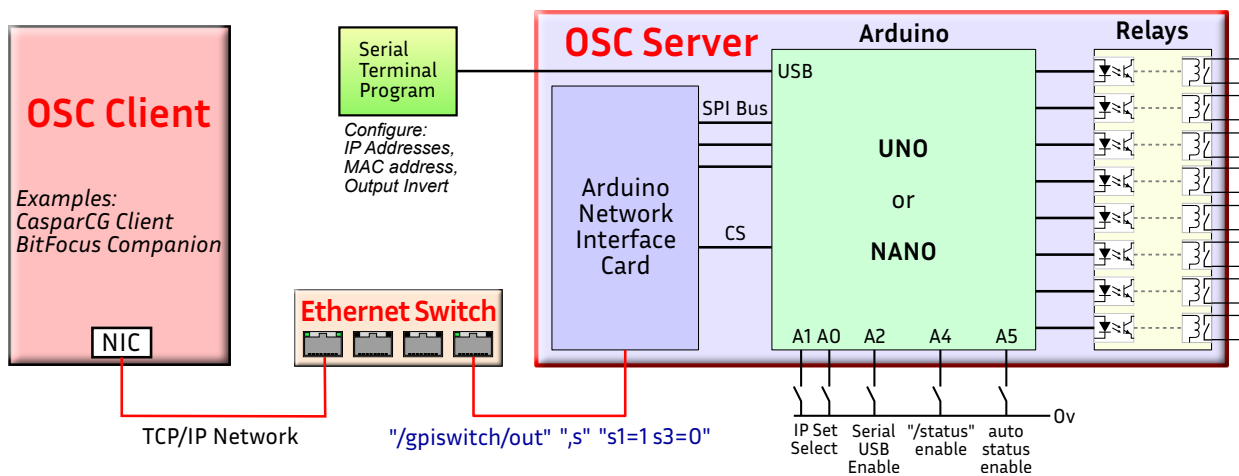


# Low-Cost General Purpose Output (GPO) Interface using OSC Messaging

This note describes a simple network-connected GPO interface with eight outputs implemented using a low-cost Arduino microcontroller with off-the-shelf relay interface board or boards. The state of each GPO output is controlled using OSC messages delivered over UDP.

The relay output states are volatile, and initialised to OFF when the unit is powered or reset. The system block diagram is below.



The diagram shows an example of the OSC control message used by the application. The message contains the OSC address (**"/gpiswitch/out"**), the parameter type (**","s"**) and the switch states for updated switches (**"s1=1 s3=0"**). All eight switches can have their state set by a single OSC message.

Assembling an instance of the interface requires some simple soldering to add the wiring between the Arduino unit and any relay interface board, and simple mechanical skills to mount the hardware in a suitable container.

Some GPO interface instances are fixed into infrastructure and only require the IP address set at installation. Other GPO interface instances are bookable resources deployed where temporary extra outputs are required, and a different IP address block may be in use. This interface uses the USB serial connection from the Arduino to set the IP address, storing the edited addresses in EEPROM inside the microcontroller. DHCP is not supported because many OSC clients only support a fixed IP addressing scheme.

The GPO interface supports deployment on commonly used networks by storing four sets of IP addresses in EEPROM memory. Two single-pole switches select the active TCP/IP address set selected when the interface boots or is reset. A single pole switch enables the serial connection to a terminal program used to set the IP addresses, MAC address and output state invert. Serial connection is undesirable when the unit is active, and the serial connection should be disabled in deployed units.

Single-pole switches or fixed wiring enable status reports from the Arduino to a host controller. Connecting pin A4 to 0V enables recognition of OSC address **"/status"** which requires no parameters. When the message is received and the processing is enabled a UDP data packet

containing the status report string is sent to the IP address and port that issued the status request. The status message is a simple string in the format "ST01001010", where the "ST" is a simple ID header and the 8 digits are the logical state of the outputs with the leftmost digit showing output 1 state and the rightmost digit showing output 8 state. This reporting mode can be used as part of an "Are you there?" connection test process.

Connecting pin A5 to 0V enables the automatic status response mode in which every switch command received causes an output status report to the IP address and port that issued the switch command.

