

# **UNOFFICIAL X32/M32 OSC REMOTE PROTOCOL**

Initiated from version 1.01 (Oct-17-2012)  
version 0.0981 (12/27/15)

**An unofficial OSC protocol document for the  
X32/M32 Digital Mixing Console families**

## *Acknowledgements*

This document regroups data contained in version 1.01 of the OSC protocol for the X32 family of products released by Behringer in Oct. 2012, and a large number of additional OSC messages for communicating with the X32, their syntax and use, along with practical examples and explanations as to how and in which context they should be used. This document should also apply to M32, a product from Midas, very similar to X32.

Behringer is not associated to the redaction of this document and no support will be provided by the company.

The author has tried to make the information contained here as accurate as possible. A few areas are still prone to inaccuracies or uncertainties as to how to best use them. Please do not hesitate to provide feedback on the X32 user forum on errors or inaccuracies. They will be corrected in futures updates.

I also want to thank X32 forums well known **Paul Vannatto** for his invaluable support during the redaction, his generous time and his advice in reviewing the numerous versions I provided him prior to this release.

As you read through this document, you may like a “hands on” experience with OSC commands, it is recommend you use a utility to send/ read commands to/from the X32. Such utilities ensure the commands will be properly formatted and offer better support for reading parameters back from X32.

**X32\_Command**<sup>1</sup> is a terminal based utility running on Windows, Linux, OSX and Raspberry platforms, supporting batch and interactive modes, and multi-format/parameters. Download it from <https://sites.google.com/site/patrickmaillot/x32>

**X32 Live Toolbox**<sup>2</sup> is a GUI based utility running in Windows, Linux and OSX. It also offers additional features such as EQ copy. Download it from <http://sourceforge.net/projects/x32livetoolbox/>

Patrick-Gilles Maillot

---

<sup>1</sup> X32\_Command: © 2014-2015 Patrick-Gilles Maillot

<sup>2</sup> X32 Live Toolbox: © 2014-2015 Paul Vannatto

## Contents

UNOFFICIAL X32/M32 OSC REMOTE PROTOCOL .....	0
Contents .....	2
DESCRIPTION .....	5
Client initiated messages (client → X32 console).....	5
Server replies or server initiated messages (X32 console → client).....	7
Type rules (get/set parameter) and data formatting.....	8
Meter requests .....	10
List of all Meter IDs:.....	10
<i>meters/0</i> .....	10
<i>meters/1</i> .....	10
<i>meters/2</i> .....	11
<i>meters/3</i> .....	11
<i>meters/4</i> .....	11
<i>meters/5</i> <chn_meter_id> <grp_meter_id>.....	11
<i>meters/6</i> <channel_id>.....	11
<i>meters/7</i> .....	11
<i>meters/8</i> .....	12
<i>meters/9</i> .....	12
<i>meters/10</i> .....	12
<i>meters/11</i> .....	12
<i>meters/12</i> .....	12
<i>meters/13</i> .....	12
X32/M32 OSC PROTOCOL PARAMETERS.....	14
Server ↔ Client communications.....	14
Configuration (/config) data .....	14
Channel (/ch) data .....	18
Aux In (/auxin) data .....	20
FX Return (/fxrtn) data .....	21
Inserts (/insert) data.....	22
Bus (/bus) data .....	22
Matrix (/mtx) data .....	24
Main Stereo (/main/st) data.....	25
Main Mono (/main/m) data .....	26
DCA groups (/dca) data .....	27
Effects (/fx) data .....	28
Output sets (/output) data .....	29
Show, Cue, Scene, Snippet, and Preset Management .....	32
Notes on the use of /showdump.....	41
Preferences (/prefs) data .....	43
USB (/usb) data .....	46
Action (/action) data .....	46
Status (/stat) data.....	48

Node (/node) command.....	53
Renew (/renew) data.....	58
Effects Parameters .....	59
Hall Reverb .....	59
Plate Reverb .....	60
Ambiance Reverb .....	60
Rich Plate Reverb.....	61
Room Reverb .....	61
Chamber Reverb.....	62
Vintage Reverb .....	62
Vintage Room .....	63
Gated Reverb.....	63
Reverse Reverb.....	64
Stereo Delay .....	64
3-Tap Delay.....	65
4-Tap Delay.....	65
Stereo Chorus.....	66
Stereo Flanger .....	66
Stereo Phaser .....	67
Dimensional Chorus.....	67
Mood Filter .....	68
Rotary Speaker .....	68
Tremolo / Panner .....	69
Sub Octaver .....	69
Delay / Chamber.....	70
Delay / Chorus .....	70
Delay /Flanger .....	71
Chorus / Chamber .....	71
Flanger / Chamber.....	72
Modulation Delay.....	72
Dual Graphic Equalizer / True Dual Graphic Equalizer .....	73
Graphic Equalizer / True Graphic Equalizer.....	73
Stereo / Dual De-Esser.....	74
Precision Limiter .....	74
Stereo / Dual Program EQ .....	75
Stereo / Dual Midrange EQ.....	76
Stereo / Dual Combinator .....	77
Stereo / Dual Fair Compressor .....	79
Stereo / Dual Leisure Compressor.....	80
Edison EX1 .....	80
Stereo / Dual Ultimo Compressor .....	81
Sound Maxer .....	81
Stereo / Dual Enhancer.....	82
Stereo / Dual Exciter.....	83
Stereo Imager .....	83
Stereo / Dual Guitar Amp .....	84
Stereo / Dual Tube Stage.....	85
Stereo / Dual Pitch Shifter .....	86
Wave Designer.....	86

User Definable Controls (ASSIGN Section) .....	87
Rotary Encoders (X32/M32 Standard).....	88
Buttons (X32/M32 Standard) .....	90
Appendix – Converting X32 fader data to decibels and vice-versa .....	93
Appendix – Scene data elements .....	94
Appendix – Snippet data elements .....	96
Appendix – Channel, Library and Routing preset files data elements .....	97
Appendix – X32/M32 Icons .....	98
Appendix – OSC over MIDI Sysex commands.....	99
Appendix – Frequency Table – 201 log scale frequency values – [20 Hz, 20 kHz] .....	100
Appendix – Frequency Table – 121 log scale frequency values – [20 Hz, 20 kHz] .....	101
Appendix – Frequency Table – 101 log scale frequency values – [20 Hz, 400 Hz] .....	102
Appendix – Q Factor Table – 72 log scale Q values – [10.0, 0.3, 72].....	103
Appendix – Hold Table – 101 log scale Hold values – [0.02, 2000.00, 101] .....	104
Appendix – Release Table – 101 log scale Release values – [5.00, 4000.00, 101] .....	105
Appendix – Level Table – 161 pseudo-log scale Level values – [-∞, +10, 161] .....	106
Appendix – RTA Decay Table – 19 log scale Decay values – [0.25, 16, 19] .....	107
Appendix – Effects enums, names and preset names table .....	108
Appendix – Programming Examples.....	109
HelloX32 (Unix).....	109
X32 Connect, Send and Receive (Unix).....	110
X32Saver (Unix) .....	113
X32Saver (Windows) .....	116
X32 data echo in Go.....	119
Appendix – Miscellaneous .....	121
Floating point data representation .....	121
Notes Features or Messages to be validated or needing further testing .....	121

## DESCRIPTION

X32 & M32 are using a communication protocol that is compatible to standard OSC with some MUSIC Group specific extensions (e.g. parameter enquiry, subscriptions). OSC packets are received on **UDP port 10023** and replies are sent back to the requester's IP/port<sup>3</sup>.

In the following, the X32/M32 (rack, console) is also called server, and a connected device or application is typically called client. Connections to the server take place over Ethernet network, UDP port 10023. The server replies on the UDP port used by the client when establishing communication.

There are different modes of operation for the X32/M32 to communicate OSC protocol:

- **Immediate:** a client such as a network connected tablet or PC application sends a request with or without parameters, and the server immediately acts or replies with the respective data.  
*Note:* a single request from the client can result in several replies from the server (this is typically the case with `/showdump`)
- **Deferred:** a client such as a network connected tablet or PC application sends a specific request without parameters (`/xremote`). When changes take place either from the server UI or from a connected client, several update messages (or replies) are returned for a period of time, until a timeout is reached.  
*Note:* a single action at the server can result in several messages from the server

```
<OSC path>:= /config | /ch | /auxin | /fxrtn | /bus | /mtx | /main | /dca |
             /fx | /outputs | /headamp | /-show | /-libs | /-prefs | /-usb |
             /-action | /-stat | ...
```

### Client initiated messages (client → X32 console)

Operation	OSC address	Parameters	Comments
Info request	/info	None	Server responds with <code>/info</code> message
Status request	/status	None	Server responds with <code>/status</code> message
Control request	/xremote	None	Triggers server to send all parameter changes to maximum four active clients. Timeout is 10 seconds, e.g. the <code>/xremote</code> command has to be renewed before this delay in order to avoid losing information from The X32 console.
Set console parameter	<OSC path>	<string   int   float   blob value>	Sets the value of a console parameter, e.g.: <code>/ch/01/mix/fader~~~~,f~~&lt;float&gt;</code> If it exists and value is in range, the new value takes place in the X32.
Get console parameter	<OSC path>	None	Requests the value of a console parameter, e.g. <code>/ch/01/mix/fader~~~~</code> If it exists, the current value is echoed back by server, e.g.: <code>/ch/01/mix/fader~~~~,f~~&lt;float&gt;</code>

<sup>3</sup> See Appendix for an example of communication program

Get console parameter	/node	<string>	Requests the values of a set of console parameters, e.g.: <i>/node~~~,s~~-prefs/iQ/01~~~~</i> The current values for the full set corresponding to the request are returned by the server in plain text (string of characters, ending with a linefeed), e.g.: <i>node~~~,s~~/-prefs/iQ/01 none Linear 0\n~~~~</i>
Save console state	/save	<string><int><string><string><int>	Saves or updates the state or state changes of the X32 as a local scene or snippet at a given index, e.g.: <i>/save~~~,sissii~~~scene~~~[1]n1~~note~~~[0]</i>
Load state to console	/load	<string><int><int><int>	Loads from a local scene or snippet at a given index state or state changes to the X32 engine, e.g.: <i>/load~~~,siiii~~~scene~~~[1][0][0]</i>
Delete console state	/delete	<string><int>	deletes the state or state changes of the X32 (a local scene or snippet) at a given index, e.g.: <i>/delete~,si~scene~~~[1]</i>
Scene data request	/showdump	None	Triggers server to send all cue, scene and snippet data to requestor
Get console meter values	/meters	<string> <optional int: chn_meter_id> <optional int: grp_meter_id> <optional int: priority>	Results in regular updates meter values as a single binary blob. Timeout is 10 seconds, e.g. <i>/meters ,s meters/1</i> will return bursts of 96 float meter values (32 input, 32 gate and 32 dynamic gain reductions) for 10s. see "Meter requests" for additional details
Subscribe to data from server	/subscribe	<string> < optional int>	Client describes to X32 server what information it is interested in receiving, and at which frequency the update is reported, until a timeout of 10 seconds is reached.eg: <i>/subscribe ,s /-stat/solosw/01</i> or <i>/subscribe ,si /-stat/solosw/01 1</i> Will report about 200 updates of the state of solo switch for channel 01 over the span of 10s. <i>/subscribe ,si /-stat/solosw/01 50</i> Will report about 4 updates of the state of solo switch for channel 01
Subscribe to data formats from server	/formatsubscribe	<string>...<string><int>...<int>	Client describes to X32 server what information it is interested in receiving, e.g.: <i>/mfm_c/dca/*/on 1 8 8</i> Reports a blob of 36 bytes for about 10s. The last <int> specifies the frequency factor of the report.
Subscribe to batch data from server	/batchsubscribe	<string>...<string><int>...<int>	Client request from X32 server data to receive, e.g.: <i>/x_meters_0 /meters/0 0 69 1</i> Reports a blob of 70 floats for about 10s. <i>/x_meters_8 /meters/8 0 5 1</i> Reports a blob of 6 floats for about 10s. <i>/mfm_a /mix/on 0 63 8</i>

			Reports a blob of 276 bytes for about 10s. The last <int> specifies the frequency factor of the report.
Renew data request	/renew	<string>	Requests renewing of data described in <string>, e.g. /renew~,s~meters/5~~~~ /renew~,s~hidden/states~~~

### Server replies or server initiated messages (X32 console → client)

Operation	OSC address	Parameters	Comments
Info request	/info	<string server_version> <string server_name> <string console_model> <string console_version>	Returns names and version numbers, e.g. : /info~~,ssss~~V2.05~~osc-server~~X32C~~2.08~~~~ (~ stands for null character)
Status request	/status	<string state> <string IP_address > <string server_name >	Returns console status and IP , e.g. : /status~,sss~~~~active~~192.168.0.64~~~~osc-server~~ (~ stands for null character)
Console changes	/-stat	<string><int float>	If /xremote is active, the X32 console reports all changes made either via /-stat or <OSC path> operations, e.g. pressing on channel one solo button will generate (if no other solo button is engaged): /-stat/solosw/01~~~~,i~~[1] /-stat/solo~,i~~[1]
Set console parameter Get console parameter	<OSC path>	<string   int   float value>	If /xremote is active, the X32 console echoes the value of a console parameter in response to a get, set from client or X32 parameter change, e.g. /ch/01/mix/01/pan ,f <float>



## Type rules (get/set parameter) and data formatting

- all parameters must be big-endian and 4-byte aligned/padded, as per OSC specification.
- padding is done with null bytes.
- float parameters must be in range 0.0 – 1.0, e.g:
  - 0.0 → 0x00000000 (big-endian)
  - 0.5 → 0x3f000000 (big-endian)
  - 1.0 → 0x3f800000 (big-endian)
- integer and float parameters are signed 32-bit values.
- strings must be null-terminated.
- enum parameters can be sent as strings or integers (see below).
- boolean parameters will map to enum type {OFF, ON} (or OSC integer [0, 1])
- blobs (arbitrary binary data) follow specific rules depending on the section they apply to (see later in this document)

An OSC command typically consists in a 4-byte padded OSC message, followed by an optional 4-byte padded type tag string, and if a type tag string is present, one or more 4-byte aligned/padded arguments.

### Examples:

A simple OSC command, with no tag string and no arguments:

```
/info~~~
```

And the answer from different X32 systems (X32 FW and SW versions may vary):

```
X32 Standard: /info~~~,ssss~~~V2.05~~~osc-server~~X32~2.12~~~~
```

```
X32 Rack: /info~~~,ssss~~~V2.05~~~osc-server~~X32RACK~2.12~~~~
```

**TBV:**

```
X32 Compact: /info~~~,ssss~~~V2.05~~~osc-server~~X32COMPACT~2.12~~~~
```

```
X32 Producer: /info~~~,ssss~~~V2.05~~~osc-server~~X32PRODUCER~2.12~~~~
```

```
X32 Core: /info~~~,ssss~~~V2.05~~~osc-server~~X32CORE~2.12~~~~
```

An OSC command with a single tag string and argument:

```
/ch/01/config/name~~,s~~name~~~~
```

An OSC command with a more complex tag string and multiple arguments:

```
/ch/01/config~~~,siii~~~name~~~~[1][2][3]
```

This is equivalent to the following 4 simpler commands:

```
/ch/01/config/name~~,s~~name~~~~
```

```
/ch/01/config/icon~~,i~~[1]
```

```
/ch/01/config/color~,i~~[2]
```

```
/ch/01/config/source~~~~,i~~[3]
```

Or in hexadecimal for the last command:

```
2f63682f30312f636f6e6669672f736f75726365000000002c6900000000003
```

[1], [2], [3] above are 32bit, big-endian integers, or 0x00000001, 0x00000002, 0x00000003

### Responses from X32/M32:

Sending on port 10023 the UDP request « /info » to a standard X32 will be replied with 48 bytes back to the sender's UDP port:

```
/info~~~,ssss~~~V2.05~~~osc-server~~X32~2.10~~~~
```

Sending on port 10023 the UDP request « /status », will be replied with 52 bytes back to the sender's UDP port:

```
/status~,sss~~~~active~~192.168.0.64~~~~osc-server~~
```

Sending on port 10023 the UDP request «/fx/4/par/23~~~~», will be replied with 24 bytes back to the sender's UDP port, for example:

```
/fx/4/par/23~~~~,f~~[float 0.5]
```

or, in hexadecimal:

```
2f66782f342f7061722f3233000000002c6600003f000000
```

[Where ~ stands for null character (\0)]

### **Special considerations for enum type.**

As stated before, enums can be sent as strings or integer; for example the value of channel 01 gate mode is listed as an “enum” type with possible values of {EXP2, EXP3, EXP4, GATE, DUCK}.

The setting “GATE” can be enabled for channel 01 by sending either one of the following:

```
/ch/01/gate/mode~~~~,s~~GATE~~~~
```

or

```
/ch/01/gate/mode~~~~,i~~[3]
```

Or in hexadecimal:

```
2f63682f30312f676174652f6d6f6465000000002c7300004741544500000000
```

or

```
2f63682f30312f676174652f6d6f6465000000002c690000000000003
```

Please note this only applies to the “enum” type; for example it does not apply to the key source setting of dynamics which only accepts an “int” value between 0 and 64.

```
/ch/[01...32]/dyn/keysrc
```

[Where ~ stands for null character (\0)]

**Note:** The X32/M32 only considers a subset of discrete values of the floating point range [0.0, 1.0], depending on the destination the float value applies to; a number of steps determines the values “known” by X32/M32.

Example: In EQ frequencies, applicable values are listed as [20.0, 20k, 201], meaning the frequency range 20Hz to 20kHz is divided into 201 discrete values, and the same applies to the “known” floating points values in the range [0.0, 1.0] used to change or control EQ frequency). An OSC floating point value outside of the known values will be rounded to the nearest known values.

This is particularly useful to convert text to float values when X32/M32 returns data in the form of text, such as with the /node commands used in scene and snippets, or when having to send data as text, such as the case of OSC data sent over MIDI Sysex commands<sup>4</sup>.

Tables in appendix to this document list common cases for frequencies, levels, etc. following a log scale.

<sup>4</sup> See appendix for section on sending OSC commands over MIDI sysex messages

## Meter requests

The `/meters` OSC command is used for obtaining Meter data, or to get a specific set of meter values. Update cycle frequency for meter data is 50 ms, and may be variable according to console's ability to fulfill requests. Timeout is 10 seconds.

