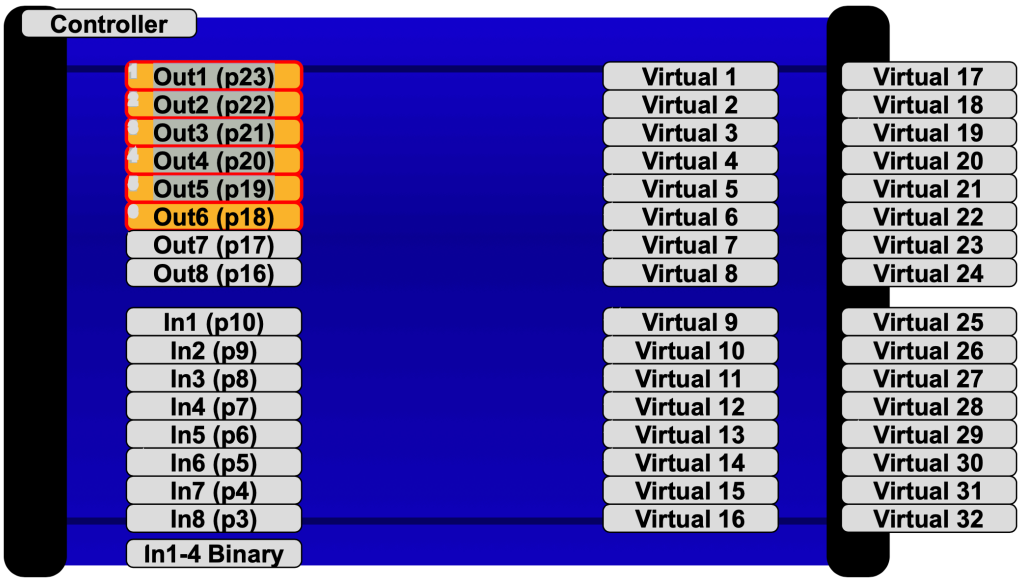
A SKAARHOJ Tally Box, the first three outputs are chosen with Program and Preview LEDs

Assuming that the tally states we want to listen to are sent out to Screen ID 1 and Indexes (Displays) 1,2 and 3, these listening actions will match red and green states of any position to the Tally Box outpus.

It's important to listen for the "Red bit" and "Green bit" instead of just listening to "Red only" or "Green only" because if then both Red and Green are sent for a given index, then none of the lights will light up contrary to the expectation.

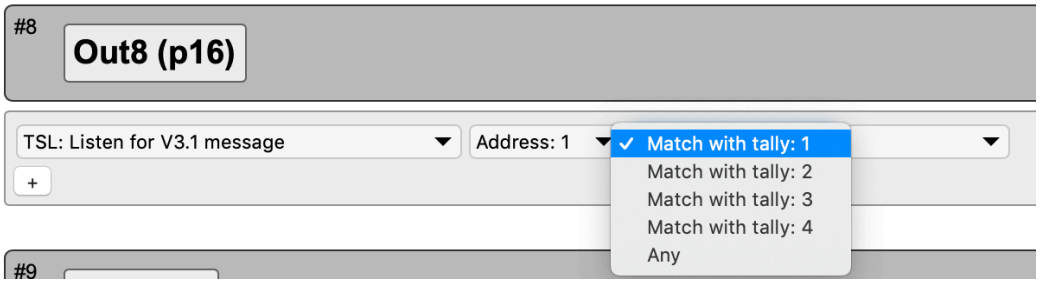


Remember the configuration of the device core: You need to know how the messages arrive; In this case it's assumed that a given Multi Cast group and port is used for the messages on the given network.