The Arduino sketch (program) can be used with UNO and NANO microcontroller boards, and can use either WizNet or ENC28J60 family ethernet interfaces. The customisation of the software is implemented through editing #define statements that control compiler operation.

The interface provides galvanic isolation by use of :-

1. A transformer built into the ethernet interface that protects the network infrastructure.
2. Opto-isolators in the input stage of a relay card protect the microcontroller from output equipment faults.
3. Relays providing opening or closing output signal connection. The relay outputs are typically connected to low-voltages on the external equipment input stages, but some relay modules *are* capable of mains load switching. Use appropriate installation mechanisms to minimise risk of mains voltage exposure to those configuring and maintaining the GPO unit.

The Arduino sketch (program) should be downloaded into the host hardware *before* the network card and relays are connected to the host.

The user must decide if they want to connect direct to the relay contacts, typically through a screw terminal block on the relay card, or provide connection to the relay outputs via standard connectors such as XLR or audio jacks.

Up to eight relay outputs are available per interface assembly when using a NANO or UNO microcontroller. Larger numbers of outputs can be accommodated by using a larger microcontroller such as the Arduino Mega. The program sketch needs editing to accommodate extra outputs.

## OSC Status Message Format

The OSC GPO can report the states of the outputs when the status message report is enabled (pin A4) and it receives the simple OSC message:

`/status`

No parameters are required, and if present they are ignored. The status reply is a string of the format "ST01100001" with the leftmost digit reporting the state of output 1. The message is sent in a UDP packet addressed to the IP address used to send the status request message. The target port number can either be the port that sent the status request, or a number set by the user during interface installation.

## Key Features

The GPO interface has a generalised design supporting a wide range of deployed operations.

- It uses off-the-shelf hardware for the microcontroller and ethernet interface. The design has been tested with Arduino UNO and Arduino NANO host microcontrollers, and with WizNet and ENC28J60 series network interface adapters.
- The GPO output interface normally uses opto-isolation between the microcontroller and the switched relay outputs, the isolator being a standard part of most relay boards.
- The software always supports 8 outputs from the microcontroller. The number of output relays can be any number between 1 and 8 depending on the relay interface card(s) fitted.
- The interface software is available under ISC open source licence. This allows end users to freely adapt the code base to support more outputs via a larger microcontroller such as the Arduino Mega.
- Customised networking properties for the module are stored in EEPROM within the Arduino. These properties are user configured via a serial port terminal connection to the interface system.
- The unit stores 4 sets of TCP/IP network address properties. This allows the interface unit to quickly set network properties when the module is moved between commonly used areas, each area having it's own network address range.
- There is a user programmed output invert property. This allows a logical 'ON' switch request to output a low voltage or a high voltage as required by the specific relay interface fitted. The invert setting is per output, enabling relay modules to be mixed as needed.
- The interface uses static outputs enabling control of rehearse/transmission style light outputs. Pulsed GPO operation is implemented by the OSC client sending time offset 'on' and 'off' messages.
- The interface is simple to power using a USB plug adapter or a brick PSU.

## Arduino Hardware Adaptation

The interface software is designed to work on both Arduino NANO and UNO microcontrollers but the software has to be slightly "customised" because the supported ethernet modules use different Arduino pins to select the interface module. This impacts of the pins that are available for the switched outputs. The customising mechanism allocates different pins as required by the ethernet module in use.

The ethernet interface modules for the UNO are (mostly) based on the WizNet 5100 or WizNet 5500 chip set and commonly include a micro-SD card holder physically mounted on the ethernet module. The SD card enable/disable is controlled by Arduino pin 4.

The NANO ethernet shields are mostly based on the ENC26J60 device family. This module does not include a micro SD card but does use Arduino pin 2 as a module select control.

The code customise mechanism uses a couple of #define statements, one defining the pin allocations that drive the output relay board, the other defining the ethernet support library required by the interface.

The pair of statements for microcontroller selection are :

```
// #define _NANO_HOST
// #define _UNO_HOST
```

The ethernet interface selection uses one of two conditional statements:

```
// #define _NIC_ENC28J60
// #define _NIC_WIZNET
```

Remove the leading `//` comment header to compile the code for the target hardware.