Meter values are returned as floats in the range 0.0 – 1.0, representing the linear audio level (digital 0 – full-scale; internal headroom allows for values up to 8.0 (+18 dBfs)).

The data returned by the X32/M32 server for `/meters` is a “blob”, or set of unformatted binary data. As a result, the format differs from what is typically returned by the X32/M32. This is essentially for efficiency/performance reasons. The format of a returned blob is as follows:

```
<meter id> ,b~~<int1><int2><nativefloat>...<nativefloat>
```

`<meter id>`: see possible values below (padded with null bytes)  
`,b~~`: indicates a blob format, padded with null bytes  
`<int1>`: the length of the blob in bytes, 32 bits big-endian coded  
`<int2>`: the number of `<nativefloats>`, 32 bits little-endian coded  
`<nativefloat>`: data or meter value(s), 32 bits floats, little-endian coded

### Example:

The following meter request is sent to an X32/M32 server:

```
/meters~,si~/meters/6~~~16
```

Where ~ stands for null character, and “16” is actually sent as a big-endian 32bit integer, i.e. 0x00000010.

```
2f6d6574657273002c7369002f6d65746572732f3600000000000010  
/ m e t e r s ~ , s i ~ / m e t e r s / 6 ~ ~ ~ [ 16]
```

The X32/M32 server will returns for approximately 10 seconds and approximately every 50ms the 4 channel strip meters (pre-fade, gate, dyn gain reduction and post-fade) values of channel 17, in a single blob, as shown in the reply message below:

```
2f6d65746572732f360000002c62000000000140400000fd1d2137fdff7f3f0000803f6ebbd534  
/ m e t e r s / 6 ~ ~ ~ , b ~ ~ [ i n t 1 ] [ i n t 2 ] [ n f l o a t ] [ n f l o a t ] [ n f l o a t ] [ n f l o a t ]
```

### List of all Meter IDs:

`/meters/0`

Returns meter values from the **METERS** page (not used for X32-Edit):

32 input channels

8 aux returns

4x2 st fx returns

16 bus masters

6 matrixes

→ returns 70 float values as single binary blob

`/meters/1`

Returns meter values from the **METERS/channel** page:

32 input channels

32 gate gain reductions  
32 dynamics gain reductions  
→ returns 96 float values as single binary blob

***/meters/2***

Returns meter values from the **METERS/mix bus** page:  
16 bus masters  
6 matrixes  
2 main LR  
1 mono M/C  
16 bus master dynamics gain reductions  
6 matrix dynamics gain reductions  
1 main LR dynamics gain reduction  
1 mono M/C dynamics gain reduction  
→ returns 49 float values as single binary blob

***/meters/3***

Returns meter values from the **METERS/aux/fx** page:  
6 aux sends  
8 aux returns  
4x2 st fx returns  
→ returns 22 float values as single binary blob

***/meters/4***

Returns meter values from the **METERS/in/out** page:  
32 input channels  
8 aux returns  
16 outputs  
16 P16 ultranet outputs  
6 aux sends  
2 digital AES/EBU out  
2 monitor outputs  
→ returns 82 float values as single binary blob

***/meters/5 <chn\_meter\_id> <grp\_meter\_id>***

Returns meter values the **Console Surface VU Meters** (channel, group and main meters):  
16 channel meters: <chn\_meter\_id> 0: channel 1-16; 1: channel 17-32; 2: aux/fx returns;  
3: bus masters  
8 group meters: <grp\_meter\_id> 1 : mix bus 1 -8; 2: mix bus 9-16; 3: matrixes  
2 main LR  
1 mono M/C  
→ returns 27 float values as single binary blob

***/meters/6 <channel\_id>***

Returns meter values from **Channel Strip Meters** (pre-fade, gate, dyn gain reduction and post-fade):  
4 channel strip meters: <channel\_id> channel 0...71]  
→ returns 4 float values as single binary blob

***/meters/7***

Returns meter values from the **Bus Send** meters:  
16 bus send meters

→ returns 16 float values (from Bus sends 1-16) as single binary blob

**/meters/8**

Returns meter values from **Matrix Send** meters:

6 Matrix send meters

→ returns 6 float values (from Matrix sends 1-6) as single binary blob

**/meters/9**

Returns meter values from **Effect Send** and **Return** meters:

2 effects send and 2 effects return meters for each FX slot (8 slots)

→ returns 32 float values (4 x FX1, 4 x FX2, ... 4 x FX8) as single binary blob

**/meters/10**

Used for some **Effects**, for example Dual DeEsser, Stereo DeEsser, Stereo Fair Compressor

→ returns 32 float values

**/meters/11**

Returns meter values from the **Monitor pages**

→ returns 5 float values (Mon Left, Mon Right, Talk A/B level, Threshold/GR, Osc Tone level) as single binary blob

**/meters/12**

Returns meter values from the **Recorder page**

→ returns 4 float values (ReInput L, ReInput R, Playback L, Playback R) as single binary blob

**/meters/13**

Details TBD

→ returns 48 float values

**/meters/14**

Used for some **Effects**, for example Precision Limiter, Combinator, Stereo Fair Compressor

→ returns 80 float values

**/meters/15**

Used for **RTA** and some **Effects**, for example Dual GEQ, Stereo GEQ

→ returns 50 32bits values as single binary blob.

The 32bits values returned are representing 100 successive *little endian coded short ints*, in the range [0x8000, 0x0000]; each short int value provides a floating point RTA db level in the range [-128.0, 0.0], by dividing the short int (converted to float) by 256.0.

For example a 32bit value of 008000c0 will represent two values, the first one being 0x8000 (or -128.0 after conversion), and the second one being 0xc000 (or -64.0 after conversion). Similarly, a 32bits value of 40e0ffff will represent two successive RTA values of -31.75db and -0.004db, respectively.

Note: a short int value of 0x0000 (or 0.0db) means signal clipping occurred.

The 100 short ints (or RTA db values) correspond to frequencies listed in the table (values in Hz) below, respectively.

20	21	22	24	26	28	30	32	34	36
39	42	45	48	52	55	59	63	68	73
78	84	90	96	103	110	118	127	136	146

156	167	179	192	206	221	237	254	272	292
313	335	359	385	412	442	474	508	544	583
625	670	718	769	825	884	947	1.02K	1.09K	1.17K
1.25K	1.34K	1.44K	1.54K	1.65K	1.77K	1.89K	2.03K	2.18K	2.33K
2.50K	2.68K	2.87K	3.08K	3.30K	3.54K	3.79K	4.06K	4.35K	4.67K
5.00K	5.36K	5.74K	6.16K	6.60K	7.07K	7.58K	8.12K	8.71K	9.33K
10.00K	10.72K	11.49K	12.31K	13.20K	14.14K	15.16K	16.25K	17.41K	18.66K

## X32/M32 OSC PROTOCOL PARAMETERS

types →		[string, enum(integer), int(integer), linf(float), logf(float), level(float), bitmap(integer)] All data is on 4 bytes or multiples of 4 bytes
range →	string	A string of characters with up to [max. characters], padded to a multiple of 4 with with \0 (null) characters
	enum	An int corresponding to an element in a [list of all possible strings]
	int	An int with value in [min. value, max. value], step size = 1
	linf	A float with value in [min. value, max. value, step size], following a linear scale
	logf	A float with value in [min. value, max. value, steps], following a log scale
	level	A float with value in [0.0...1.0 (+10 dB), steps]: 4 'linear' dB ranges: 0.0...0.0625 (-∞, -90...-60 dB), 0.0625...0.25 (-60...-30 dB), 0.25...0.5 (-30...-10dB) and 0.5...1.0 (-10...+10dB) <i>(see conversion help in appendix)</i>
	%int	An int corresponding to the bitwise OR of multiple bits (0 or 1)

### Server ↔ Client communications

The following tables describe communication messages that can be initiated by the client, or by the server as a response to the client or as update data.

#### Configuration (/config) data

<path>	<type>	<range>	<unit>
<b>config data</b>			
/config/chlink/1-2	enum	{OFF, ON}, int with value 0 or 1 indicating whether channels 1 and 2 are linked or not	
/config/chlink/3-4	enum	{OFF, ON}	
/config/chlink/5-6	enum	{OFF, ON}	
/config/chlink/7-8	enum	{OFF, ON}	
/config/chlink/9-10	enum	{OFF, ON}	
/config/chlink/11-12	enum	{OFF, ON}	
/config/chlink/13-14	enum	{OFF, ON}	
/config/chlink/15-16	enum	{OFF, ON}	
/config/chlink/17-18	enum	{OFF, ON}	
/config/chlink/19-20	enum	{OFF, ON}	
/config/chlink/21-22	enum	{OFF, ON}	

/config/chlink/23-24	enum	{OFF, ON}	
/config/chlink/25-26	enum	{OFF, ON}	
/config/chlink/27-28	enum	{OFF, ON}	
/config/chlink/29-30	enum	{OFF, ON}	
/config/chlink/31-32	enum	{OFF, ON}	
/config/auxlink/1-2	enum	{OFF, ON}	
/config/auxlink/3-4	enum	{OFF, ON}	
/config/auxlink/5-6	enum	{OFF, ON}	
/config/auxlink/7-8	enum	{OFF, ON}	
/config/fxlink/1-2	enum	{OFF, ON}	
/config/fxlink/3-4	enum	{OFF, ON}	
/config/fxlink/5-6	enum	{OFF, ON}	
/config/fxlink/7-8	enum	{OFF, ON}	
/config/buslink/1-2	enum	{OFF, ON}	
/config/buslink/3-4	enum	{OFF, ON}	
/config/buslink/5-6	enum	{OFF, ON}	
/config/buslink/7-8	enum	{OFF, ON}	
/config/buslink/9-10	enum	{OFF, ON}	
/config/buslink/11-12	enum	{OFF, ON}	
/config/buslink/13-14	enum	{OFF, ON}	
/config/buslink/15-16	enum	{OFF, ON}	
/config/mtxlink/1-2	enum	{OFF, ON}	
/config/mtxlink/3-4	enum	{OFF, ON}	
/config/mtxlink/5-6	enum	{OFF, ON}	
/config/mute/[1...6]	enum	{OFF, ON}: Mute Group selection	
/config/linkcfg/hadly	enum	{OFF, ON}: Sets Delay + HA link	
/config/linkcfg/eq	enum	{OFF, ON}: Sets EQ link	
/config/linkcfg/dyn	enum	{OFF, ON}: Sets Dynamics link	
/config/linkcfg/fdrmute	enum	{OFF, ON}: Sets Mute/Fader link	
/config/mono/mode	enum	int with value 0 or 1 representing {LR+M, LCR}	
/config/mono/link	enum	{OFF, ON} Sets M/C Depends on Main L/R	
/config/solo/level	level	[0.0...1.0 (+10 dB), 161]	dB
/config/solo/source	enum	Int [0...6] representing {OFF, LR, LR+C, LR PFL, LR AFL, AUX 56, AUX 78}	
/config/solo/sourcetrिम	linf	[-18.000, 18.000, 0.500]	dB
/config/solo/chmode	enum	int with value 0 or 1 representing {PFL, AFL}	
/config/solo/busmode	enum	{PFL, AFL}	
/config/solo/dcamode	enum	{PFL, AFL}	
/config/solo/exclusive	enum	{OFF, ON}	
/config/solo/followsel	enum	{OFF, ON}	
/config/solo/followsolo	enum	{OFF, ON}	



/config/solo/dimatt	linf	[-40.000, 0.000, 1.000]	dB
/config/solo/dim	enum	{OFF, ON}	
/config/solo/mono	enum	{OFF, ON}	
/config/solo/delay	enum	{OFF, ON}	
/config/solo/delaytime	linf	[0.300, 500.000, 0.100]	ms
/config/solo/masterctrl	enum	{OFF, ON}	
/config/solo/mute	enum	{OFF, ON}	
/config/solo/dimpfl	enum	{OFF, ON}	
/config/talk/enable	enum	{OFF, ON}	
/config/talk/source	enum	int with value 0 or 1 representing {INT, EXT}	
/config/talk/A/level	level	[0.0...1.0 (+10 dB), 161] <sup>5</sup>	dB
/config/talk/A/dim	enum	{OFF, ON}	
/config/talk/A/latch	enum	{OFF, ON}	
/config/talk/A/destmap	%int	[0, 262143] (bitmap)	
/config/talk/B/level	level	[0.0...1.0 (+10 dB), 161]	dB
/config/talk/B/dim	enum	{OFF, ON}	
/config/talk/B/latch	enum	{OFF, ON}	
/config/talk/B/destmap	%int	[0, 262143] (bitmap)	
/config/osc/level	level	[0.0...1.0 (+10 dB), 161]	dB
/config/osc/f1	logf	[20.000, 20000, 121] <sup>6</sup>	Hz
/config/osc/f2	logf	[20.000, 20000, 121]	Hz
/config/osc/fsel	enum	int with value 0 or 1 representing {F1, F2}	
/config/osc/type	enum	int with value [0...2] representing {SINE, PINK, WHITE}	
/config/osc/dest	int	int with value [0...25] representing {MixBus1...16, L, R, L+R, M/C, Matrix1...6}	
/config/routing/IN/1-8	enum	int with value [0...19] representing	
/config/routing/IN/9-16		{AN1-8, AN9-16, AN17-24, AN25-32,	
/config/routing/IN/17-24		A1-8, A9-16, A17-24, A25-32, A33-40, A41-48,	
/config/routing/IN/25-32		B1-8, B9-16, B17-24, B25-32, B33-40, B41-48, CARD1-8,	
		CARD9-16, CARD17-24, CARD25-32}	
/config/routing/IN/AUX	enum	int with value [0...12] representing	
		{AUX1-4, AN1-2, AN1-4, AN1-6, A1-2, A1-4 A1-6,	
		B1-2, B1-4, B1-6, CARD1-2, CARD1-4, CARD1-6}	
/config/routing/AES50A/1-8	enum	int with value [0...25] representing	
/config/routing/AES50A/9-16		{AN1-8, AN9-16, AN17-24, AN25-32,	
/config/routing/AES50A/17-24		A1-8, A9-16, A17-24, A25-32, A33-40, A41-48,	
/config/routing/AES50A/25-32		B1-8, B9-16, B17-24, B25-32, B33-40, B41-48,	
/config/routing/AES50A/33-40		CARD1-8, CARD9-16, CARD17-24, CARD25-32,	
/config/routing/AES50A/41-48		OUT1-8, OUT9-16, P161-8, P16 9-16, AUX1-6/Mon,	
		AuxIN1-6/TB}	
/config/routing/AES50B/1-8			
/config/routing/AES50B/9-16			
/config/routing/AES50B/17-24			

<sup>5</sup> See Appendix section for detailed values

<sup>6</sup> See Appendix section for detailed values

/config/routing/AES50B/25-32			
/config/routing/AES50B/33-40			
/config/routing/AES50B/41-48			
/config/routing/CARD/1-8			
/config/routing/CARD/9-16			
/config/routing/CARD/17-24			
/config/routing/CARD/25-32			
/config/routing/OUT/1-4	enum	int with value [0...25] representing	
/config/routing/OUT/9-12		{AN1-4, AN9-12, AN17-20, AN25-28, A1-4, A9-12, A17-20, A25-28, A33-36, A41-44, B1-4, B9-12, B17-20, B25-28, B33-46, B41-44, CARD1-4, CARD9-12, CARD17-20, CARD25-28, OUT1-4, OUT9-12, P161-4, P169-12, AUX/CR, AUX/TB}	
/config/routing/OUT/5-8	enum	int with value [0...25] representing	
/config/routing/OUT/13-16		{AN5-8, AN13-16, AN21-24, AN29-32, A5-8, A13-16, A21-24, A29-32, A37-40, A45-48, B5-8, B13-16, B21-24, B29-32, B37-40, B45-48, CARD5-8, CARD13-16, CARD21-24, CARD29-32, OUT5-8, OUT13-16, P165-8, P1613-16, AUX/CR, AUX/TB}	
/config/userctrl/A/color	enum	int with value [0...15] representing	
/config/userctrl/B/color		{OFF, RD, GN, YE, BL, MG, CY, WH, OFFi, RDi, GNi, YEi, BLi, MGi, CYi, WHi}	
/config/userctrl/C/color			
/config/userctrl/A/enc/1...4	string	String up to 7 characters representing encoder assignment and unction.	
/config/userctrl/B/enc/1...4		See User Control Chapter for full details.	
/config/userctrl/C/enc/1...4			
/config/userctrl/A/btn/5...12	string	User assignable set A, B, or C: Button 5 to 12	
/config/userctrl/B/btn/5...12		See User Control Chapter for full details.	
/config/userctrl/C/btn/5...12			
/config/tape/gainL	linf	[-6.000, 24.000, 0.500]	dB
/config/tape/gainR	linf	[-6.000, 24.000, 0.500]	dB
/config/tape/autoplay	enum	{OFF, ON}	
		USB recorder play mode: single or folder	