The same type of configuration is used on a product like ETH-GPI Link

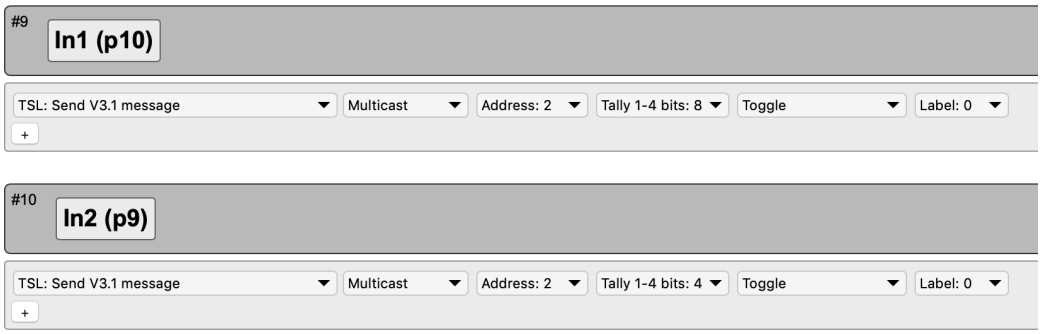
Pairing the state of a TSL 3.1 message with a GPI output on ETH-GPI Link



This action will close the relay if a V3.1 message arrives for address 1 with bit number 1 set.

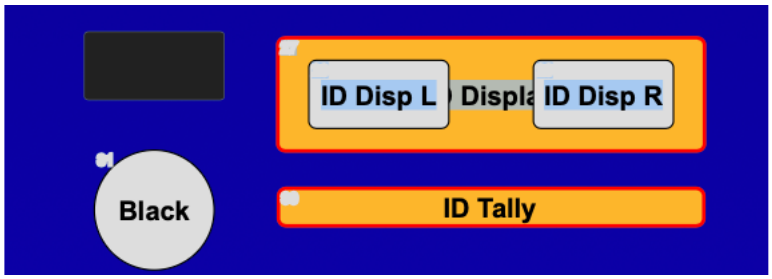
It's assumed that a proper configuration for receiving messages is also set up (see previous example)

Sending TSL messages with GPI inputs on a ETH-GPI Link

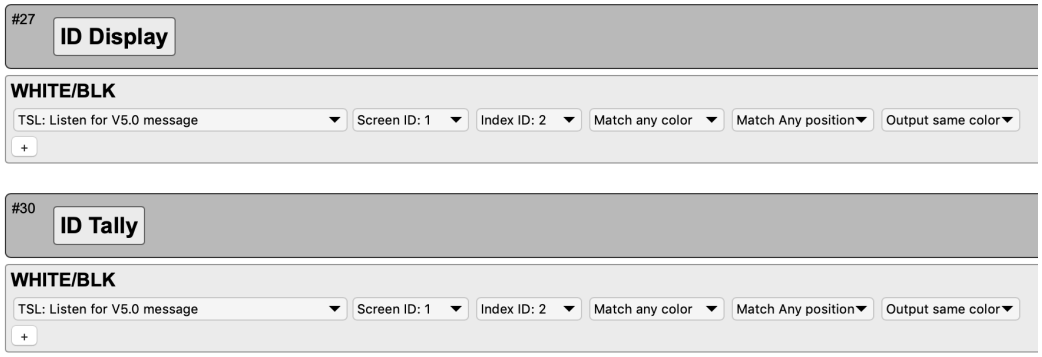


This configuration will send out TSL v3.1 messages on the network to a multicast group when the inputs are closed to GND. The messages are sent to address 2 and with bit 4 set in the first case (binary 1000=8) and bit 3 set in the second case (binary 0100=4). For each time the trigger happens the bits are set or cleared (toggling on and off)