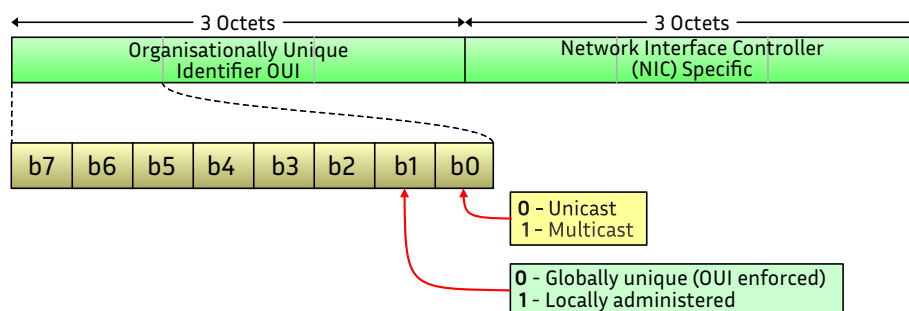
Many ethernet shields for the UNO include an SD card socket that can be read or written via the SPI connection to the microcontroller card. The ethernet shield uses pin D4 to control the SD card output. D4 must be high when the SD card is unused. D4 is therefore not available for GPO output. The common ENC28J60 shield for a NANO uses pin D2 for the ethernet card enable, making D2 unavailable.

The standard output pin allocations for typical UNO and NANO GPO boxes are shown in the table below:

GPO No	UNO Pin	NANO Pin
1	D2	D3
2	D3	D4
3	D5	D5
4	D6	D6
5	D7	D7
6	D8	D8
7	D9	D9
8	A3	A3

## Ethernet MAC Number

Every ethernet interface requires a 48-bit Media Access Control (MAC) number that is unique in the network segment on which the interface is used. This number may be globally allocated and stored in the interface firmware, or locally administered. The address type is signalled in the lowest two bits of the first address byte.



Modern practice operates the six octets as two groups each of three octets. The first three identify the manufacturer of the NIC, and the second three are a unique value allocated by the ethernet interface manufacturer.

The ethernet NIC in an Arduino installation requires a MAC address, but most hardware does not have a unique value allocated during manufacture. An exception is a modern ethernet UNO/MEGA shield from the official Arduino shop which have a sticker showing the globally allocated value for that interface. The value is not hard-coded into the interface and must be passed to the hardware as part of the application program `setup()` function operations. Most

example Arduino sketches show a default value of 0xDE 0xAD 0xBE 0xEF 0xFE 0xED (DEAD BEEF FEED).

This GPO interface does not require an edited Sketch with different MAC for each deployed system. Instead it uses the serial terminal capability within the Arduino to allow the MAC value to be entered and stored in EEPROM within the Arduino microcontroller. It is still important to use a unique MAC value with all the networks where the device is deployed.

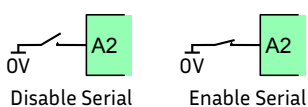
If the ethernet module has a sticker with a globally administered address use this value in the GPO configuration. Otherwise, use a MAC address that starts 0xDE (defined as unicast locally administered) plus 5 bytes chosen by the user.

Where multiple OSC GPO units may be deployed on a network keep a list of the addresses allocated to ensure unique allocation.

## Serial Link Control

The Arduino includes a USB connection used for downloading the Sketch (program) and transferring serial data between a host computer terminal program and the Arduino.

The OSC GPO application uses the serial link to allow the MAC address, relay output state invert and IP properties to be edited and stored in the internal EEPROM. In normal OSC GPO operation the serial link is unused and disabled.

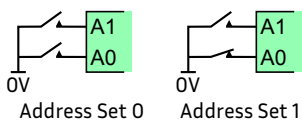


When the application boots it sets Arduino pin A2 (Analogue 2) to operate as a digital input with an internal input pull-up resistor. Once the mode has been set, the pin input level is tested. If the pin is open circuit it reports a HIGH input, and the serial link is not enabled. If the pin is connected to earth the input reports LOW during the boot process and the serial link is enabled. The serial properties are:

9600 baud, 8 data bits, no parity, single stop bit

The low level is only required during the unit boot because the pin is only read during the boot process, so a momentary action switch or shorting link can be used when the serial link is needed.