## Channel (/ch) data

channel [01...32] (channel id 0...31)			
/ch/[01...32]/config/name	string	[12]	
/ch/[01...32]/config/icon	int	[1...74] (see appendix for a list of icons)	
/ch/[01...32]/config/color	enum	int with value [0...15] representing {OFF, RD, GN, YE, BL, MG, CY, WH, OFFi, RDi, GNi, YEi, BLi, MGi, CYi, WHi}	
/ch/[01...32]/config/source	int	int with value [0...64] representing {OFF, In01...32, Aux 1...6, USB L, USB R, Fx 1L...Fx 4R, Bus 01...16}	
/ch/[01...32]/delay/on	enum	{OFF, ON}	
/ch/[01...32]/delay/time	linf	[0.300, 500.000, 0.100]	ms
/ch/[01...32]/preamp/trim	linf	[-18.000, 18.000, 0.250] (digital sources only)	dB
/ch/[01...32]/preamp/invert	enum	{OFF, ON}	
/ch/[01...32]/preamp/hpon	enum	{OFF, ON}	
		Sets Phantom power off or on	
/ch/[01...32]/preamp/hpslope	enum	{12, 18, 24}	
/ch/[01...32]/preamp/hpf	logf	[20.000, 400.000, 101] <sup>7</sup>	Hz
/ch/[01...32]/gate/on	enum	{OFF, ON}	
/ch/[01...32]/gate/mode	enum	int [0...4] representing {EXP2, EXP3, EXP4, GATE, DUCK}	
/ch/[01...32]/gate/thr	linf	[-80.000, 0.000, 0.500]	dB
/ch/[01...32]/gate/range	linf	[3.000, 60.000, 1.000]	dB
/ch/[01...32]/gate/attack	linf	[0.000, 120.000, 1.000]	ms
/ch/[01...32]/gate/hold	logf	[0.020, 2000, 101] <sup>8</sup>	ms
/ch/[01...32]/gate/release	logf	[5.000, 4000.000, 101] <sup>9</sup>	ms
/ch/[01...32]/gate/keysrc	int	int with value [0...64] representing {OFF, In01...32, Aux 1...6, USB L, USB R, Fx 1L...Fx 4R, Bus 01...16}	
/ch/[01...32]/gate/filter/on	enum	{OFF, ON}	
/ch/[01...32]/gate/filter/type	enum	int with value [0...8] representing Keysolo (Solo/Q) {LC6, LC12, HC6, HC12, 1.0, 2.0, 3.0, 5.0, 10.0}	
/ch/[01...32]/gate/filter/f	Logf	[20.000, 20000, 201] <sup>10</sup>	Hz
/ch/[01...32]/dyn/on	enum	{OFF, ON}	
/ch/[01...32]/dyn/mode	enum	{COMP, EXP}	
/ch/[01...32]/dyn/det	enum	{PEAK, RMS}	
/ch/[01...32]/dyn/env	enum	{LIN, LOG}	
/ch/[01...32]/dyn/thr	linf	[-60.000, 0.000, 0.500]	dB
/ch/[01...32]/dyn/ratio	enum	int with value [0...11] representing {1.1, 1.3, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0, 7.0, 10, 20, 100}	
/ch/[01...32]/dyn/knee	linf	[0.000, 5.000, 1.000]	

<sup>7</sup> See Appendix section for detailed values

<sup>8</sup> See Appendix section for detailed values

<sup>9</sup> See Appendix section for detailed values

<sup>10</sup> See Appendix section for detailed values

/ch/[01...32]/dyn/mgain	linf	[0.000, 24.000, 0.500]	dB
/ch/[01...32]/dyn/attack	linf	[0.000, 120.000, 1.000]	ms
/ch/[01...32]/dyn/hold	logf	[0.020, 2000, 101]	ms
/ch/[01...32]/dyn/release	logf	[5.000, 4000.000, 101]	ms
/ch/[01...32]/dyn/pos	enum	{PRE, POST}	
/ch/[01...32]/dyn/keysrc	int	int with value [0...64] representing {OFF, In01...32, Aux 1...6, USB L, USB R, Fx 1L...Fx 4R, Bus 01...16}	
/ch/[01...32]/dyn/mix	linf	[0, 100, 5]	%
/ch/[01...32]/dyn/auto <sup>11</sup>	enum	{OFF, ON}	
/ch/[01...32]/dyn/filter/on	enum	{OFF, ON}	
/ch/[01...32]/dyn/filter/type	enum	int with value [0...8] representing Keysolo (Solo/Q) {LC6, LC12, HC6, HC12, 1.0, 2.0, 3.0, 5.0, 10.0}	
/ch/[01...32]/dyn/filter/f	logf	[20.000, 20000, 201]	Hz
/ch/[01...32]/insert/on	enum	{OFF, ON}	
/ch/[01...32]/insert/pos	enum	{PRE, POST}	
/ch/[01...32]/insert/sel	enum	int with value [0...22] representing {OFF, FX1L, FX1R, FX2L, FX2R, FX3L, FX3R, FX4L, FX4R, FX5L, FX5R, FX6L, FX6R, FX7L, FX7R, FX8L, FX8R, AUX1, AUX2, AUX3, AUX4, AUX5, AUX6}	
/ch/[01...32]/eq/on	enum	{OFF, ON}	
/ch/[01...32]/eq/[1...4]/type	enum	int [0...5] representing {LCut, LShv, PEQ, VEQ, HShv, HCut}	
/ch/[01...32]/eq/[1...4]/f	logf	[20.000, 20000, 201]	Hz
/ch/[01...32]/eq/[1...4]/g	linf	[-15.000, 15.000, 0.250]	dB
/ch/[01...32]/eq/[1...4]/q	logf	[10.000, 0.3, 72]	
/ch/[01...32]/mix/on	enum	{OFF, ON}	
/ch/[01...32]/mix/fader	level	[0.0...1.0(+10dB), 1024]	dB
/ch/[01...32]/mix/st	enum	{OFF, ON}	
/ch/[01...32]/mix/pan	linf	[-100.000, 100.000, 2.000]	
/ch/[01...32]/mix/mono	enum	{OFF, ON}	
/ch/[01...32]/mix/mlevel	level	[0.0...1.0 (+10 dB), 161]	dB
/ch/[01...32]/mix/[01...16]/on	enum	{OFF, ON}	
/ch/[01...32]/mix/[01...16]/level	level	[0.0...1.0 (+10 dB), 161]	dB
/ch/[01...32]/mix/01/pan	linf	[-100.000, 100.000, 2.000]	
/ch/[01...32]/mix/01/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST, GRP}	
/ch/[01...32]/mix/03/pan	linf	[-100.000, 100.000, 2.000]	
/ch/[01...32]/mix/03/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST, GRP}	
/ch/[01...32]/mix/05/pan	linf	[-100.000, 100.000, 2.000]	
/ch/[01...32]/mix/05/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST, GRP}	
/ch/[01...32]/mix/07/pan	linf	[-100.000, 100.000, 2.000]	
/ch/[01...32]/mix/07/type	enum	int [0...5] representing	

<sup>11</sup> This command is available starting with FW 2.10

/ch/[01...32]/mix/09/pan	linf	{IN/LC, <-EQ, EQ->, PRE, POST, GRP}
/ch/[01...32]/mix/09/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST, GRP}
/ch/[01...32]/mix/11/pan	linf	[-100.000, 100.000, 2.000]
/ch/[01...32]/mix/11/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST, GRP}
/ch/[01...32]/mix/13/pan	linf	[-100.000, 100.000, 2.000]
/ch/[01...32]/mix/13/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST, GRP}
/ch/[01...32]/mix/15/pan	linf	[-100.000, 100.000, 2.000]
/ch/[01...32]/mix/15/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST, GRP}
/ch/[01...32]/grp/dca	%int	[0, 255] (bitmap)
/ch/[01...32]/grp/mute	%int	[0, 63] (bitmap)

#### Aux In (/auxin) data

auxin [01...08] (channel id 32...39)

/auxin/[01...08]/config/name	string	[12]	
/auxin/[01...08]/config/icon	int	[1...74] (see appendix for a list of icons)	
/auxin/[01...08]/config/color	enum	int with value [0...15] representing {OFF, RD, GN, YE, BL, MG, CY, WH, OFFi, RDi, GNi, YEi, BLi, MGi, CYi, WHi}	
/auxin/[01...08]/config/source	int	int with value [0...64] representing {OFF, In01...32, Aux 1...6, USB L, USB R, Fx 1L...Fx4R, Bus 01...16}	
/auxin/[01...08]/preamp/trim	linf	[-18.000, 18.000, 0.250]	dB
/auxin/[01...08]/preamp/invert	enum	{OFF, ON}	
/auxin/[01...08]/eq/on	enum	{OFF, ON}	
/auxin/[01...08]/eq/[1...4]/type	enum	int [0...5] representing {LCut, LShv, PEQ, VEQ, HShv, HCut}	
/auxin/[01...08]/eq/[1...4]/f	logf	[20.000, 20000, 201]	Hz/dB
/auxin/[01...08]/eq/[1...4]/g	linf	[-15.000, 15.000, 0.250]	
/auxin/[01...08]/eq/[1...4]/q	logf	[10.000, 0.3, 72]	
/auxin/[01...08]/mix/on	enum	{OFF, ON}	
/auxin/[01...08]/mix/fader	level	[0.0...1.0(+10dB), 1024]	
/auxin/[01...08]/mix/st	enum	{OFF, ON}	
/auxin/[01...08]/mix/pan	linf	[-100.000, 100.000, 2.000]	
/auxin/[01...08]/mix/mono	enum	{OFF, ON}	
/auxin/[01...08]/mix/mlevel	level	[0.0...1.0 (+10 dB), 161]	
/auxin/[01...08]/mix/[01...16]/on	enum	{OFF, ON}	
/auxin/[01...08]/mix/[01...16]/level	level	[0.0...1.0 (+10 dB), 161]	
/auxin/[01...08]/mix/01/pan	linf	[-100.000, 100.000, 2.000]	
/auxin/[01...08]/mix/01/type	enum	int [0...5] representing	

			{IN/LC, <-EQ, EQ->, PRE, POST, GRP}
/auxin/[01...08]/mix/03/pan	linf		[-100.000, 100.000, 2.000]
/auxin/[01...08]/mix/03/type	enum	int [0...5] representing	{IN/LC, <-EQ, EQ->, PRE, POST, GRP}
/auxin/[01...08]/mix/05/pan	linf		[-100.000, 100.000, 2.000]
/auxin/[01...08]/mix/05/type	enum	int [0...5] representing	{IN/LC, <-EQ, EQ->, PRE, POST, GRP}
/auxin/[01...08]/mix/07/pan	linf		[-100.000, 100.000, 2.000]
/auxin/[01...08]/mix/07/type	enum	int [0...5] representing	{IN/LC, <-EQ, EQ->, PRE, POST, GRP}
/auxin/[01...08]/mix/09/pan	linf		[-100.000, 100.000, 2.000]
/auxin/[01...08]/mix/09/type	enum	int [0...5] representing	{IN/LC, <-EQ, EQ->, PRE, POST, GRP}
/auxin/[01...08]/mix/11/pan	linf		[-100.000, 100.000, 2.000]
/auxin/[01...08]/mix/11/type	enum	int [0...5] representing	{IN/LC, <-EQ, EQ->, PRE, POST, GRP}
/auxin/[01...08]/mix/13/pan	linf		[-100.000, 100.000, 2.000]
/auxin/[01...08]/mix/13/type	enum	int [0...5] representing	{IN/LC, <-EQ, EQ->, PRE, POST, GRP}
/auxin/[01...08]/mix/15/pan	linf		[-100.000, 100.000, 2.000]
/auxin/[01...08]/mix/15/type	enum	int [0...5] representing	{IN/LC, <-EQ, EQ->, PRE, POST, GRP}
/auxin/[01...08]/grp/dca	%int		[0, 255] (bitmap)
/auxin/[01...08]/grp/mute	%int		[0, 63] (bitmap)

### FX Return (/fxrtn) data

#### fxrtn [01...08] (channel id 40...47)

/fxrtn/[01...08]/config/name	string	[12]	
/fxrtn/[01...08]/config/icon	int	[1...74] (see appendix for a list of icons)	
/fxrtn/[01...08]/config/color	enum	int with value [0...15] representing	{OFF, RD, GN, YE, BL, MG, CY, WH, OFFi, RDi, GNi, YEi, BLi, MGi, CYi, WHi}
/fxrtn/[01...08]/eq/on	enum	{OFF, ON}	
/fxrtn/[01...08]/eq/[1...4]/type	enum	int [0...5] representing	{LCut, LShv, PEQ, VEQ, HShv, HCut}
/fxrtn/[01...08]/eq/[1...4]/f	logf	[20.000, 20000, 201]	Hz
/fxrtn/[01...08]/eq/[1...4]/g	linf	[-15.000, 15.000, 0.250]	dB
/fxrtn/[01...08]/eq/[1...4]/q	logf	[10.000, 0.3, 72]	
/fxrtn/[01...08]/mix/on	enum	{OFF, ON}	
/fxrtn/[01...08]/mix/fader	level	[0.0...1.0(+10dB), 1024]	dB
/fxrtn/[01...08]/mix/st	enum	{OFF, ON}	
/fxrtn/[01...08]/mix/pan	linf	[-100.000, 100.000, 2.000]	dB
/fxrtn/[01...08]/mix/mono	enum	{OFF, ON}	
/fxrtn/[01...08]/mix/mlevel	level	[0.0...1.0 (+10 dB), 161]	dB
/fxrtn/[01...08]/mix/[01...16]/on	enum	{OFF, ON}	

/fxrtn/[01...08]/mix/[01...16]/level	level	[0.0...1.0 (+10 dB), 161]	dB
/fxrtn/[01...08]/mix/01/pan	linf	[-100.000, 100.000, 2.000]	
/fxrtn/[01...08]/mix/01/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST, GRP}	
/fxrtn/[01...08]/mix/03/pan	linf	[-100.000, 100.000, 2.000]	
/fxrtn/[01...08]/mix/03/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST, GRP}	
/fxrtn/[01...08]/mix/05/pan	linf	[-100.000, 100.000, 2.000]	
/fxrtn/[01...08]/mix/05/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST, GRP}	
/fxrtn/[01...08]/mix/07/pan	linf	[-100.000, 100.000, 2.000]	
/fxrtn/[01...08]/mix/07/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST, GRP}	
/fxrtn/[01...08]/mix/09/pan	linf	[-100.000, 100.000, 2.000]	
/fxrtn/[01...08]/mix/09/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST, GRP}	
/fxrtn/[01...08]/mix/11/pan	linf	[-100.000, 100.000, 2.000]	
/fxrtn/[01...08]/mix/11/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST, GRP}	
/fxrtn/[01...08]/mix/13/pan	linf	[-100.000, 100.000, 2.000]	
/fxrtn/[01...08]/mix/13/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST, GRP}	
/fxrtn/[01...08]/mix/15/pan	linf	[-100.000, 100.000, 2.000]	
/fxrtn/[01...08]/mix/15/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST, GRP}	
/fxrtn/[01...08]/grp/dca	%int	[0, 255] (bitmap)	
/fxrtn/[01...08]/grp/mute	%int	[0, 63] (bitmap)	

### Inserts (/insert) data

#### insert

/insert/aux[1...6]	int	Channel the aux input [1...6] is inserted into
--------------------	-----	--

### Bus (/bus) data

#### bus [01...16] (channel id 48...63)

/bus/[01...16]/config/name	string	[12]	
/bus/[01...16]/config/icon	int	[1...74] (see appendix for a list of icons)	
/bus/[01...16]/config/color	enum	int with value [0...15] representing {OFF, RD, GN, YE, BL, MG, CY, WH, OFFi, RDi, GNi, YEi, BLi, MGi, CYi, WHi}	
/bus/[01...16]/dyn/on	enum	{OFF, ON}	
/bus/[01...16]/dyn/mode	enum	{COMP, EXP}	
/bus/[01...16]/dyn/det	enum	{PEAK, RMS}	
/bus/[01...16]/dyn/env	enum	{LIN, LOG}	
/bus/[01...16]/dyn/thr	linf	[-60.000, 0.000, 0.500]	dB

/bus/[01...16]/dyn/ratio	enum	int with value [0...11] representing {1.1, 1.3, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0, 7.0, 10, 20, 100}	
/bus/[01...16]/dyn/knee	linf	[0.000, 5.000, 1.000]	
/bus/[01...16]/dyn/mgain	linf	[0.000, 24.000, 0.500]	dB
/bus/[01...16]/dyn/attack	linf	[0.000, 120.000, 1.000]	ms
/bus/[01...16]/dyn/hold	logf	[0.020, 2000, 101]	ms
/bus/[01...16]/dyn/release	logf	[5.000, 4000.000, 101]	ms
/bus/[01...16]/dyn/pos	enum	{PRE, POST}	
/bus/[01...16]/dyn/keysrc	int	int with value [0...64] representing {OFF, In01...32, Aux 1...6, USB L, USB R, Fx 1L...Fx 4R, Bus 01...16}	
/bus/[01...16]/dyn/mix	linf	[0, 100, 5]	%
/bus/[01...16]/dyn/auto <sup>12</sup>	enum	{OFF, ON}	
/bus/[01...16]/dyn/filter/on	enum	{OFF, ON}	
/bus/[01...16]/dyn/filter/type	enum	int with value [0...8] representing Keysolo (Solo/Q) {LC6, LC12, HC6, HC12, 1.0, 2.0, 3.0, 5.0, 10.0}	
/bus/[01...16]/dyn/filter/f	logf	[20.000, 20000, 201]	Hz
/bus/[01...16]/insert/on	enum	{OFF, ON}	
/bus/[01...16]/insert/pos	enum	{PRE, POST}	
/bus/[01...16]/insert/sel	enum	int with value [0...22] representing {OFF, FX1L, FX1R, FX2L, FX2R, FX3L, FX3R, FX4L, FX4R, FX5L, FX5R, FX6L, FX6R, FX7L, FX7R, FX8L, FX8R, AUX1, AUX2, AUX3, AUX4, AUX5, AUX6}	
/bus/[01...16]/eq/on	enum	{OFF, ON}	
/bus/[01...16]/eq/[1...6]/type	enum	int [0...5] representing {LCut, LShv, PEQ, VEQ, HShv, HCut}	
/bus/[01...16]/eq/[1...6]/f	logf	[20.000, 20000, 201]	Hz
/bus/[01...16]/eq/[1...6]/g	linf	[-15.000, 15.000, 0.250]	dB
/bus/[01...16]/eq/[1...6]/q	logf	[10.000, 0.3, 72]	
/bus/[01...16]/mix/on	enum	{OFF, ON}	
/bus/[01...16]/mix/fader	level	[0.0...1.0(+10dB), 1024]	dB
/bus/[01...16]/mix/st	enum	{OFF, ON}	
/bus/[01...16]/mix/pan	linf	[-100.000, 100.000, 2.000]	
/bus/[01...16]/mix/mono	enum	{OFF, ON}	
/bus/[01...16]/mix/mlevel	level	[0.0...1.0(+10dB), 161]	dB
/bus/[01...16]/mix/[01...06]/on	enum	{OFF, ON}	
/bus/[01...16]/mix/[01...06]/level	level	[0.0...1.0(+10dB), 161]	dB
/bus/[01...16]/mix/01/pan	linf	[-100.000, 100.000, 2.000]	
/bus/[01...16]/mix/03/pan	linf	[-100.000, 100.000, 2.000]	
/bus/[01...16]/mix/05/pan	linf	[-100.000, 100.000, 2.000]	
/bus/[01...16]/mix/01/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST }	
/bus/[01...16]/mix/03/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST }	
/bus/[01...16]/mix/05/type	enum	int [0...5] representing	