Setting display message and Tally LED on an RCPv2 with TSL messages

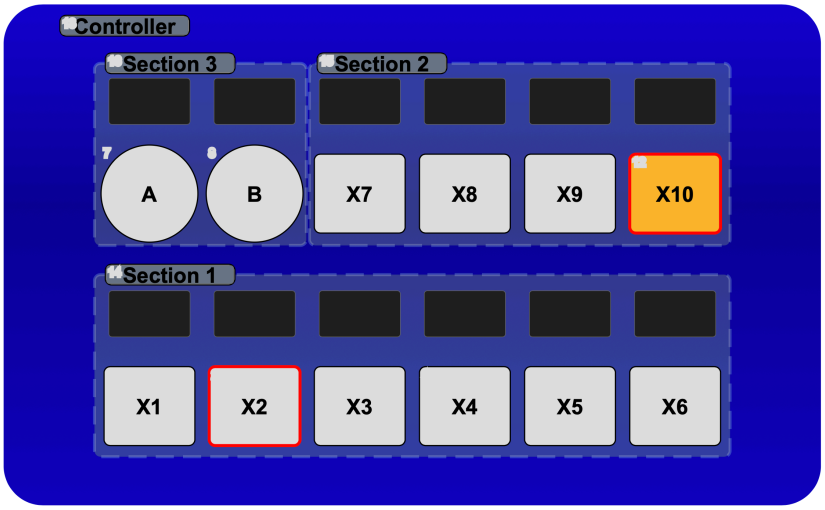


Display and Tally light is set on an RCPv2

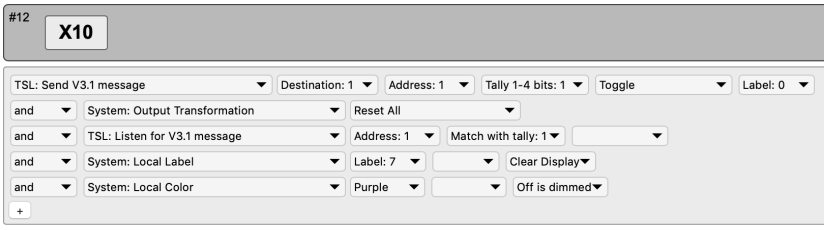


Each item essentially has the same listen-action set, listening for TSL messages sent to Screen ID 1 and Index ID 2. The message set for the display could have any options set beyond that as it only picks up the label included in the messages while the action for the LED is set to match any color, any position and output the same color which essentially passes through the off/red/green/amber state coming in from the message.

Using TSL messages with buttons on a Quick Pad



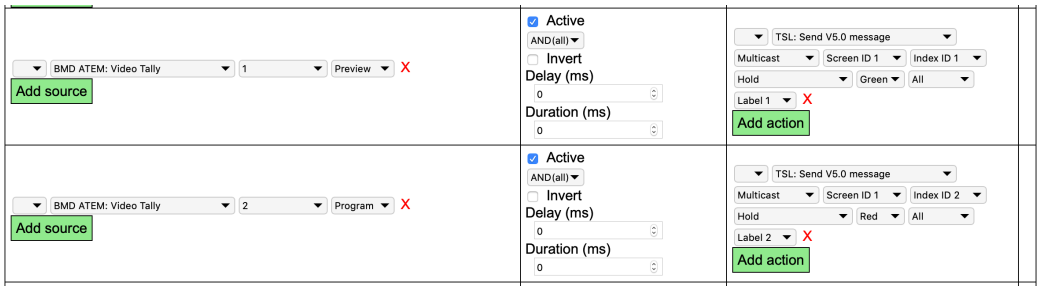
Button X10 on a quick pad is selected



The button will send a TSL v3.1 message to address 1 on destination 1 (a specific IP instead of multicast). In the message, bit 1 will be toggled on/off on repeated button presses.

The rest of the action is about how the button color is set and what is shown in the display. It starts with a “Output Transformation” that clears all previous display and color content from the first action. Then the next “TSL: Listen for..” action essentially reads the state of the bit we just toggled in order to dim/highlight the button. This is followed by system actions that sets a label and color tone

Protocol Translation: Adding TSL sending support to any device



In this configuration we are listening to the video tally state (preview and program) for input 1 on an ATEM switcher and when the state changes we likewise change the state of bits in a TSL v5.0 message.