## IP Address Sets



The OSC GPO unit supports four sets of IP addresses stored in the internal EEPROM of the Arduino. The multiple address are useful where a unit is frequently moved between two or three networks. The active address set is controlled via the state of pins A1 and A0 which are read when the OSC GPO boots. The pins are set as inputs with internal pull-up resistors and operate with inverted sense, so with the two pins open circuit the unit uses address set 0.

Switch 1	Switch 0	Address Set
Open	Open	0
Open	Closed	1
Closed	Open	2
Closed	Closed	3

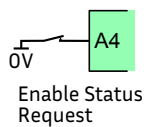
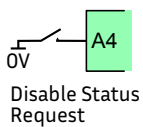
Each address set contains 5 elements:

Element No.	Property Name	Example
1	IP Address	192.168.42.190
2	Subnet Mask	255.255.255.0
3	Gateway address	192.168.42.1
4	DNS Server	192.168.42.1
5	Port Number	2000

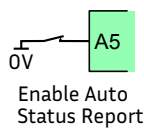
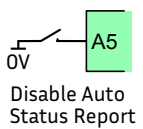
Although the application uses a static IP address and does not require packet routing it may need to set a non-default subnet mask, for example 255.255.240.0. The ethernet library call that sets the subnet mask also requires the gateway address and dns address passed to the function. Set the DNS and gateway to the first address in the subnet range.

For example if using an IP address of 10.206.84.191 and a subnet mask of 255.255.240.0 set the DNS and gateway address to 10.206.80.1

## Status Report Control



The state of Arduino pin A4 is read as part of the power-up or reset. If the pin is left open-circuit the software does not process the `/status` OSC message. Connect the pin to 0V to enable recognition of the message.



The state of Arduino pin A5 is read as part of the power-up or reset. If the pin is left open-circuit the software does not enable the auto-report mode. Connect the pin to 0V to enable the auto-status report where a status message is sent in response to each valid switch request.

It is possible to enable both status report modes when required.

## Entering Address Data

The MAC address, output invert flags and IP address sets are all edited then stored into EEPROM using the Arduino USB serial communication port. Hence the OSC GPO unit **must** be connected to a host computer running a terminal program. Example terminal programs include:

- Arduino Development software **Serial Monitor** window (Windows and Mac OS),
- **Putty** on a Windows OS host (download from <https://putty.org/>),
- **SerialTools** on MacOS (free from the Apple Application store).
- **Putty** for Mac OS (available from several web sites)

If the Arduino development environment is available, this is the simplest terminal to configure and use. If the Serial Monitor is not visible in the Arduino development software enable it using the Tools... Serial Monitor menu.

Whichever terminal software is used the data link properties must be set as 9600 baud, 8 data bits, no parity with one stop bit.

The OSC GPO serial input processing uses line feed as the end-of-line delimiter. The text parser ignores carriage return characters, hence the terminal program can be set to output line feed or carriage return plus line feed when the return key is pressed.

An advantage of the Arduino development tool is the **Tools Port:** menu lists the available serial ports. The Serial Monitor window shows if the currently selected port is communicating with an Arduino, providing a simple comms check.

Windows OS host computers normally have port names that start with **com**, for example **com5**. Mac OS host communications ports are all members of the /dev root folder, hence their name starts /dev. The full name is often a long string such as **/dev/cu.usbmodem14401**.

The edit process can only modify the *active* IP address set. Select the OSC GPO address set for edit using the switches or links on the unit. Reset or power the unit to activate the address set selection. Also ensure the serial link control is set to the position that enables serial communications (data pin A2 connected to 0V).

If the serial communications are enabled the terminal displays information about the current address set. The display will be similar to the example below.

```
Ethernet GPO Interface
Using IP Address set 0
Invert control 00000000
MAC DE:AC:AD:03:00:01
IP 192.168.42.201 port 2000
Mask 255.255.255.0
GW 192.168.42.1
DNS 192.168.42.1
Status reply port 5001
Status request disabled
Auto status reply enabled
```

Unused EEPROM cells have all bits at one, an octet value of FF hexadecimal. Hence a new Arduino board that has only had the operating program loaded, or an IP address set that has not yet been configured, shows the value FF in all octets of the address listings.