<sup>12</sup> This command is available starting with FW 2.10



{IN/LC, <-EQ, EQ->, PRE, POST }

/bus/[01...16]/grp/dca %int [0, 255] (bitmap)  
/bus/[01...16]/grp/mute %int [0, 63] (bitmap)

### Matrix (/mtx) data

mtx [01...06] (channel id 64...69)

/mtx/[01...06]/config/name	string	[12]	
/mtx/[01...06]/config/icon	int	[1...74] (see appendix for a list of icons)	
/mtx/[01...06]/config/color	enum	int with value [0...15] representing {OFF, RD, GN, YE, BL, MG, CY, WH, OFFi, RDi, GNi, YEi, BLi, MGi, CYi, WHi}	
/mtx/[01...06]/config/preamp/invert	enum	{OFF, ON}	
/mtx/[01...06]/dyn/on	enum	{OFF, ON}	
/mtx/[01...06]/dyn/mode	enum	{COMP, EXP}	
/mtx/[01...06]/dyn/det	enum	{PEAK, RMS}	
/mtx/[01...06]/dyn/env	enum	{UN, LOG}	
/mtx/[01...06]/dyn/thr	linf	[-60.000, 0.000, 0.500]	dB
/mtx/[01...06]/dyn/ratio	enum	int with value [0...11] representing {1.1, 1.3, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0, 7.0, 10, 20, 100}	
/mtx/[01...06]/dyn/knee	linf	[0.000, 5.000, 1.000]	
/mtx/[01...06]/dyn/mgain	linf	[0.000, 24.000, 0.500]	dB
/mtx/[01...06]/dyn/attack	linf	[0.000, 120.000, 1.000]	Ms
/mtx/[01...06]/dyn/hold	logf	[0.020, 2000, 101]	Ms
/mtx/[01...06]/dyn/release	logf	[5.000, 4000.000, 101]	Ms
/mtx/[01...06]/dyn/pos	enum	{PRE, POST}	
/mtx/[01...06]/dyn/mix	linf	[0, 100, 5]	%
/mtx/[01...06]/dyn/auto <sup>13</sup>	enum	{OFF, ON}	
/mtx/[01...06]/dyn/filter/on	enum	{OFF, ON}	
/mtx/[01...06]/dyn/filter/type	enum	int with value [0...8] representing Keysolo (Solo/Q) {LC6, LC12, HC6, HC12, 1.0, 2.0, 3.0, 5.0, 10.0}	
/mtx/[01...06]/dyn/filter/f	logf	[20.000, 20000, 201]	Hz
/mtx/[01...06]/insert/on	enum	{OFF, ON}	
/mtx/[01...06]/insert/pos	enum	{PRE, POST}	
/mtx/[01...06]/insert/sel	enum	int with value [0...22] representing {OFF, FX1L, FX1R, FX2L, FX2R, FX3L, FX3R, FX4L, FX4R, FX5L, FX5R, FX6L, FX6R, FX7L, FX7R, FX8L, FX8R, AUX1, AUX2, AUX3, AUX4, AUX5, AUX6}	
/mtx/[01...06]/eq/on	enum	{OFF, ON}	
/mtx/[01...06]/eq/[1...6]/type	enum	int [0...5] representing {LCut, LShv, PEQ, VEQ, HShv, HCut} For /mtx/01/ and mtx/06/ type extends to int [0...13] adding {BU6, BU12, BS12, LR12, BU18, BU24, BS24, LR24}. In that case /mtx/02/ and /mtx/05/ are	

<sup>13</sup> This command is available starting with FW 2.10

		ignored, repectively.	
/mtx/[01...06]/eq/[1...6]/f	logf	[20.000, 20000, 201]	Hz
/mtx/[01...06]/eq/[1...6]/g	linf	[-15.000, 15.000, 0.250]	dB
/mtx/[01...06]/eq/[1...6]/q	logf	[10.000, 0.3, 72]	
/mtx/[01...06]/mix/on	enum	{OFF, ON}	
/mtx/[01...06]/mix/fader	level	[0.0...1.0(+10dB), 1024]	dB

### Main Stereo (/main/st) data

#### main stereo (channel id 70)

/main/st/config/name	string	[12]	
/main/st/config/icon	int	[1...74] (see appendix for a list of icons)	
/main/st/config/color	enum	int with value [0...15] representing {OFF, RD, GN, YE, BL, MG, CY, WH, OFFi, RDi, GNi, YEi, BLi, MGi, CYi, WHi}	
/main/st/dyn/on	enum	{OFF, ON}	
/main/st/dyn/mode	enum	{COMP, EXP}	
/main/st/dyn/det	enum	{PEAK, RMS}	
/main/st/dyn/env	enum	{LIN, LOG}	
/main/st/dyn/thr	linf	[-60.000, 0.000, 0.500]	dB
/main/st/dyn/ratio	enum	int with value [0...11] representing {1.1, 1.3, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0, 7.0, 10, 20, 100}	
/main/st/dyn/knee	linf	[0.000, 5.000, 1.000]	
/main/st/dyn/mgain	linf	[0.000, 24.000, 0.500]	dB
/main/st/dyn/attack	linf	[0.000, 120.000, 1.000]	ms
/main/st/dyn/hold	logf	[0.020, 2000, 101]	ms
/main/st/dyn/release	logf	[5.000, 4000.000, 101]	ms
/main/st/dyn/pos	enum	{PRE, POST}	
/main/st/dyn/mix	linf	[0, 100, 5]	%
/main/st/dyn/auto <sup>14</sup>	enum	{OFF, ON}	
/main/st/dyn/filter/on	enum	{OFF, ON}	
/main/st/dyn/filter/type	enum	int with value [0...8] representing Keysolo (Solo/Q) {LC6, LC12, HC6, HC12, 1.0, 2.0, 3.0, 5.0, 10.0}	
/main/st/dyn/filter/f	logf	[20.000, 20000, 201]	Hz
/main/st/insert/on	enum	{OFF, ON}	
/main/st/insert/pos	enum	{PRE, POST}	
/main/st/insert/sel	enum	int with value [0...22] representing {OFF, FX1L, FX1R, FX2L, FX2R, FX3L, FX3R, FX4L, FX4R, FX5L, FX5R, FX6L, FX6R, FX7L, FX7R, FX8L, FX8R, AUX1, AUX2, AUX3, AUX4, AUX5, AUX6}	
/main/st/eq/on	enum	{OFF, ON}	
/main/st/eq/[1...6]/type	enum	int [0...5] representing {LCut, LShv, PEQ, VEQ, HShv, HCut}	

<sup>14</sup> This command is available starting with FW 2.10

/main/st/eq/[1...6]/f	logf	[20.000, 20000, 201]	Hz
/main/st/eq/[1...6]/g	linf	[-15.000, 15.000, 0.250]	dB
/main/st/eq/[1...6]/q	logf	[10.000, 0.3, 72]	
/main/st/mix/on	enum	{OFF, ON}	
/main/st/mix/fader	level	[0.0...1.0(+10dB), 1024]	dB
/main/st/mix/pan	linf	[-100.000, 100.000, 2.000]	
/main/st/mix/[01...06]/on	enum	{OFF, ON}	
/main/st/mix/[01...06]/level	level	[0.0...1.0(+10dB), 161]	dB
/main/st/mix/01/pan	linf	[-100.000, 100.000, 2.000]	
/main/st/mix/03/pan	linf	[-100.000, 100.000, 2.000]	
/main/st/mix/05/pan	linf	[-100.000, 100.000, 2.000]	
/main/st/mix/01/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST }	
/main/st/mix/03/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST }	
/main/st/mix/05/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST }	

### Main Mono (/main/m) data

#### main mono (channel id 71)

/main/m /config/name	string	[12]	
/main/m /config/icon	Int	[1...74] (see appendix for a list of icons)	
/main/m /config/color	enum	int with value [0...15] representing {OFF, RD, GN, YE, BL, MG, CY, WH, OFFi, RDi, GNi, YEi, BLi, MGi, CYi, WHi}	
/main/m /dyn/on	enum	{OFF, ON}	
/main/m /dyn/mode	enum	{COMP, EXP}	
/main/m /dyn/det	enum	{PEAK, RMS}	
/main/m /dyn/env	enum	{LIN, LOG}	
/main/m /dyn/thr	linf	[-60.000, 0.000, 0.500]	dB
/main/m /dyn/ratio	enum	int with value [0...11] representing {1.1, 1.3, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0, 7.0, 10, 20, 100}	
/main/m /dyn/knee	linf	[0.000, 5.000, 1.000]	
/main/m /dyn/mgain	linf	[0.000, 24.000, 0.500]	dB
/main/m /dyn/attack	linf	[0.000, 120.000, 1.000]	Ms
/main/m /dyn/hold	logf	[0.020, 2000, 101]	Ms
/main/m /dyn/release	logf	[5.000, 4000.000, 101]	Ms
/main/m /dyn/pos	enum	{PRE, POST}	
/main/m /dyn/mix	linf	[0, 100, 5]	%
/main/m /dyn/auto <sup>15</sup>	enum	{OFF, ON}	
/main/m /dyn/filter/on	enum	{OFF, ON}	
/main/m /dyn/filter/type	enum	int with value [0, 8] representing Keysolo (Solo/Q) {LC6, LC12, HC6, HC12, 1.0, 2.0, 3.0, 5.0, 10.0}	
/main/m /dyn/filter/f	logf	[20.000, 20000, 201]	Hz

<sup>15</sup> This command is available starting with FW 2.10

/main/m/insert/on	enum	{OFF, ON}	
/main/m/insert/pos	enum	{PRE, POST}	
/main/m/insert/sel	enum	int with value [0...22] representing {OFF, FX1L, FX1R, FX2L, FX2R, FX3L, FX3R, FX4L, FX4R, FX5L, FX5R, FX6L, FX6R, FX7L, FX7R, FX8L, FX8R, AUX1, AUX2, AUX3, AUX4, AUX5, AUX6}	
/main/m/eq/on	enum	{OFF, ON}	
/main/m/eq/[1...6]/type	enum	int [0...5] representing {LCut, LShv, PEQ, VEQ, HShv, HCut}	
/main/m/eq/[1...6]/f	logf	[20.000, 20000, 201]	Hz
/main/m/eq/[1...6]/g	linf	[-15.000, 15.000, 0.250]	dB
/main/m/eq/[1...6]/q	logf	[10.000, 0.3, 72]	
/main/m/mix/on	enum	{OFF, ON}	
/main/m/mix/fader	level	[0.0...1.0(+10dB), 1024]	dB
/main/m/mix/[01...06]/on	enum	{OFF, ON}	
/main/m/mix/[01...06]/level	level	[0.0...1.0(+10dB), 161]	dB
/main/m/mix/01/pan	linf	[-100.000, 100.000, 2.000]	
/main/m/mix/03/pan	linf	[-100.000, 100.000, 2.000]	
/main/m/mix/05/pan	linf	[-100.000, 100.000, 2.000]	
/main/m/mix/01/type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST }	
/main/m/mix/03/ type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST }	
/main/m/mix/05/ type	enum	int [0...5] representing {IN/LC, <-EQ, EQ->, PRE, POST }	

### DCA groups (/dca) data

dca groups (no channel id!)

/dca/[1...8]/on	enum	{OFF, ON}	
/dca/[1...8]/fader	level	[0.0...1.0(+10dB), 1024]	dB
/dca/[1...8]/config/name	string	[12]	
/dca/[1...8]/config/icon	Int	[1...74] (see appendix for a list of icons)	
/dca/[1...8]/config/color	enum	int with value [0...15] representing {OFF, RD, GN, YE, BL, MG, CY, WH, OFFi, RDi, GNi, YEi, BLi, MGi, CYi, WHi}	

## Effects (/fx) data

### effects fx [1...4]

/fx/[1...4]/type	enum	int [0...60] representing {HALL, AMBI, RPLT, ROOM, CHAM, PLAT, VREV, VRM, GATE, RVRS, DLY, 3TAP, 4TAP, CRS, FLNG, PHAS, DIMC, FILT, ROTA, PAN, SUB, D/RV, CR/R, FL/R, D/CR, D/FL, MODD, GEQ2, GEQ, TEQ2, TEQ, DES2, DES, P1A, P1A2, PQ5, PQ5S, WAVD, LIM, CMB, CMB2, FAC, FAC1M, FAC2, LEC, LEC2, ULC, ULC2, ENH2, ENH, EXC2, EXC, IMG, EDI, SON, AMP2, AMP, DRV2, DRV, PIT2, PIT} <sup>16</sup>
/fx/[1...4]/source/l	enum	int with value [0...17] representing {INS, MIX1, MIX2, MIX3, MIX4, MIX5, MIX6, MIX7, MIX8, MIX9, MIX10, MIX11, MIX12, MIX13, MIX14, MIX15, MIX16, M/C}
/fx/[1...4]/source/r	enum	int with value [0...17] representing {INS, MIX1, MIX2, MIX3, MIX4, MIX5, MIX6, MIX7, MIX8, MIX9, MIX10, MIX11, MIX12, MIX13, MIX14, MIX15, MIX16, M/C}
/fx/[1...4]/par/[01...64]	linf/logf	Up to 64 parameters, depending on selected effect type. See Effect Parameters Chapter

### Effects fx[5...8]

/fx/[5...8]/type	enum	int [0...33] representing {GEQ2, GEQ, TEQ2, TEQ, DES2, DES, P1A, P1A2, PQ5, PQ5S, WAVD, LIM, FAC, FAC1M, FAC2, LEC, LEC2, ULC, ULC2, ENH2, ENH, EXC2, EXC, IMG, EDI, SON, AMP2, AMP, DRV2, DRV, PHAS, FILT, PAN, SUB} <sup>17</sup>
/fx/[5...8]/par/[01...64]	linf/logf	Up to 64 parameters, depending on selected effect type. See Effect Parameters Chapter

<sup>16</sup> See Appendix for table of enum/name/type

<sup>17</sup> See Appendix for table of enum/name/type

## Output sets (/output) data

### outputs main [01...16]

/main/[01...16]/src	int	int value [0...76] representing {OFF, Main L, Main R, M/C, MixBus 01...16, Matrix 1...6, DirectOut Ch 01...32, DirectOut Aux 1...8, DirectOut FX 1L...4R, Monitor L, Monitor R, Talkback}
/outputs/main/[01...16]/pos	enum	int [0...8] representing {IN/LC, IN/LC+M, <-EQ, <-EQ+M, EQ->, EQ->+M, PRE, PRE+M, POST}
/outputs/main/[01...16]/delay/on	enum	{OFF, ON}
/outputs/main/[01...16]/delay/time	linf	[0.300, 500.000, 0.100] ms

### outputs aux [01...06]

/outputs/aux/[01...06]/src	Int	int value [0...76] representing {OFF, Main L, Main R, M/C, MixBus 01...16, Matrix 1...6, DirectOut Ch 01...32, DirectOut Aux 1...8, DirectOut FX 1L...4R, Monitor L, Monitor R, Talkback}
/outputs/aux/[01...06]/pos	enum	int [0...8] representing {IN/LC, IN/LC+M, <-EQ, <-EQ+M, EQ->, EQ->+M, PRE, PRE+M, POST}

### outputs P16 [01...16]

/outputs/p16/[01...16]/src	Int	int value [0...76] representing {OFF, Main L, Main R, M/C, MixBus 01...16, Matrix 1...6, DirectOut Ch 01...32, DirectOut Aux 1...8, DirectOut FX 1L...4R, Monitor L, Monitor R, Talkback}
/outputs/p16/[01...16]/pos	enum	int [0...8] representing {IN/LC, IN/LC+M, <-EQ, <-EQ+M, EQ->, EQ->+M, PRE, PRE+M, POST}
/outputs/p16/[01...16]/iQ/group <sup>18</sup>	enum	Int [0...3] representing the group the iQ speaker is associated to, in the range {OFF, A, B}  <i>0: OFF</i> <i>1: A</i> <i>2: B</i>
/outputs/p16/[01...16]/iQ/speaker <sup>19</sup>	enum	int [0...6] representing the type of Turbosound iQ speakers connected to the output, in the range {none, iQ8, iQ10, iQ12, iQ15, iQ15B, iQ18B}  <i>0: none</i> <i>1: iQ8</i> <i>2: iQ10</i> <i>3: iQ12</i> <i>4: iQ15</i> <i>5: iQ15B</i> <i>6: iQ18B</i>

<sup>18</sup> This command is available starting with FW 2.12

<sup>19</sup> This command is available starting with FW 2.12

<code>/outputs/p16/[01...16]/iQ/eq<sup>20</sup></code>	enum	int [0...4] representing a frequency response setting for the respective speaker. Possible values are: {Linear, Live, Speech, Playback, User}
		<i>0: Linear (default setting)</i> <i>1: Live (typical live sound setting)</i> <i>2: Speech (optimal speech intelligibility setting)</i> <i>3: Playback (ideal setting for music playback)</i> <i>4: User (response curve set in the iQ speaker sub-menu)</i>
<code>/outputs/p16/[01...16]/iQ/model<sup>21</sup></code>	int	An integer representing a sound Model, either a Turbosound signature voicing or a DSP model of an industry standard product. The value is within a range depending on the type of speaker modeling set for the respective speaker:
		<i>iQ8 : [0...5]: iQ8, E8, F8+, UPJunior, PS8, NuQ8-DP</i> <i>iQ10: [0...4]: iQ10, F10+, UPJ-1P, PS10-R2, NuQ10-DP</i> <i>iQ12: [0...7]: iQ12, E12, JF29NT, ELX112P, PRX612M, F12+, UPA-1P, NuQ12-DP</i> <i>iQ15: [0...7]: iQ15, JF59NT, ELX115P, PRX615M, F15+, UPQ-1P, PS15-R2, NuQ15-DP</i> <i>iQ15B: [0...3]: iQ15B, E15X, S15+, B-15DP</i> <i>iQ18B: [0...4]: iQ18B, ELX18P, PRX6118S, S18+, B-18DP</i>

#### outputs AES [01...2]

<code>/outputs/aes/[01...02]/src</code>	int	int value [0...76] representing {OFF, Main L, Main R, M/C, MixBus 01...16, Matrix 1...6, DirectOut Ch 01...32, DirectOut Aux 1...8, DirectOut FX 1L...4R, Monitor L, Monitor R, Talkback}
<code>/outputs/aes/[01...02]/pos</code>	enum	int [0...8] representing {IN/LC, IN/LC+M, <-EQ, <-EQ+M, EQ->, EQ->+M, PRE, PRE+M, POST}

#### outputs REC [01...02]

<code>/outputs/rec/[01...02]/src</code>	int	int value [0...76] representing: {OFF, Main L, Main R, M/C, MixBus 01...16, Matrix 1...6, DirectOut Ch 01...32, DirectOut Aux 1...8, DirectOut FX 1L...4R, Monitor L, Monitor R, Talkback}
<code>/outputs/rec/[01...02]/pos</code>	enum	int [0...8] representing {IN/LC, IN/LC+M, <-EQ, <-EQ+M, EQ->, EQ->+M, PRE, PRE+M, POST}

#### headamp [000...127]

<code>/headamp/[000...127]/gain</code>	linf	[-12.000, 60.000, 0.500]	dB
<code>/headamp/[000...127]/phantom</code>	enum	{OFF, ON}	
		<i>/headamp index: 000...031: local XLR inputs</i>	

<sup>20</sup> This command is available starting with FW 2.12

<sup>21</sup> This command is available starting with FW 2.12

*032...079: AES50 port A connected devices*  
*080...127: AES50 port B connected devices*



## Show, Cue, Scene, Snippet, and Preset Management

This section covers the `/showdump`, `/-show`, `/-libs`, `/add`, `/copy`, `/save`, `/load`, `/delete`, `/rename`, and `/-undo` commands typically used to manage Show, Cues, Scenes, Snippets and Presets.

### Shows, Cues, Scenes, Snippets

The X32/M32 family of products is capable of handling a single show at a time.

A show is made of a list of Cues, referencing Scenes and Snippets. A show can contain up to 100 distinct cues. Cue numbering consists of 3 numbers in the form xxx.x to offer a hierarchy scheme. Cues can also have a flag to skip them at read/execute time.

X32/M32 systems can manage 100 different Scenes and 100 different Snippets, each numbered 0 to 99. Scenes files consist in a large collection of data resulting from and with a similar format as the output of `/node` commands. Snippets are similar in structure but applied to smaller sets, with finer granularity to what can be controlled (saved or restored).

When restoring Scenes, a series of flags will enable protection of existing (already in place) parameters. These flags are listed as "Scene Safes" and address rather large groups of elements. A different set of flags enables what is actually saved with Snippets, controlling the affected areas in a much finer manner.

A complete list of elements found in Scenes and Snippet files is given in appendix of this document.

Scene 0 cannot have scene safes associated to it.

### Presets

The X32/M32 family of products can accept 3 different types of presets: Channel, Effects and Routing. Presets are files which can be loaded in one of the 3 x 100 preset memory slots of the X32/M32. They consist of `/node` like commands dedicated to the domain they address.

- Channel presets contain `/node` commands used for X32/M32 channels (i.e. beginning with `[/]ch/[01...32]/...`)
- Routing presets contain `/node` commands used for X32/M32 Routing (i.e. stating with `[/]config/routing/...`)
- Effects presets contain `/node` commands used for X32/M32 Effects (i.e. beginning with `[/]fx/[1...8]/...`)

In the case of Channels and Effects, the corresponding header is not present as Channel and Effect presets are dedicated to specific channels or effect slots.

A complete list of elements found in Channel, Effects and Routing preset files is given in appendix of this document.

Shows, Scenes, Snippets and Presets can be saved to and retrieved from memory or the USB drive with appropriate controls available on the different systems. They can also be controlled with the OSC commands below.

The `/showdump` command can be a way to read from the server all information related to Cues, Scenes, and Snippets for the current Show.

The `/-show` command is used to get/set elements and values related to Shows, Cues, Scenes and Snippets.

The `/-libs` command is used for listing and dealing with presets.

The `/add`, `/copy`, `/save`, `/load`, `/delete` and `/rename` commands are used to manage or update internal entities such as cues, scenes, snippets and presets within the X32/M32. A scene will save all data while a snippet will save small changes made to an existing scene. If the scene, snippet or preset already exists at the index provided in the command, the element at that given index is updated with new information. Otherwise, a new internal entity is created at the given index. These operators manage data between the X32/M32 internal storage (not the USB drive) and the X32/M32 audio engine state or preset libraries in memory.

Finally, the `/-undo` command is used to get back to a previous state/time.

### Show, Cue, Scene, Snippet, and Preset Management

<code>/showdump</code>	none	Request the X32/M32 to send all Cue, Scene and Snippet related data.
		Cues, Scenes and Snippets data are returned using one or more “ <i>node</i> ” format messages (see below). If no cues, scene, or snippets exist, only the first line is reported by the command (see below).
<code>/-show/prepos/current</code>	int	Scene page cue, scene or snippet slot highlighted line/index is <int> value. int = [1-099] <b>Note:</b> selection of cue/scene/snippet depends on the <code>/-prefs/show_control</code> command value. Scene 0 always exists and has no safes. It is a good practice not to change it, and use it as a start point for copy to another scene location before editing. It will also ensure by reloading it that your system will be back to an X32/M32 known factory state (unless you change scene 0 to reflect your own default state, of course).
<code>/-show/showfile/show/name</code>	string	Name of the current show
<b>Note/Bug:</b> In 2.08 the name changes only after the “utility” screen has been selected within the Scene/home screen		
<code>/-show/showfile/show/inputs</code>	%int	Param safe page Scene safe parameters Input channels selection: <i>bit 0: Preamp</i> <i>bit 1: Config</i> <i>bit 2: EQ</i> <i>bit 3: Gate &amp; Comp</i> <i>bit 4: Insert</i> <i>bit 5: Groups</i> <i>bit 6: Faders, Pan</i> <i>bit 7: Mute</i> e.g.: <int> = <i>0x00000024</i> : EQ and Groups are safe
<code>/-show/showfile/show/mxsends</code>	%int	Param safe page Scene safe parameters Input channels selection:

```

bit 0: Mix 1 Sends
...
bit 15: Mix 16 Sends
e.g.:
<int> = 0x00008001: mix 1 and 16 are safe
/-show/showfile/show/mxbuses int Param safe page Scene safe parameters Mix Buses
selection:
bit 0: Mix Sends
bit 1: Config
bit 2: EQ
bit 3: Comp
bit 4: Insert
bit 5: Groups
bit 6: Faders, Pan
bit 7: Mute
e.g.:
<int> = 0x00000024: EQ and Groups are safe
/-show/showfile/show/console %int Param safe page Scene safe parameters Console
selection:
bit 0: Configuration
bit 1: Solo
bit 2: Routing
bit 3: Outpatch

/-show/showfile/show/chan16 %int chan safe page Chanel safe parameters selection:
bit 0: chan 1
...
bit 15: chan 16
e.g.:
<int> = 0x00001002: chan 2 and 16 are safe
/-show/showfile/show/chan32 %int chan safe page Chanel safe parameters selection:
bit 0: chan 17
...
bit 15: chan 32
e.g.:
<int> = 0x00000001: chan 17 is safe
/-show/showfile/show/return %int chan safe page Return & Aux safe parameters selection:
bit 0: Aux 1
...
bit 7: Aux 8
bit 8: FX 1L
...
bit 15: FX 4R
/-show/showfile/show/buses %int chan safe page Buses safe parameters selection:
bit 0: Mix 1
...
bit 15: Mix 16
/-show/showfile/show/lrmtx dca %int chan safe page Buses safe parameters selection:
bit 0: Mtx 1
...
bit 5: Mtx 6
bit 6: L/R
bit 7: Mono/Center
bit 8: DCA group 1
...
bit 15: DCA group 8

```

<code>/-show/showfile/show/effects</code>	<code>%int</code>	chan safe page Effects Slots safe parameters selection: <i>bit 0: FX1</i> ... <i>bit 7: FX 8</i>
<code>/-show/showfile/cue/[000-099]/numb</code>	<code>int</code>	number of cue in the form xxx.x.x, saved at position [000-099] A value of 10327 means cue 103.2.7 A value of 49999 means cue 499.9.9 A value of 50000 means 500.0.0 (displayed as 500)
<code>/-show/showfile/cue/[000-099]/name</code>	<code>string</code>	Name of cue at position [000-099]
<code>/-show/showfile/cue/[000-099]/skip</code>	<code>int</code>	0 (no Skip) or 1 (Skip) for cue at position [000-099]
<code>/-show/showfile/cue/[000-099]/scene</code>	<code>int</code>	Associate Scene <int> with cue at position [000-099]
<code>/-show/showfile/cue/[000-099]/bit</code>	<code>int</code>	Associate Snippet <int> with cue at position [000-099]
<code>/-show/showfile/cue/[000-099]/miditype</code>	<code>int</code>	Associate MIDI type <int> with cue at position [000-099]. <int> can be one of: 0: none 1: program change 2: control change 3: note
<code>/-show/showfile/cue/[000-099]/midichan</code>	<code>int</code>	Set MIDI channel number to <int>
<code>/-show/showfile/cue/[000-099]/midipara1</code>	<code>int</code>	Set Midi parameter 1 value to <int>
<code>/-show/showfile/cue/[000-099]/midipara2</code>	<code>int</code>	Set Midi parameter 2 value to <int>
<code>/-show/showfile/scene/[000-099]/name</code>	<code>string</code>	Scene "Name" parameter for scene [000-099]
<code>/-show/showfile/scene/[000-099]/notes</code>	<code>string</code>	Scene "Notes" parameter for scene [000-099]
<code>/-show/showfile/scene/[000-099]/safes</code>	<code>%int</code>	Scene "Scene Safes" parameters selection for scene [000-099] <i>bit 1: Talkback</i> <i>bit 2: Effects</i> <i>bit 3: Mix Buses</i> <i>bit 4: Chan Process</i> <i>bit 5: Configuration</i> <i>bit 6: Preamp (HA)</i> <i>bit 7: Output Patch</i> <i>bit 8: Routing I/O</i> e.g.: <int> = 0x00000106: Routing I/O, Talkback and Effects are safe
<code>/-show/showfile/scene/[000-099]/hasdata</code>	<code>int</code>	Scene at position [000-099] has valid data 0: No 1: Yes
<code>/-show/showfile/snippet/[000-</code>	<code>string</code>	Snippet "Name" parameter for Snippet [000-099]

099]/name		
<code>/-show/showfile/snippet/[000-099]/eventtyp</code>	%int	Parameter Filters & Effects affected by snippet in the form of bitwise operation: <i>bit 0: Preamp HA</i> <i>bit 13: FX 1</i> <i>bit 1: Config</i> <i>bit 14: FX 2</i> <i>bit 2: EQ</i> <i>bit 15: FX 3</i> <i>bit 3: Gate &amp; Comp</i> <i>bit 16: FX 4</i> <i>bit 4: Insert</i> <i>bit 17: FX 5</i> <i>bit 5: Groups</i> <i>bit 18: FX 6</i> <i>bit 6: Fader, Pan</i> <i>bit 19: FX 7</i> <i>bit 7: Mute</i> <i>bit 20: FX 8</i> <i>bit 8: Send 1-8</i> <i>bit 9: Send 9-12</i> <i>bit 10: Send 13-16</i> <i>bit 21: Config</i> <i>bit 11: Send M/C</i> <i>bit 22: Solo</i> <i>bit 12: Send Matrix</i> <i>bit 23: Routing</i> <i>Bit 24: Out Patch</i>
<code>/-show/showfile/snippet/[000-099]/channels</code>	%int	Channels affected by snippet in the form of bitwise operation: <i>bit 0: channel 1</i> ... <i>bit 31: channel 32</i>
<code>/-show/showfile/snippet/[000-099]/auxbuses</code>	%int	Returns and Buses affected by snippet in the form of bitwise operation: <i>bit 0: Aux 1</i> ... <i>bit 15: FX 4R</i> <i>bit 16: Mix 1</i> ... <i>bit 31: Mix 16</i>
<code>/-show/showfile/snippet/[000-099]/maingrps</code>	%int	Main/Matrix/Group affected by snippet in the form of bitwise operation: <i>bit 0: Matrix 1</i> ... <i>bit 15: DCA Group 8</i>
<code>/-show/showfile/snippet/[000-099]/hasdata</code>	int	Snippet at position [000-099] has valid data 0: No 1: Yes
<code>/-libs/ch/[001-100]/pos</code>	int	The position of the channel preset number [001-100]
<code>/-libs/ch/[001-100]/name</code>	string	Name of the channel preset
<code>/-libs/ch/[001-100]/type</code>	int	Type of the channel preset
<code>/-libs/ch/[001-100]/flags</code>	%int	Lists the scope elements for the channel preset index [001-100] in the form of bitwise operation: <i>bit 0: preamp phantom ON</i> <i>bit 1: config: delay ON</i> <i>bit 2: LoCut ON</i> <i>bit 3: Gate ON</i> <i>bit 4: EQ ON</i> <i>bit 5: Dyn ON</i> <i>bit 6: 0</i> <i>bit 7: 0</i> <i>bit 8: preset has a preamp section</i> <i>bit 9: preset has a config section</i> <i>bit 10: preset has a LoCut section</i>

		<i>bit 11: preset has a Gate section</i> <i>bit 12: preset has a EQ section</i> <i>bit 13: preset has a Dyn section</i> <i>bit 14: 0</i> <i>bit 15: 0</i>
<code>/-libs/ch/[001-100]/hasdata</code>	int	[0, 1] depending on the validity of the channel preset.
<code>/-libs/fx/[001-100]/pos</code>	int	The position of the effect preset number [001-100]
<code>/-libs/fx/[001-100]/name</code>	string	Name of the effect preset
<code>/-libs/fx/[001-100]/type</code>	int	Type of the effect preset
<code>/-libs/fx/[001-100]/flags</code>	%int	Use as an int to list the effect type “Ambiance”, Plate Reverb”, etc. at the right of the effect name on the X32/M32 screen. <sup>22</sup> <b>Note: int values do not match with FX enums!</b>
<code>/-libs/fx/[001-100]/hasdata</code>	int	[0, 1] depending on the validity of the effect preset.
<code>/-libs/r/[001-100]/pos</code>	int	The position of the routing preset number [001-100]
<code>/-libs/r/[001-100]/name</code>	string	Name of the effect routing
<code>/-libs/r/[001-100]/type</code>	int	Type of the effect routing
<code>/-libs/r/[001-100]/flags</code>	%int	Unused (all 0).
<code>/-libs/r/[001-100]/hasdata</code>	int	[0, 1] depending on the validity of the routing preset.
<code>/copy</code>	string, int, int, [%int]	<p>Copies an X32/M32 internal set to another. The type of internal set is listed with the first <i>&lt;string&gt;</i> parameter and can be <i>scene</i>, <i>snippet</i><sup>23</sup>, <i>libchan</i>, <i>libfx</i> or <i>librout</i> for scene, snippet, channel, effect and routing presets respectively.</p> <p>The next <i>&lt;int&gt;</i> is the source index, and is followed by another <i>&lt;int&gt;</i> representing the destination index.</p> <p><b>Index values start at 0.</b></p> <p>In the case of channel copy, the last <i>&lt;%int&gt;</i> is a bitmap field specifying which attribute or subset is copied:</p> <p><i>bit 0: headamp</i>  <i>bit 1: config</i>  <i>bit 2: gate</i>  <i>bit 3: dynamics</i>  <i>bit 4: EQ</i>  <i>bit 5: sends</i></p>
<code>/add</code>	string, int, string	<p>Adds a cue element to the current show in the X32/M32 internal memory.</p> <p>The first parameter is a string: <i>cue</i></p> <p>The second parameter is an <i>&lt;int&gt;</i> representing the cue index and subindex. For example</p> <p>cue index data <i>1.0.0</i> will have <i>int=100</i> for value;  cue index data <i>12.5.2</i> will have <i>int=1252</i> for value;</p>

<sup>22</sup> See Appendix for table of enum/name/type

<sup>23</sup> It seems `/copy` does not allow snippets copying in FW 2.14.

The third parameter is a *<string>* representing the cue name.

The added cue will save the current values of *skip*, *scene*, *snippet*, *midichan*, *midipar1* and *midipar2* associated with the cue

/save

string, Saves or updates in the X32/M32 internal memory a  
int, scene, snippet or preset at a given index with  
[int | information specific to the object saved;  
string,  
...] The first parameter is a string representing the type of  
element to save. It can be one of: *scene*, *snippet*,  
*libchan*, *libfx* or *librout* for saving a scene, a  
snippet, a channel preset, an effect preset or a routing  
reset, respectively.

The other parameters depend on the object to save.

**Scenes:** the first parameter is followed by *<int>*,  
*<string>*, *<string>* representing respectively the  
scene position **index** in the range [000-099] and the  
**name** and **note** given to the scene.

**Snippets:** the first parameter is followed by *<int>*,  
*<string >* representing respectively the snippet  
position **index** in the range [000-099] and the **name**  
given to the snippet. The snippet is saved accordingly to  
parameter filters set for *eventtyp*, *channels*,  
*auxbuses* and *maingrps* associated with the snippet.

**Channel presets:** the first parameter is followed by  
*<int>*, *<string>*, *<int>* representing respectively the  
channel preset position **index** in the range [000-099],  
the **name** of the preset, and the last *<int>* parameter  
specifies the channel **index** relevant to the preset  
starting at 0 / ch01.

**Effect presets:** the first parameter is followed by *<int>*,  
*<string>*, *<int>* representing respectively the effect  
preset position **index** in the range [000-099], the **name**  
of the preset, and the last *<int>* parameter specifies the  
effect slot **index** relevant to the preset starting at 0,  
in the range [0...7].

**Routing presets:** the first parameter is followed by  
*<int>*, *<string>* representing respectively the routing  
preset position **index** in the range [000-099], and the  
**name** of the preset.

Upon completion, the server returns a status indicating if the operation failed [0] or was successful [1], e.g.:  
->X: /save ,sissi scene 45 test note 0  
X->: /save~~~,si~scene~~~[1]

/load

string, Loads from the X32/M32 internal memory a scene,  
int snippet or a preset listed at a given index with  
[,int[, information specific to the state/audio engine;  
int]]  
The first parameter is a string representing the type of element to save. It can be one of: *scene*, *snippet*, *libchan*, *libfx* or *librout* for loading a scene, a snippet, a channel preset, an effect preset or a routing preset, respectively.

The second parameter represents the **index** of the element to load, in the range [000-099].