Check the host to OSC GPO connection by typing the command **show** and pressing the return key. The OSC GPO sends a report similar to the startup display, but with a different first line display, no indication which set number is being edited and no information about the status modes.

```
Edited Properties
Invert control 00000000
MAC DE:AC:AD:03:00:01
IP 192.168.42.201 port 2000
Mask 255.255.255.0
GW 192.168.42.1
DNS 192.168.42.1
Status reply port 2001
```

The same format of data display is output after each valid data entry. It is assumed the person editing the content is knowledgeable about TCP/IP address usage. Some input validation is applied, but the limited amount of memory in the Arduino means extensive checking is not possible.

If the entered information has detectably wrong syntax or an unusable value the current edit properties are not shown, just a single line that states **Error**.

The edited values are only stored in the EEPROM when the user inputs the **save** command, and the new values are activated when the OSC GPO is reset or re-powered.

All edit commands are entered using **lower case**, with a single space character between elements of the command. The formats of the valid edit commands are listed below.

**show**

Display the contents of the edit stores.

**set mac DE:AC:AD:01:00:01**

Defines the MAC number to be used by the ethernet interface. The validation tests check that the address is 17 characters long, all address bytes are hexadecimal, and there is a colon separator between each octet value. The MAC address is common to all address sets.

**set invert 00001001**

Defines the processing actions in the GPO output. The final element of the command must contain eight characters, all with a value of 0 or 1.

When a bit is set at 1 the associated output is inverted relative to the remote controlled set state. This invert capability supports relay boards that use active low interfaces remaining unpowered until an ON state is requested by the OSC client. The leftmost digit in the string controls GP08, the rightmost controls GP01.

**set ip 192.168.42.42**

Sets the IP address for the OSC GPO device to the address provided in the command. The input value is checked and an error reported if the value is not a valid IP4 address.

**set port 2001**

Sets the IP port number of the interface. OSC clients send packets to this port number. The provided value is checked as a valid unsigned number on the range 0 to 65535.

**set mask 255.255.255.0**

Sets the network mask.

**set dns 192.168.42.1**

Sets the address for the DNS service. Because the OSC GPO uses static addresses the DNS address is not used, but it should be set to a standard default value. Set the DNS address as the first allowed address within the address block defined by the network mask.

**set gw 192.168.42.1**

Set the gateway address used for packet routing. Usually set as the same address used for the DNS. Because the OSC GPO is a UDP receive only function it does not require a gateway routing function, but the ethernet library has to receive a value so that it can set the network mask property to a non-default value (one other than 25.255.255.0).

**set statusport 1234**

Set the port number that will receive status message reports. If this port number is set as 0, the messages are sent to the port number that issued the status request.

**save**

Saves the edited address set into the EEPROM store. The store mechanism checks the existing value in the EEPROM and only writes changed values. This minimises erase cycle wear in the hardware, although the EEPROM in the Arduino should support at least 100,000 erase cycles.

After the values are saved, **reboot** the OSC GPO unit and check the addresses reported at power up. Finally disable the serial port use and reset the OSC GPO unit.



# OSC Control Message Format

The OSC GPO supports simple OSC message format only. It does not recognise messages wrapped in a bundle.

The OSC address is:

`/gpiswitch/out`

The switch number and state of the switch are sent to the OSC end address using a string parameter. Valid switch numbers are 1 to 8. The structure of the string parameter supports changing the state of single or multiple switches in the same message.