The next two parameters are not necessarily present, depending on the type of element being loaded

Channel presets:

The third parameter represents the channel **index** the preset is loaded to, in the range [0-71]

The fourth parameter, a value [0...63] represents the **scope** of elements being loaded to the channel, built from "or"ing bits as follows:

*Bit 0: Head Amp*  
*Bit 1: Configuration*  
*Bit 2: Gate*  
*Bit 3: Compressor*  
*Bit 4: Equalizer*  
*Bit 5: Sends*

Effects presets:

The third parameter represents the effect **index** the preset is loaded to, in the range [0...7]

Routing presets:

No additional parameters

Upon completion, the server returns a status indicating if the operation failed [0] or was successful [1], e.g.:

->X: /load ,si scene 99  
X->: /load~,si~scene~~~[1]

/rename

string, Renames in the X32/M32 internal memory a scene,  
int, snippet or a preset listed at a given index;  
string  
The first parameter is a string representing the type of



element to save. It can be one of: *scene, snippet, libchan, libfx or librout* for loading a scene, a snippet, a channel preset, an effect preset or a routing reset, respectively.

The second parameter represents the **index** of the element to load, in the range [000-099].

The third parameter, a string, is the new **name** assigned to the element

Upon completion, the server returns a status indicating if the operation failed [0] or was successful [1], e.g.:

```
->X: /load ,si scene 99
X->: /load~,si~scene~~~[1]
```

`/delete` string, int Deletes from the X32/M32 internal memory an element at given index;

The first parameter is a string: *scene, snippet, libchan, libfx or librout*, giving the type of element to delete.

The second parameter is an **index** of the element to delete in the range [000-099].

Upon completion, the server returns a status indicating if the operation failed [0] or was successful [1], e.g.:

```
->X: /delete ,si scene 99
X->: /delete~,si~scene~~~[1]
```

`/-undo/time` string Display/Set **[TBV]** a time value for changes, for example in selecting a scene. Time is coded as string, e.g.

**Note:** It seems there's only 1 undo step in X32 FW 2.08

```
18:36:54
```

*If string is empty: there are no more undo steps available*

**Note/bug:** in FW 2.08, it seems that Scenes and Snippets numbers associated to Cues are not always listed correctly on the X32 LCD SCENES screen, under home page; they can appear listed on the first line rather than respective of their associated Cue index. Selecting/Associating Scenes and Snippets to Cues AFTER cues are created seem to avoid this issue.

**Note/bug:** in FW 2.08, specifically on X32 CORE, it seems that upon asking to load a show from a USB drive, the last snippet from the show is not loaded; it is therefore advisable to add an empty snippet at the end of the list of snippets. This does not happen on X32 standard.

**Note:** The *"node"* returning command does not comply to OSC standard (no leading *"/"*); the returned string is *"\n"* (a.k.a *0x0a*) terminated, which makes it suitable for direct printing or editing with a standard text editor.

### Notes on the use of /showdump

/showdump will trigger the X32/M32 server to dump all Scene and Snippet data back to the requesting client. This can generate a large amount of data back to the client, with possible overruns in UDP packets. It is important to ensure a very reliable connection is in place between the X32/M32 and the receiving device.

By experience, 54Mbps/s WIFI is not recommended for Shows with more than 20 lines (cues, scenes or snippets) as there is a high probability of UDP buffer overflow/overrun. Higher data throughput rate are recommended, or better, a 100Mbps/s wired connection.

Replies to client are formatted as per the "node" exchange format, as shown in the examples below.

X32/M32 does not have any scene or snippet, the answer to a /showdump request is:

```
node~~~~,s~/--showfile/show "MyShow" 0 0 0 0 0 0 0 0 0 0 "2.08"
```

X32/M32 has a single scene (in scene slot 01, name: AAA, note: aaa) with Routing IO and Output Patch selected as Scene Safes, no snippet, the answer to a /showdump request is:

```
node~~~~,s~/--show/showfile/show "MyShow" 0 0 0 0 0 0 0 0 0 0 "2.08"  
node~~~~,s~/--show/showfile/scene/001 "AAA" "aaa" %110000000 1
```

X32/M32 has a single scene (in scene slot 01, name: AAA, note: aaa) with all items selected as Scene Safes, no snippet, the answer to a /showdump request is:

```
node~~~~,s~/--show/showfile/show "MyShow" 0 0 0 0 0 0 0 0 0 0 "2.08"  
node~~~~,s~/--show/showfile/scene/001 "AAA" "aaa" %111111110 1
```

We now add a new scene (in scene slot 02, name: BBB, note: bbb) with Talkback selected as Scene Safes, no snippet, the answer to a /showdump request is:

```
node~~~~,s~/--show/showfile/show "MyShow" 0 0 0 0 0 0 0 0 0 0 "2.08"  
node~~~~,s~/--show/showfile/scene/001 "AAA" "aaa" %111111110 1  
node~~~~,s~/--show/showfile/scene/002 "BBB" "bbb" %000000010 1
```

Keeping the 2 scenes created above, we create a snippet in slot 00, with name: Aaa, selecting Parameter Filter Preamp(HA) and Channels Ch. The answer to a /showdump request is:

```
node~~~~,s~/--show/showfile/show "MyShow" 0 0 0 0 0 0 0 0 0 0 "2.08"  
node~~~~,s~/--show/showfile/scene/001 "AAA" "aaa" %111111110 1  
node~~~~,s~/--show/showfile/scene/002 "BBB" "bbb" %000000010 1  
node~~~~,s~/--show/showfile/snippet/000 "Aaa" 1 1 0 0 1
```

Updating snippet in slot 00 with selecting Main/Matrix/Group parameter DCA 8, and saving the snippet to slot00 with no other changes, the answer to a /showdump request is:

```
node~~~~,s~/--show/showfile/show "MyShow" 0 0 0 0 0 0 0 0 0 0 "2.08"  
node~~~~,s~/--show/showfile/scene/001 "AAA" "aaa" %111111110 1  
node~~~~,s~/--show/showfile/scene/002 "BBB" "bbb" %000000010 1  
node~~~~,s~/--show/showfile/snippet/000 "Aaa" 1 1 0 32768 1
```

Keeping all data unchanged, we create a cue, name it "CCC", at index 1, selecting scene = -1 (none) and snippet = -1 (none). The answer to a /showdump request is:

```
node~~~~,s~/--show/showfile/show "MyShow" 0 0 0 0 0 0 0 0 0 0 "2.08"  
node~~~~,s~/--show/showfile/cue/000 100 "CCC" 0 -1 -1 0 1 0 0  
node~~~~,s~/--show/showfile/scene/001 "AAA" "aaa" %111111110 1  
node~~~~,s~/--show/showfile/scene/002 "BBB" "bbb" %000000010 1  
node~~~~,s~/--show/showfile/snippet/000 "Aaa" 1 1 0 32768 1
```

Keeping all data unchanged, we create a new cue, name it "Ccc", at index 1.1, selecting scene 01 and snippet = -1. The answer to a `/showdump` request is:

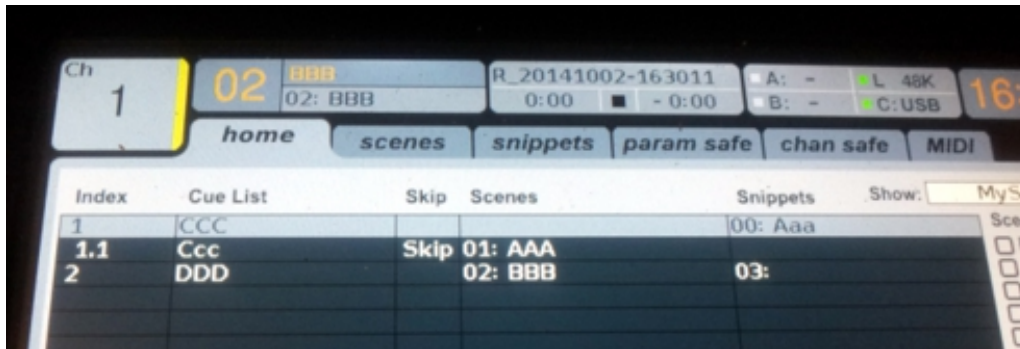
```
node~~~~,s~/show/showfile/show "MyShow" 0 0 0 0 0 0 0 0 0 0 "2.08"
node~~~~,s~/show/showfile/cue/000 100 "CCC" 0 -1 -1 0 1 0 0
node~~~~,s~/show/showfile/cue/001 110 "Ccc" 0 1 -1 0 1 0 0
node~~~~,s~/show/showfile/scene/001 "AAA" "aaa" %111111110 1
node~~~~,s~/show/showfile/scene/002 "BBB" "bbb" %000000010 1
node~~~~,s~/show/showfile/snippet/000 "Aaa" 1 1 0 32768 1
```

Selecting "skip" on cue Ccc, the node answer appears as:

```
node~~~~,s~/show/showfile/cue/001 110 "Ccc" 1 1 -1 0 1 0 0
```

Updating cue CCC with snippet 0 (Aaa) selected, the node answer appears as:

```
node~~~~,s~/show/showfile/cue/001 100 "CCC" 0 -1 0 0 1 0 0
```



Keeping all data unchanged, we create a new cue at index 2, name it "DDD", selecting scene 02 and snippet = 03. The answer to a `/showdump` request is:

```
node~~~~,s~/show/showfile/show "MyShow" 2 2 1 1 0 0 1 1 0 0 "2.08"
node~~~~,s~/show/showfile/cue/000 100 "CCC" 0 -1 0 0 1 0 0
node~~~~,s~/show/showfile/cue/001 110 "Ccc" 1 1 -1 0 1 0 0
node~~~~,s~/show/showfile/cue/002 200 "DDD" 0 2 3 0 1 0 0
node~~~~,s~/show/showfile/scene/001 "AAA" "aaa" %111111110 1
node~~~~,s~/show/showfile/scene/002 "BBB" "bbb" %000000010 1
node~~~~,s~/show/showfile/snippet/000 "Aaa" 1 1 0 32768 1
```

## Preferences (/prefs) data

Preferences data			
/-prefs/style	string	Name given to your prefs ex: "Patrick". Will be "ablesque" after factory reset	
/-prefs/bright	float	Main LCD Brightness: [0.0...1.0]	
/-prefs/lcdcont	float	Channel LCD Contrast: [0.0...1.0]	
/-prefs/ledbright	float	LED Brightness: [0.0...1.0]	
/-prefs/lamp	float	Lamp dim value: [0.0...1.0]	
/-prefs/lampon	enum	{OFF, ON} Lamp is : 0: off 1: on	
/-prefs/clockrate	enum	int [0, 1] representing {48K, 44K1} "Sample Rate" in Global	
/-prefs/clocksource	enum	int [0...3] representing {INT, AES50A, AES50B, Exp Card} "Synchronization" in Global	
/-prefs/confirm_general	enum	{OFF, ON} "General" in Config->Confirm Pop-ups	
/-prefs/confirm_overwrite	enum	{OFF, ON} "Overwrite" in Config->Confirm Pop-ups	
/-prefs/confirm_sceneload	enum	{OFF, ON} "Scene Load" in Config->Confirm Pop-ups	
/-prefs/viewwrtn	enum	{OFF, ON} "Return to Last" in Config->View Preferences	
/-prefs/selffollowsbank	enum	{OFF, ON} "Sel follows Bank" in Config->View Preferences	
/-prefs/sceneadvance	enum	{OFF, ON} "Scene Go Next" in Config->General Prefs	
/-prefs/safe_masterlevels	enum	{OFF, ON} "Safe Main Levels" in Config->General Prefs	
/-prefs/haflags	%int	Global parameters: <int> is a bitmask bit 0: Lock Stagebox bit 1: X32 HA Gain split mode bit 2: AES50/A HA Gain split mode bit 3: AES50/B HA Gain split mode	
/-prefs/autosel	enum	{OFF, ON} "Auto Select" in Config->View Preferences	
/-prefs/show_control	enum	int [0...2] representing {CUES, SCENES, SNIPPETS} "Show Control" in Config	
/-prefs/clockmode	enum	{OFF, ON} "12h Clock Mode" in Config->General Prefs	
/-prefs/hardmute	enum	{OFF, ON} "Hard Mutes" in Config->Mute System	
/-prefs/dcamute	enum	{OFF, ON} "DCA groups" in Config->Mute System	
/-prefs/invertmute	enum	{OFF, ON} "Invert Leds" in Config->Mute System	
/-prefs/remote/enable	enum	{OFF, ON} set or report X32/M32 remote enable state	
/-prefs/remote/protocol	int	Protocol type for X32/M32 Remote 0: Mackie HCU 1: Mackie HUI 2: Generic CC	
/-prefs/remote/port	int	Port used for MIDI remote 0: MIDI in/Out 1: Card MIDI 2: RTP MIDI	
/-prefs/remote/ioenable	%int	Enables X32/M32's Remote mode <int> defines the set of Remote features, using bitwise OR operator, bit 0: MIDI In/Out, bit 1: Card MIDI bit 2: RTP MIDI bit 3: Rx PrgC bit 4: Tx PrgC	

*bit 5: Rx Fader*  
*bit 6: TX Fader*  
*bit 7: Rx Mute*  
*bit 8: Tx Mute*  
*bit 9: Rx Pan*  
*bit 10: Tx Pan*  
*bit 11: X/OSC*

<code>/-prefs/iQ/[01-16]/iQmodel</code>	enum	int [0...6] representing {none, iQ8, iQ10, iQ12, iQ15, iQ15B, iQ18B} Type of Turbosound iQ speakers <i>0: none</i> <i>1: iQ8</i> <i>2: iQ10</i> <i>3: iQ12</i> <i>4: iQ15</i> <i>5: iQ15B</i> <i>6: iQ18B</i>	
<code>/-prefs/iQ/[01-16]/?</code>	enum	int [0...4] representing {Linear, Live, Speech, Playback, User} Type of Turbosound iQ speakers <i>0: Linear</i> <i>1: Live</i> <i>2: Speech</i> <i>3: Playback</i> <i>4: User</i>	
<code>/-prefs/iQ/[01-16]/?</code>	?	0...	
<code>/-prefs/card/UFifc</code>	int		
<code>/-prefs/card/UFmode</code>	int		
<code>/-prefs/card/USBmode</code>	int	X-USB card settings <i>0: 32in/32out</i> <i>1: 16in/16out</i> <i>2: 32in/8out</i> <i>3: 8in/32out</i> <i>4: 8in/8out</i> <i>5: 2in/2out</i>	
<code>/-prefs/card/ADATwc</code>	int		
<code>/-prefs/card/ADATsync</code>	int		
<code>/-prefs/card/MADlmode</code>	int		
<code>/-prefs/card/MADlin</code>	int		
<code>/-prefs/card/MADlout</code>	int		
<code>/-prefs/card/MADlsrc</code>	int		
<code>/-prefs/rta/visibility</code>	enum	int [0...12] representing RTA EQ Overlay value {OFF, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80}	%
<code>/-prefs/rta/gain</code>	linf	[0.0, 60.0, 6] RTA gain value (steps of 6dB)	dB
<code>/-prefs/rta/autogain</code>	enum	{OFF, ON} RTA autogain <i>0: disabled (OFF)</i> <i>1: set/enabled (ON)</i>	
<code>/-prefs/rta/source</code>	int	RTA source:	

```

0:      none
1:      Monitor
2...33: Ch01...Ch32
34...41: Aux1...Aux8
42...49: FX1L...FX4R
50...65: Bus01...Bus16
66...71: Mtx1...Mtx6
72:     Main
73:     Mono

(see also /-stat/rtaresource)

```

```

/-prefs/rta/pos          enum    {PRE, POST}
                          RTA chain position
                          0: Pre EQ
                          1: Post EQ
/-prefs/rta/mode         enum    {BAR, SPEC}
                          RTA display mode selection
                          0: Bar[graph]
                          1: Spec[trograph]
/-prefs/rta/option       %int    Describes which RTA options are set, using bitwise OR
                          operator,
                          bit 0 = Pre EQ
                          bit 1 = Spectrograph
                          bit 2 = Use RTA source
                          bit 3 = Post GEQ
                          bit 4 = Spectrograph
                          bit 5 = Solo Priority

                          e.g. int = 0x0021: solo priority and Pre EQ are set
/-prefs/rta/det          enum    {RMS, PEAK} RTA detector selection
/-prefs/rta/decay        logf    [0.25, 16, 19] RTA adjustable decay time24
/-prefs/rta/peakhold     enum    {OFF, 1...8} RTA peak hold

/-prefs/ip/dhcp          enum    {OFF, ON}. Use with Caution!
/-prefs/ip/addr/[0...3] int     IP address value. Use with Caution!
/-prefs/ip/mask/[0...3] int     IP mask value. Use with Caution!
/-prefs/ip/gateway/[0...3] int    IP gateway value. Use with Caution!

```

<sup>24</sup> See Appendix section for detailed values

## USB (/usb) data

This section enables accessing and setting some of the parameters of the USB stick.

**Note:** all options may not be enabled/documentated.

USB (/usb)		
/usb/path	string	Name of the current directory, e.g.: <i>/usb/path~,s~~Db blues 48kHz~~~~</i>
/usb/title	string	Name of a file in the current directory of USB stick, e.g.: <i>/usb/title~,s~~Candy-DB~~~~</i>
/usb/dir/dirpos	int	Current directory entry number
/usb/dir/maxpos	int	Number of entries of the current directory in USB stick, e.g.: <i>/usb/dir/maxpos~~~~,i~~&lt;int=16&gt;</i>
/usb/dir/001...999/type	string	The name of file at position 000...999 of current directory in USB stick, e.g.: <i>/usb/dir/006/type~~,s~~~~~</i>
/usb/dir/001...999/name	string	The name of file at position 000...999 of current directory in USB stick, e.g.: <i>/usb/dir/006/name~~,s~~Candy.wav~~</i> <i>The file "candy.wav" is at position 6 in</i> <i>the current directory</i>  Can also return the name of a directory in the usb stick, e.g.: <i>/usb/dir/001/name~~,s~~[...]~~~~</i> <i>/usb/dir/002/name~~,s~~[DBlues 41Khz]~~</i> <i>/usb/dir/003/name~~,s~~[Db blues 48kHz]~~</i>

## Action (/action) data

Action data		
/action/playtrack	int	Plays track from USB recorder <i>-1: previous track</i> <i>1: next track</i>
/action/recselect	int	Select and execute record <int> in the current directory. Records are numbered 1...n
/action/goscene	int	Loading a saved scene; the Scene number to load is given as an int parameter ranging from 0 to 99
/action/gosnippet	int	Loading a saved snippet; the Snippet number to load is given as an int parameter ranging from 0 to 99
/action/gocue	int	Loading a saved cue; the Cue number to load is given as an int parameter ranging from 0 to 99
/action/setrtasrc	int	Selects the source used for RTA display: <int> represents the channel # <i>0-31: Ch 1-32</i> <i>32-63: Ch 33-64</i>

64-47: Aux in /USB  
48-63: Bus master  
64-69: Matrix 1-6  
70: L/R  
71: Mono/Center  
72 : Monitor

<code>/-action/undopt</code>	int	Sets checkpoint to get back to upon issuing an undo command
<code>/-action/newscreen</code>	int	LCD screen display has changed? (to be verified) 0: no 1: yes
<code>/-action/setip</code>	int	0 by default; changing to 1 resets Network parameters. Use with Caution!