Single switch control string: `"s2=1"`

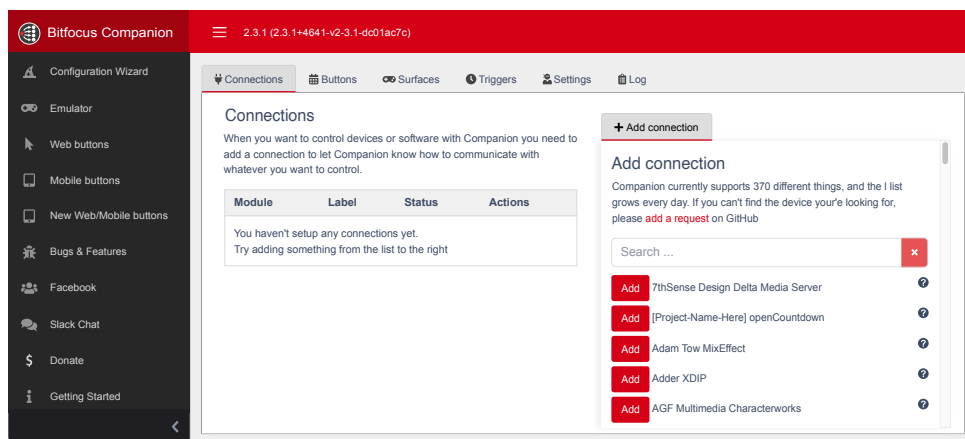
Multiple switches control string: `"s1=0 s2=1 s3=1 s4=0"`

The text at the left of the equals identifies the switch number, and the value to the right of the equal is the required switch state. State values of 0 and 1 are allowed. Multiple switch data uses a single space separator between the state definitions.

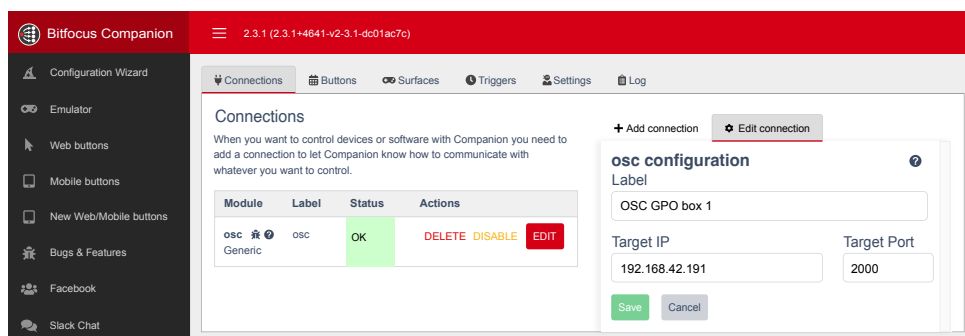
## Control OSC GPO outputs from BitFocus Companion

GPO control uses an instance of the generic OSC protocol available built into Companion. Assume there is an OSC GPO unit at address 192.168.42.191 using port 2000.

Open the Companion GUI, selecting the Connections tab.



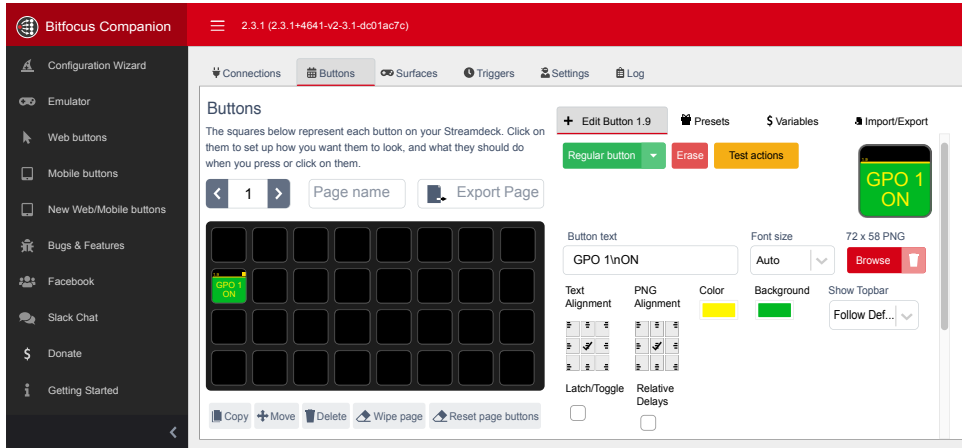
Type **generic osc** into the Add connection tab search box. A single entry is displayed. Click the Add button. The display changes to the **Edit Connection** tab.



Enter a recognisable identifier into the osc configuration **Label** field. This label will be used in the list of objects shown when editing a button control assignment.

Enter the IP address for the OSC GPO device and the port number, then click the Save button.

Switch to the Button editor. Select a button in the keypad view, and set the button type as **Regular button**. Add a name for the button function, setting colours as required.



**Press actions**

+ Add key down/on action Browse

Scroll the button edit form to show the actions entry section. Click on the Browse button to add an action to run when the Companion button is pressed.

**Browse Actions**

Search ...

Internal

OSC GPO box 1

Done

A form with a list of defined devices is shown, which includes an entry with the name defined for the OSC messages to the GPO unit.

Click on the **OSC GPO** text and the form updates to show the available actions.

**Browse Actions**

Search ...

internal

OSC GPO box 1

Send message without arguments	Add
Send integer	Add
Send float	Add
Send string	Add
Send message with multiple arguments	Add

Done

Click the **Send string line Add** button.

Click the Done button to stop adding actions to the button press list.

**Press actions**

OSC GPO box 1: Send string	OSC Path
Delay	/gpiswitch/out
0 ms	Value
	s1=1

Enter the OSC message address in the OSC Path field. Enter the switch id and state in the value field. The example at the left shows the string to set switch 1 ON. Note the switch prefix **MUST** be lower case "s".

If the OSC GPO box is connected to the network pressing the button in the Companion GUI activates output 1.

**Press actions**

OSC GPO box 1: Send string	OSC Path
Delay	/gpiswitch/out
0 ms	Value
	s1=1

OSC GPO box 1: Send string	OSC Path
Delay	/gpiswitch/out
100 ms	Value
	s1=0

When a pulse is required on an output add two Send string actions to the button press actions. Set the same OSC path in both items, but use state value of 1 for the first entry and a state value of 0 for the second entry.

Set the delay value for the second entry to the needed pulse duration.

Multiple output values can be set in a single message. The example at the left shows a command string that targets four switches. The switch definitions are separated by a space character. The switch definitions may be sent in any order, hence the strings "s1=1 s2=1 s3=0 s8=1" and "s2=1 s3=0 s1=1 s8=1" both set switches 1,2 and 8 ON with switch 3 off.

**Press actions**

OSC GPO box 1: Send string	OSC Path
Delay	/gpiswitch/out
0 ms	Value
	s1=1 s2=1 s3=0 s8=1

## Control OSC GPO outputs from SVT CasparCG Client

The SVT standard client for CasparCG can output OSC messages. Sending OSC messages requires one or more named destinations defined in the output section of the client configuration OSC tab. The significant area of this tab is shown by the red rectangle in the example screen below.

**Settings**

General Live Stream Servers Mixers GPI **OSC**

OSC Input

☒ Enable OSC Monitor

UDP Port: 6250

☒ Enable OSC Control

☒ Enable OSC Control through Web Socket

UDP Port: 3250 WebSocket Port: 4250

OSC Output

Name	Address	Port	Description
OSC GPO box 1	192.168.42.191	2000	Multi-Trigger

+ -

\* It is recommended you restart CasparCG Client for the changes to take effect

OK

Use the + button to add an output destination. The name entered in the first column is displayed in the destination selection field of the OSC output widget. The name is linked to an IP destination by the values in the Address and Port columns. The description column is optional.

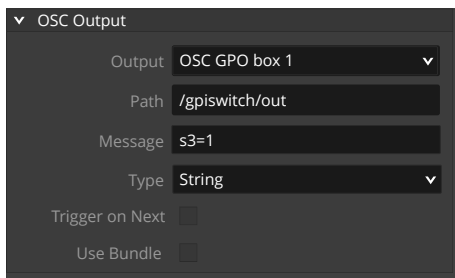
Add an OSC output control to the rundown. This control is available in the Tools... Other library listing, or using a mouse right-click and selecting the Tools... Other context menu.

Rundown 01

OSC Output

Delay: 0 UID:

Select the OSC output item in the rundown. Examine the Inspector display in the right-hand column of the client.