## Status (/stat) data

Status data (screen, tape, fader groups, solo, etc.)

<code>/-stat/selidx</code>	int	Select channel index <i>0-31: Ch 1-32</i> <i>32-39: Aux in 1-8</i> <i>40-47: FxRtn 1-8</i> <i>48-63: Bus master</i> <i>64-69: Matrix 1-6</i> <i>70: L/R</i> <i>71: Mono/Center</i>
<code>/-stat/chfaderbank</code>	int	Selected Main channel fader bank: <i>0: CH 1-16</i> <i>1: CH 17-32</i> <i>2: Aux in / USB / FX returns</i> <i>3: Bus masters</i>
<code>/-stat/grpfaderbank</code>	int	Selected Group channel fader bank: <i>0: DCA 1-8</i> <i>1: BUS 1-8</i> <i>2: BUS 9-16</i> <i>3: Matrix 1-6, Main C</i>
<code>/-stat/sendsonfader</code>	enum	{OFF, ON} state of Sends on Faders
???	int	0
???	int	0
<code>/-stat/solo</code>	enum	{OFF, ON} (read only) State of SOLO / Clear SOLO button <i>0: no SOLO selected</i> <i>1: at least one SOLO selected</i> <i>(Clear SOLO blinking)</i>
<code>/-stat/keysolo</code>	enum	{OFF, ON}
<code>/-stat/userbank</code>	int	Display User ASIGN bank settings on X32/M32 (pressing on SET A/B/C buttons): <i>0: User bank A</i> <i>1: User bank B</i> <i>2: User bank C</i>
<code>/-stat/autosave</code>	enum	{OFF, ON} X32/M32 saves automatically its state (every 2mns?)
<code>/-stat/lock</code>	int	X32/M32 Lock status: <i>0: unlocked</i> <i>1: locked</i>
<code>/-stat/usbmounted</code>	enum	{OFF, ON} USB drive mount status: <i>0: not mounted</i> <i>1: mounted</i>
<code>/-stat/remote</code>	enum	{OFF, ON} Remote mode: <i>0: X32 in Audio Console mode</i> <i>1: X32 in DAW mode</i>
???	enum	{BAR, ...}
<code>/-stat/rtamodegeq</code>	enum	{BAR, SPEC} RTA display mode selection for GEQ effect <i>0: Bar[graph]</i> <i>1: Spec[trograph]</i>
???	enum	{OFF, ON}
<code>/-stat/rtageqpost</code>	enum	{OFF, ON}

		RTA chain position selection for GEQ effect
		0: <i>pre</i>
		1: <i>post</i>
<code>/-stat/rtasource</code>	int	RTA source: 0...31: <i>Channel 01...32, PRE-EQ</i> 32...39: <i>Aux 01...08, PRE-EQ</i> 40...47: <i>Fxrtn 1L...4R, PRE-EQ</i> 48...63: <i>Bus 01...16, PRE-EQ</i> 64...69: <i>Matrix 01...06, PRE-EQ</i> 70: <i>L/R, PRE-EQ</i> 71: <i>Mono, PRE-EQ</i> 72: <i>Monitor, PRE-EQ</i> ... 98...129: <i>Channel 01...32, POST-EQ</i> 130...137: <i>Aux 01...08, POST-EQ</i> 138...145: <i>Fxrtn 1L...4R, POST-EQ</i> 146...161: <i>Bus 01...16, POST-EQ</i> 162...167: <i>Matrix 01...06, POST-EQ</i> 168: <i>L/R, POST-EQ</i> 169: <i>Mono, POST-EQ</i> 170: <i>Monitor, POST-EQ</i>
		!! after Console Reset, the value of RTA source may not reflect the <i>METERS/RTA</i> screen settings (see also <code>/-prefs/rta/source</code> )
<code>/-stat/xcardtype</code>	int	Type of card installed (seems to be informative only) Valid values: 0: <i>none</i> 1: <i>X-UF 32in/32out Firewire/USB Card</i> 2: <i>X-USB 32in/32out USB Card</i> 3: <i>X-DANTE 32in/32out Dante Card</i> 4: <i>X-ADAT 4in/4out 32ch ADAT Card</i> 5: <i>X-MADI 32ch MADI Card</i> 6: <i>DN32-USB 32in/32out USB Card</i> 7: <i>DN32-DANTE 32in/32out Dante Card</i> 8: <i>DN32-ADAT 4in/4out 32ch ADAT Card</i> 9: <i>DN32-MADI 32ch MADI Card</i>
<code>/-stat/xcardsync</code>	enum	{OFF, ON}
<code>/-stat/geqonfdr</code>	enum	{OFF, ON} for EQ on faders: 0: <i>off</i> 1: <i>on</i>
<code>/-stat/geqpos</code>	%int	EQ on faders window position Bitwise OR between the FX number and the 8 band window start position: <FX#>   <start pos>, e.g.: <i>0x100...0x800   0x00...0x17</i>  <i>/-stat/geqpos~~~,i~~&lt;0x00000105&gt;</i>
		Means EQ on faders for effect slot #1, fader window starting at fader 5, covering bands 50...250Hz
<code>/-stat/screen/screen</code>	int	X32/M32 LCD active screen: 0: <i>HOME screen</i> 1: <i>METERS screen</i> 2: <i>ROUTING screen</i>

			3: <i>SETUP</i> screen
			4: <i>LIBRARY</i> screen
			5: <i>EFFECTS</i> screen
			6: <i>MONITOR</i> screen
			7: <i>USB RECORDER</i> screen
			8: <i>SCENES</i> screen
			9: <i>ASSIGN</i> screen
			10: <i>LOCK</i> screen (get only, can only be set via / <i>-stat/lock</i> command)
<i>/-stat/screen/mutegrp</i>	enum		{OFF, ON} 0: Turn off mutegrp screen 1: Turn on mutegrp screen
<i>/-stat/screen/utils</i>	enum		{OFF, ON} 0: Turn off utils screen 1: Turn on utils screen
<i>/-stat/screen/CHAN/page</i>	int		X32/M32 page in "Home" screen 0: <i>home</i> 1: <i>config</i> 2: <i>gate</i> 3: <i>dyn</i> 4: <i>eq</i> 5: <i>sends</i> 6: <i>main</i>
<i>/-stat/screen/METER/page</i>	int		X32/M32 page in "Meters" screen 0: <i>channel</i> 1: <i>mix bus</i> 2: <i>aux/fx</i> 3: <i>in/out</i> 4: <i>rta</i>
<i>/-stat/screen/ROUTE/page</i>	int		X32/M32 page in "Routing" screen 0: <i>home</i> 1: <i>out 1-16</i> 2: <i>aux out</i> 3: <i>p16 out</i> 4: <i>card out</i> 5: <i>aes50-a</i> 6: <i>aes50-b</i> 7: <i>xlr out</i>
<i>/-stat/screen/SETUP/page</i>	int		X32/M32 page in "Setup" screen 0: <i>global</i> 1: <i>config</i> 2: <i>remote</i> 3: <i>network</i> 4: <i>scribble strip</i> 5: <i>preamps</i> 6: <i>card</i>
<i>/-stat/screen/LIB/page</i>	int		X32/M32 page in "Library" screen, loading presets and options is translated into individual settings. 0: <i>channel</i> 1: <i>effects</i> 2: <i>routing</i>
<i>/-stat/screen/FX/page</i>	int		X32/M32 page in "Effects" screen 0: <i>home</i> 1: <i>fx1</i> 2: <i>fx2</i> ...

		7: <i>fx7</i> 8: <i>fx8</i>
<code>/-stat/screen/MON/page</code>	int	X32/M32 page in "Monitor" screen 0: <i>monitor</i> 1: <i>talkback A</i> 2: <i>talkback B</i> 3: <i>oscillator</i>
<code>/-stat/screen/USB/page</code>	int	X32/M32 page in "USB Recorder" screen 0: <i>home</i> 1: <i>config</i>
<code>/-stat/screen/SCENE/page</code>	int	X32/M32 page in "Scene" screen 0: <i>home</i> 1: <i>scenes</i> 2: <i>snippets</i> 3: <i>param safe</i> 4: <i>chan safe</i> 5: <i>MIDI</i>
<code>/-stat/screen/ASSIGN/page</code>	int	X32/M32 page in "Assign" screen 0: <i>home</i> 1: <i>Set A</i> 2: <i>Set B</i> 3: <i>Set C</i>
<code>/-stat/screen/USER/page</code>	int	Select user page in "Home"-screen
<code>/-stat/tape/state</code>	int	Sets playback/record state: 0: <i>stop (inclusive full rewind)</i> 1: <i>playback pause</i> 2: <i>play</i> 3: <i>record pause</i> 4: <i>record</i> 5: <i>Fast forward</i> 6: <i>Rewind</i>
<code>/-stat/solosw/[id]</code>	enum	{OFF, ON} 0/1 for on/off of solo switch <i>[id]: 01-32: Ch 01-32</i> <i>33-40: Aux in 1-8</i> <i>41-48: FxRtn 1-8</i> <i>49-64: Bus master 01-16</i> <i>65-70: Matrix 1-6</i> <i>71: L/R</i> <i>72: Mono/Center</i> <i>73-80: DCA 1-8</i>
<code>/-stat/aes50/A</code>	string	
<code>/-stat/aes50/B</code>	string	
<code>/-stat/aes50/state</code>	int	
<code>/-stat/tape/state</code>	Int	Tape state: 0: <i>Stop</i> 1: <i>Pause</i> 2: <i>Play</i> 3: <i>Pause Record</i> 4: <i>Record</i> 5: <i>FF</i> 6: <i>REW</i>

<code>/-stat/tape/file</code>	string	File path , ex: <code>"/dir000/R_20130105-205752.wav"</code>
<code>/-stat/tape/etime</code>	int	Elapsed time in seconds during playback or recording file (every second)
<code>/-stat/tape/rtime</code>	int	Remaining time in seconds of playing media or file (every second)

## Node (/node) command

The /node command can be used by clients to request values and data for the OSC path provided with the request. The server returns the full set of data associated to the request in a single string of text, ending with a linefeed.

Node command		
/node	string	Request the X32/M32 to return the data associated with the OSC path given in argument.

### Example of request:

```
/node~~~,s~~headamp/124~ !! note: no '/' before the request string
```

### Example of associated response from the server:

```
node~~~~,s~~/headamp/124 +0.0 OFF\n~~~~
```

List of accepted/known <OSC path> parameters. All the items below follow a /node~~~,s~~ "header".

```
config/chlink  
config/auxlink  
config/fxlink  
config/buslink  
config/mtxlink  
config/mute  
config/linkcfg  
config/mono  
config/solo  
config/talk  
config/talk/A  
config/talk/B  
config/osc  
config/routing/IN  
config/routing/AES50A  
config/routing/AES50B  
config/routing/CARD  
config/routing/OUT  
config/userctrl/{A,B,C}  
config/userctrl/{A,B,C}/enc  
config/userctrl/{A,B,C}/btn  
config/tape
```

```
ch/[01...32]/config  
ch/[01...32]/delay  
ch/[01...32]/preamp  
ch/[01...32]/gate  
ch/[01...32]/gate/filter  
ch/[01...32]/dyn  
ch/[01...32]/dyn/filter  
ch/[01...32]/insert  
ch/[01...32]/eq  
ch/[01...32]/eq/[1...4]  
ch/[01...32]/mix  
ch/[01...32]/mix/[01...16]  
ch/[01...32]/grp
```

```
auxin/[01...08]/config
```

auxin/[01...08]/preamp  
auxin/[01...08]/eq  
auxin/[01...08]/eq/[1...4]  
auxin/[01...08]/mix  
auxin/[01...08]/mix/[01...16]  
auxin/[01...08]/grp

fxrtn/[01...08]/config  
fxrtn/[01...08]/eq  
fxrtn/[01...08]/eq[1...4]  
fxrtn/[01...08]/mix  
fxrtn/[01...08]/mix/[01...16]  
fxrtn/[01...08]/grp

bus/[01...16]/config  
bus/[01...16]/dyn  
bus/[01...16]/dyn/filter  
bus/[01...16]/insert  
bus/[01...16]/eq  
bus/[01...16]/eq[1...6]  
bus/[01...16]/mix  
bus/[01...16]/mix/[01...06]  
bus/[01...16]/grp

mtx/[01...06]/config  
mtx/[01...06]/preamp  
mtx/[01...06]/dyn  
mtx/[01...06]/dyn/filter  
mtx/[01...06]/insert  
mtx/[01...06]/eq  
mtx/[01...06]/eq[1...6]  
mtx/[01...06]/mix

main/st/config  
main/st/dyn  
main/st/dyn/filter  
main/st/insert  
main/st/eq  
main/st/eq[1...6]  
main/st/mix  
main/st/mix/[01...06]  
main/m/config  
main/m/dyn  
main/m/dyn/filter  
main/m/insert  
main/m/eq  
main/m/eq[1...6]  
main/m/mix  
main/m/mix/[01...06]

dca/[1...8]  
dca/[1...8]/config

fx/[1...8]  
fx/[1...8]/source  
fx/[1...8]/par

outputs/main/[01...16]

```

outputs/main/[01...16]/delay
outputs/aux/[01...06]
outputs/p16/[01...16]
outputs/p16/[01...16]/iQ
outputs/aes/[01...02]
outputs/rec/[01...02]

headamp/[000...127]

-show
-show/prepos
-show/prepos/current

-show/showfile
-show/showfile/inputs
-show/showfile/mxsends
-show/showfile/mxbuses
-show/showfile/console
-show/showfile/chan16
-show/showfile/chan32
-show/showfile/return
-show/showfile/buses
-show/showfile/lrmtxaca
-show/showfile/effects

-show/showfile/cue
-show/showfile/cue/[000...099]
-show/showfile/cue/[000...099]/numb
-show/showfile/cue/[000...099]/name
-show/showfile/cue/[000...099]/skip
-show/showfile/cue/[000...099]/scene
-show/showfile/cue/[000...099]/bit
-show/showfile/cue/[000-099]/miditype
-show/showfile/cue/[000-099]/midichan
-show/showfile/cue/[000-099]/midipara1
-show/showfile/cue/[000-099]/midipara2

-show/showfile/scene
-show/showfile/scene/[000...099]
-show/showfile/scene/[000...099]/name
-show/showfile/scene/[000...099]/notes
-show/showfile/scene/[000...099]/safes
-show/showfile/scene/[000...099]/hasdata

-show/showfile/snippet
-show/showfile/snippet/[000...099]
-show/showfile/snippet/[000...099]/name
-show/showfile/snippet/[000...099]/eventtyp
-show/showfile/snippet/[000...099]/channels
-show/showfile/snippet/[000...099]/auxbuses
-show/showfile/snippet/[000...099]/maingrps
-show/showfile/snippet/[000...099]/hasdata

-libs/ch
-libs/ch/[001-100]
-libs/ch/[001-100]/pos
-libs/ch/[001-100]/name
-libs/ch/[001-100]/flags

```



```
-libs/ch/[001-100]/hasdata

-libs/fx
-libs/fx/[001-100]
-libs/fx/[001-100]/pos
-libs/fx/[001-100]/name
-libs/fx/[001-100]/flags
-libs/fx/[001-100]/hasdata

-libs/r
-libs/r/[001-100]
-libs/r/[001-100]/pos
-libs/r/[001-100]/name
-libs/r/[001-100]/flags
-libs/r/[001-100]/hasdata

-prefs
-prefs/style
-prefs/bright
-prefs/lcdcont
-prefs/ledbright
-prefs/lamp
-prefs/lampon
-prefs/clockrate
-prefs/clocksource
-prefs/confirm_general
-prefs/confirm_overwrite
-prefs/confirm_sceneload
-prefs/viewrtn
-prefs/selffollowsbank
-prefs/sceneadvance
-prefs/safe_masterlevels
-prefs/haflags
-prefs/autosel
-prefs/show_control
-prefs/clockmode
-prefs/hardmute
-prefs/dcamute
-prefs/invertmute

-prefs/rta
-prefs/rta/visibility
-prefs/rta/gain
-prefs/rta/autogain
-prefs/rta/source
-prefs/rta/pos
-prefs/rta/mode
-prefs/rta/option
-prefs/rta/det
-prefs/rta/decay
-prefs/rta/peakhold

-prefs/ip
-prefs/ip/dhcp
-prefs/ip/addr
-prefs/ip/mask
-prefs/ip/gateway

-prefs/remote
```

-prefs/remote/enable  
-prefs/remote/protocol  
-prefs/remote/port  
-prefs/remote/ioenable  
  
-prefs/iQ  
-prefs/iQ/[01-16]  
  
-usb  
-usb/path  
-usb/title  
-usb/dir  
-usb/dirpos  
-usb/maxpos  
-usb/dir/[000...999]  
-usb/dir/[000...999]/name  
  
-action  
-action/recselect  
-action/newscreen  
-action/goscene  
-action/gocue  
-action/setip  
  
-stat  
-stat/selidx  
-stat/chfaderbank  
-stat/grpfaderbank  
-stat/sendsonfader  
-stat/...  
-stat/...  
-stat/keysolo  
-stat/userbank  
-stat/autosave  
-stat/lock  
-stat/usbmounted  
-stat/remote  
-stat/...  
-stat/rtamodegeq  
-stat/...  
-stat/rtageqpost  
-stat/rtasource  
-stat/xcardtype  
-stat/xcardsync  
-stat/geqonfdr  
-stat/geqpos  
  
-stat/solosw  
-stat/solosw/[01...72]  
  
-stat/screen  
-stat/screen/screen  
-stat/screen/mutegrp  
-stat/screen/utis  
  
-stat/screen/CHAN [/page]  
-stat/screen/METER [/page]  
-stat/screen/ROUTE [/page]  
-stat/screen/SETUP [/page]

```

-stat/screen/LIB [/page]
-stat/screen/FX [/page]
-stat/screen/MON [/page]
-stat/screen/USB [/page]
-stat/screen/SCENE [/page]
-stat/screen/ASSIGN [/page]
-stat/screen/USER [/page]

-stat/tape
-stat/tape/state
-stat/tape/file
-stat/tape/etime
-stat/tape/rtime

```

**Note/bug:** the response from the Server is “*node...*” and not “/*node...*” as one could expect. This is not OSC compliant.