The OSC Output Inspector panel shows the following settings:

- Output: OSC GPO box 1
- Path: /gpiswitch/out
- Message: s3=1
- Type: String
- Trigger on Next: ☐
- Use Bundle: ☐

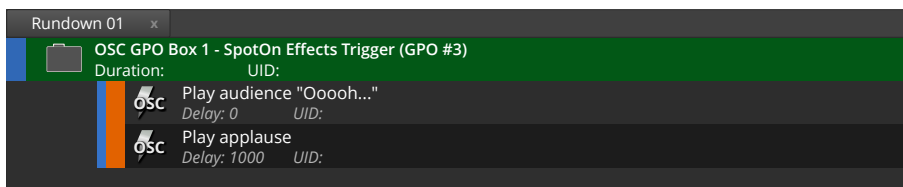
Select the message destination from the drop down list in the Output field. Enter the target address in the Path field.

Set the message type to String. The example at the left sets switch 3 ON.

Do **NOT** tick the Use Bundle option.

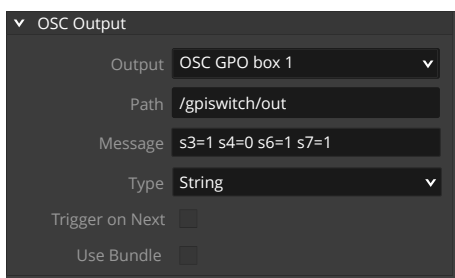
When a pulsed GPO output is required use two OSC output widgets combined into a group. Set the first output item output delay to zero, and the second item output delay to define the pulse duration.

Edit the Metadata Label fields to describe the function of each element. An example of such labelling is shown below.



The screenshot shows a rundown titled "Rundown 01" with the following elements:

- OSC GPO Box 1 - SpotOn Effects Trigger (GPO #3)
  - Duration: UID:
- OSC Play audience "Ooooh..."
  - Delay: 0 UID:
- OSC Play applause
  - Delay: 1000 UID:



The OSC Output Inspector panel shows the following settings:

- Output: OSC GPO box 1
- Path: /gpiswitch/out
- Message: s3=1 s4=0 s6=1 s7=1
- Type: String
- Trigger on Next: ☐
- Use Bundle: ☐

Multiple outputs can be switched in a single message as shown in the example at the left. Separate the switch element strings by a space character. The order of the switch numbers does not matter.

## Sources of Arduino and Relay Modules

The popularity of Arduino for many projects has produced a wide range of hardware suppliers. The open source nature of the Arduino board design means that there are also low-cost clones. Some suppliers offer screw terminal adapters for the Arduino, simplifying inter-module wiring.

### RS Components - [web link](#)

The RS stocked ranges include UNO and NANO microcontrollers manufactured under the official Arduino label. RS also sell the official 4-channel relay shield and the WizNet Ethernet shield for UNO and MEGA boards. The ethernet module can piggy-back onto the controller card, but the relay unit requires wiring because some relay control pins are used by the ethernet interface.

### CPC/Farnell - [web link](#)

CPC sell a wide range of Arduino and Arduino clone modules.

### Arduino Store via Amazon.co.uk - [web link](#)

The full range of Arduino microcontrollers and shields are available from this web store.

### AZDelivery via Amazon.co.uk - [web link](#)

This web store offers low-cost clones for many of the Arduino microcontrollers. They also have multi-unit bundles offering lower cost per unit than single purchase prices. The store sells terminal adapter boards (screwdriver connection module) for the NANO processor range, and an ENC28J60 series ethernet module for the NANO processors. Most microcontrollers are delivered with a USB A to Arduino USB cable.

AZDelivery also stock a range of relay interface modules, including one, two, four and eight relay per card modules. These require soldered connections to their controller board.

### Elegoo via Amazon.co.uk- [web link](#)

Sell Arduino clones and small relay cards.

## Appendix A - EEPROM Storage allocations

The memory address blocks used to store the configuration data are shown in the table below.

Start	End	Information Stored
0x00	0x05	MAC address
0x10	0x17	Output invert controls. 0=no invert, 1=invert.
0x20	0x33	User IP address set 0
0x40	0x53	User IP address set 1
0x60	0x73	User IP address set 2
0x80	0x93	User IP address set 3

With each User IP address set store block the relative offsets are:

0x00 IP address  
0x04 DNS address  
0x08 Subnet mask  
0x0C Gateway address  
0x10 IP port  
0x12 Status report target port number

Written by	Andy Woodhouse
Version	1.02
Date	5 <sup>th</sup> August 2023