### Renew (/renew) data

#### Renew data

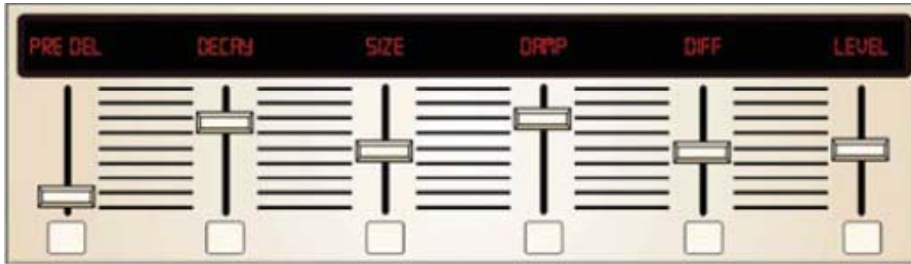
Path	Type	Description
/renew	string	Element to be renewed (sent back to requestor) Ex: /meters/5 Returns messages containing a blob of 132 bytes following “meter/5”  /hidden/states Returns a blob of 200 bytes following “/hidden/states”  /hidden/solo Returns a blob of 344 bytes following “/hidden/solo”  /hidden/prefs Returns a blob of 88 bytes following “/hidden/prefs”

## Effects Parameters

This section describes the parameters' list, order, name, type and value range for the different effects available with the X32/M32. The parameters described here correspond to the up to 64 parameters that can follow a `/fx/[1..8]/par/[01..64]` message.

Parameters can be sent one by one, or combined in lists -alternating types as needed-, which is generally more efficient.

### Hall Reverb



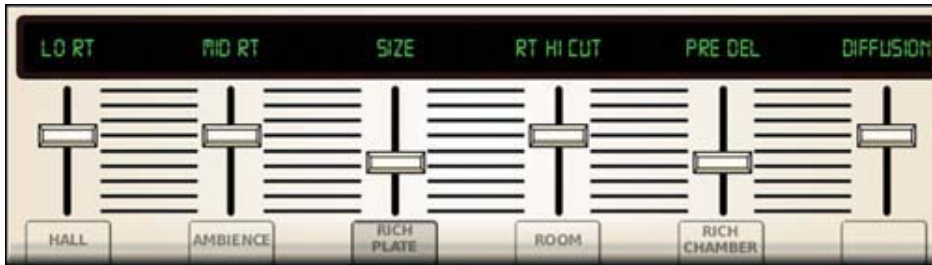
Effect Name	Parameters	Parameter Name	Type & Range
HALL (hall reverb)	ffffffffffff	Pre Delay	linf [0...200]
		Decay	logf [0.2...5]
		Size	linf [2...100]
		Damping	logf [1k...20k]
		Diffuse	linf [1...30]
		Level	linf [-12...+12]
		Lo Cut	logf [10...500]
		Hi Cut	logf [200...20k]
		Bass Multi	logf [0.5...2]
		Spread	linf [0...50]
		Shape	linf [0...250]
		Mod Speed	linf [0...100]

**Plate Reverb**



Effect Name	Parameters	Parameter Name	Type & Range
PLAT (plate reverb)	ffffffffffff	Pre Delay	linf [0...200]
		Decay	logf [0.2...10]
		Size	linf [2...100]
		Damping	logf [1k...20k]
		Diffuse	linf [1...30]
		Level	linf [-12...+12]
		Lo Cut	logf [10...500]
		Hi Cut	logf [200...20k]
		Bass Multi	logf [0.5...2]
		Xover	logf [10...500]
		Mod	linf [0...50]
		Mod Speed	linf [0...100]

**Ambiance Reverb**



Effect Name	Parameters	Parameter Name	Type & Range
AMBI (ambiance reverb)	ffffffffffff	Pre Delay	linf [0...200]
		Decay	logf [0.2...7.3]
		Size	linf [2...100]
		Damping	logf [1k...20k]
		Diffuse	linf [1...30]
		Level	linf [-12...+12]
		Lo Cut	logf [10...500]
		Hi Cut	logf [200...20k]
		Modulate	linf [0...100]
		Tail Gain	linf [0...100]

### Rich Plate Reverb



Effect Name	Parameters	Parameter Name	Type & Range
RPLT (rich plate reverb)	f f f f f f f f f f f f f f f f f f	Pre Delay	linf [0...200]
		Decay	logf [0.3...29]
		Size	linf [4...39]
		Damping	logf [1k...20k]
		Diffuse	linf [1...30]
		Level	linf [-12...+12]
		Lo Cut	logf [10...500]
		Hi Cut	logf [200...20k]
		Bass Multi	logf [0.25...4]
		Spread	linf [0...50]
		Attack	linf [0...100]
		Spin	linf [0...100]
		Echo L	linf [0...1200]
		Echo R	linf [0...1200]
Echo Feed L	linf [-100...+100]		
Echo Feed L	linf [-100...+100]		

### Room Reverb

Effect Name	Parameters	Parameter Name	Type & Range
ROOM (room reverb)	f f f f f f f f f f f f f f f f f f	Pre Delay	linf [0...200]
		Decay	logf [0.3...29]
		Size	linf [4...72]
		Damping	logf [1k...20k]
		Diffuse	linf [1...30]
		Level	linf [-12...+12]
		Lo Cut	logf [10...500]
		Hi Cut	logf [200...20k]
		Bass Multi	logf [0.25...4]
		Spread	linf [0...50]
		Shape	linf [0...250]
		Spin	linf [0...100]
		Echo L	linf [0...1200]
		Echo R	linf [0...1200]
		Echo Feed L	linf [-100...+100]
Echo Feed L	linf [-100...+100]		

### Chamber Reverb

Effect Name	Parameters	Parameter Name	Type & Range
CHAM (chamber reverb)	fffffffffffffffffff	Pre Delay	linf [0...200]
		Decay	logf [0.3...29]
		Size	linf [4...72]
		Damping	logf [1k...20k]
		Diffuse	linf [1...30]
		Level	linf [-12...+12]
		Lo Cut	logf [10...500]
		Hi Cut	logf [200...20k]
		Bass Multi	logf [0.25...4]
		Spread	linf [0...50]
		Shape	linf [0...250]
		Spin	linf [0...100]
		Reflection L	linf [0...500]
		Reflection R	linf [0...500]
		Reflection Gain L	linf [0...100]
Reflection Gain R	linf [0...100]		

### Vintage Reverb



Effect Name	Parameters	Parameter Name	Type & Range
VREV (vintage reverb)	ffffiifffff	Pre Delay	linf [0...120]
		Decay	logf [0.3...4.5]
		Modulate	linf [0...10]
		Vintage	enum [OFF, ON]
		Position	enum [FRONT, REAR]
		Level	linf [-12...+12]
		Lo Cut	logf [10...500]
		Hi Cut	logf [200...20k]
		Lo Multiply	logf [0.5...2]
Hi Multiply	logf [0.25...1]		

### Vintage Room



Effect Name	Parameters	Parameter Name	Type & Range
VRM (vintage room)	ffffffffffffi	Reverb Delay	linf [0...20]
		Decay	logf [0.1...20]
		Size	linf [0...10]
		Density	linf [1...30]
		ER Level	linf [0...190]
		Level	linf [-12...+12]
		Lo Multiply	logf [0.1...10]
		Hi Multiply	logf [0.1...10]
		Lo Cut	logf [10...500]
		Hi Cut	logf [200...20k]
		ER Left	linf [0...10]
		ER Right	linf [0...10]
		Freeze	enum [OFF, ON]

### Gated Reverb



Effect Name	Parameters	Parameter Name	Type & Range
GATE (gated reverb)	fffffffffff	Pre Delay	linf [0...200]
		Decay	logf [140...1000]
		Attack	linf [0...30]
		Density	linf [1...30]
		Spread	linf [0...100]
		Level	linf [-12...+12]
		Lo Cut	logf [10...500]
		Hi Cut	logf [200...20k]
		Hi Shv Gain	linf [-30...0]
		Diffuse	linf [1...30]

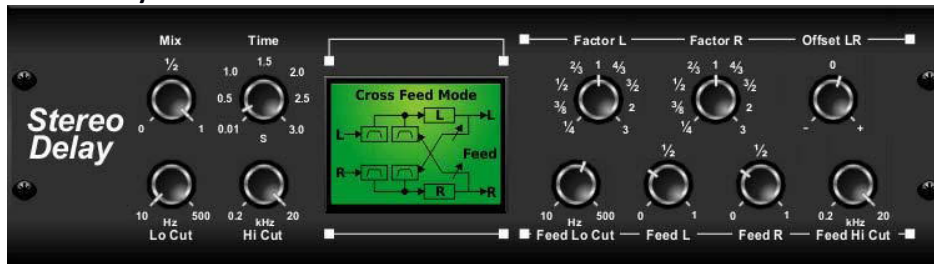


### Reverse Reverb



Effect Name	Parameters	Parameter Name	Type & Range
RVRS (reverse reverb)	ffffffffff	Pre Delay	linf [0...200]
		Decay	logf [140...1000]
		Rise	linf [0...50]
		Diffuse	linf [1...30]
		Spread	linf [1...100]
		Level	linf [-12...+12]
		Lo Cut	logf [10...500]
		Hi Cut	logf [200...20k]
		Hi Shv Gain	linf [-30...0]

### Stereo Delay



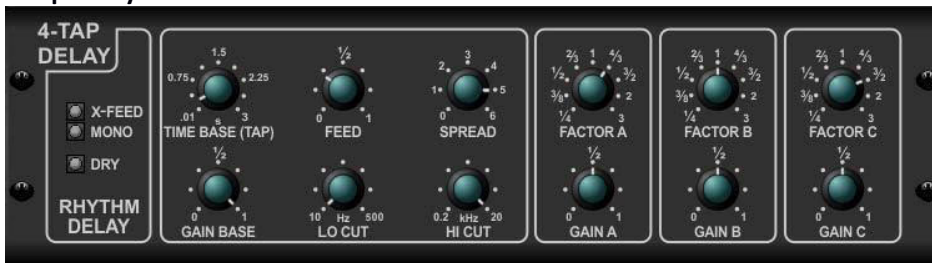
Effect Name	Parameters	Parameter Name	Type & Range
DLY (stereo delay)	ffiiiiiiiiii	Mix	linf [0...100]
		Time	linf [0...3000]
		Mode	enum [ST, X, M]
		Factor L	enum [1/4, 3/8, 1/2, 2/3, 1, 4/3, 3/2, 2, 3]
		Factor R	enum [1/4, 3/8, 1/2, 2/3, 1, 4/3, 3/2, 2, 3]
		Offset L/R	linf [-100...+100]
		Lo Cut	logf [10...500]
		Hi Cut	logf [200...20k]
		Feed Lo Cut	logf [10...500]
		Feed Left	linf [1...100]
		Feed Right	linf [1...100]
		Feed Hi Cut	logf [200...20k]

### 3-Tap Delay



Effect Name	Parameters	Parameter Name	Type & Range
3TAP (3-tap delay)	ffffffiffiffiii	Dry	linf [0...3000]
		Gain Base	linf [0...100]
		Pan Base	linf [-100...+100]
		Feedback	linf [0...100]
		Lo Cut	logf [10...500]
		Hi Cut	logf [200...20k]
		Factor A	enum [1/4, 3/8, 1/2, 2/3, 1, 4/3, 3/2, 2, 3]
		Gain A	linf [0...100]
		Pan A	linf [-100...+100]
		Factor B	enum [1/4, 3/8, 1/2, 2/3, 1, 4/3, 3/2, 2, 3]
		Gain B	linf [0...100]
		Pan B	linf [-100...+100]
		Cross Feed	enum [OFF, ON]
		Mono	enum [OFF, ON]
Dry	enum [OFF, ON]		

### 4-Tap Delay



Effect Name	Parameters	Parameter Name	Type & Range
4TAP (4-tap delay)	ffffffifififiii	Time	linf [1...3000]
		Gain Base	linf [0...100]
		Feedback	linf [0...100]
		Lo Cut	logf [10...500]
		Hi Cut	logf [200...20k]
		Spread	linf [0...6]
		Factor A	enum [1/4, 3/8, 1/2, 2/3, 1, 4/3, 3/2, 2, 3]
		Gain A	linf [0...100]
		Factor B	enum [1/4, 3/8, 1/2, 2/3, 1, 4/3,

			3/2, 2, 3]
		Gain B	linf [0...100]
		Factor C	enum [1/4, 3/8, 1/2, 2/3, 1, 4/3, 3/2, 2, 3]
		Gain C	linf [0...100]
		Cross Feed	enum [OFF, ON]
		Mono	enum [OFF, ON]
		Dry	enum [OFF, ON]

### Stereo Chorus



Effect Name	Parameters	Parameter Name	Type & Range
CRS (stereo chorus)	ffffffffffff	Speed	logf [0.05...5]
		Depth L	linf [0...100]
		Depth R	linf [0...100]
		Delay L	logf [0.5...20]
		Delay R	logf [0.5...20]
		Mix	linf [0...100]
		Lo Cut	logf [10...500]
		Hi Cut	logf [200...20k]
		Phase	linf [0...180]
		Wave	linf [0...100]
		Spread	linf [0...100]

### Stereo Flanger

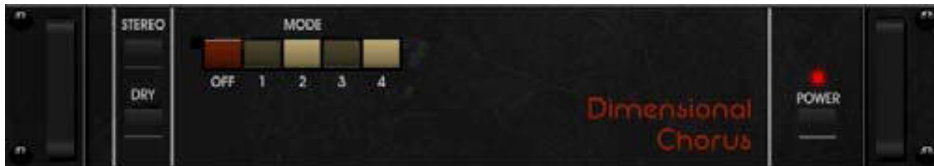
Effect Name	Parameters	Parameter Name	Type & Range
FLNG (stereo flanger)	ffffffffffff	Speed	logf [0.05...5]
		Depth L	linf [0...100]
		Depth R	linf [0...100]
		Delay L	logf [0.5...20]
		Delay R	logf [0.5...20]
		Mix	linf [0...100]
		Lo Cut	logf [10...500]
		Hi Cut	logf [200...20k]
		Phase	linf [0...180]
		Feed Lo Cut	logf [10...500]
		Feed Hi Cut	logf [200...20k]
		Feed	linf [-90...+90]

### Stereo Phaser



Effect Name	Parameters	Parameter Name	Type & Range
PHAS (stereo Phaser)	f f f f f f f f f f f f	Speed	logf [0.05...5]
		Depth	linf [0...100]
		Resonance	linf [0...80]
		Base	linf [0...50]
		Stages	linf [2...12]
		Mix	linf [0...100]
		Wave	linf [-50...+50]
		Phase	linf [0...180]
		Env. Modulation	linf [-100...+100]
		Attack	logf [10...1000]
		Hold	logf [1...2000]
		Release	logf [10...1000]

### Dimensional Chorus



Effect Name	Parameters	Parameter Name	Type & Range
DIMC (dimensional chorus)	i i i i i i i i	Active	enum [OFF, ON]
		Mode	enum [M, ST]
		Dry	enum [OFF, ON]
		Mode 1	enum [OFF, ON]
		Mode 2	enum [OFF, ON]
		Mode 3	enum [OFF, ON]
		Mode 4	enum [OFF, ON]

### Mood Filter



Effect Name	Parameters	Parameter Name	Type & Range
FILT (mood filter)	ffffififfffi	Speed	logf [0.05...20]
		Depth	linf [0...100]
		Resonance	linf [0...100]
		Base	logf [10...15000]
		Mode	enum [LP, HP, BP, NO]
		Mix	linf [0...100]
		Wave	enum [TRI, SIN, SAW, SAW-, RMP, SQU, RND]
		Phase	linf [0...180]
		Env. Modulation	linf [-100...+100]
		Attack	logf [10...250]
		Release	logf [10...500]
		Drive	linf [0...100]
		4 Pole	enum [OFF, ON]
		Side Chain	enum [OFF, ON]

### Rotary Speaker



Effect Name	Parameters	Parameter Name	Type & Range
ROTA (rotary speaker)	ffffffii	Lo Speed	logf [0.1...4]
		Hi Speed	logf [2...10]
		Accelerate	linf [0...100]
		Distance	linf [0...100]
		Balance	linf [-100...+100]
		Mix	linf [0...100]
		Stop	enum [OFF, ON]
		Slow	enum [OFF, ON]

### Tremolo / Panner



Effect Name	Parameters	Parameter Name	Type & Range
PAN (tremolo / panner)	f f f f f f f f f f	Speed	logf [0.05...4]
		Phase	linf [0...180]
		Wave	linf [-50...+50]
		Depth	linf [0...100]
		Env. Speed	linf [0...100]
		Env. Depth	linf [0...100]
		Attack	logf [10...1000]
		Hold	logf [1...2000]
		Release	logf [10...1000]

### Sub Octaver



Effect Name	Parameters	Parameter Name	Type & Range
SUB (suboctaver)	i i f f f i i f f f	Active	enum [OFF, ON]
		Range	enum [LO, MID, HI]
		Dry	linf [0...100]
		Octave -1	linf [0...100]
		Octave -2	linf [0...100]
		Active	enum [OFF, ON]
		Range	enum [LO, MID, HI]
		Dry	linf [0...100]
		Octave -1	linf [0...100]
		Octave -2	linf [0...100]

## Delay / Chamber



Effect Name	Parameters	Parameter Name	Type & Range
D/RV (delay / chamber)	fiffffffffff	Time	linf [1...3000]
		Pattern	enum [1/4, 1/3, 3/8, 1/2, 2/3, 3/4, 1, 1/4X, 1/3X, 3/8X, 1/2X, 2/3X, 3/4X,1X]
		Feed Hi Cut	logf [1000...20000]
		Feedback	linf [0...100]
		Cross Feed	linf [0...100]
		Balance	linf [-100...+100]
		Pre Delay	linf [0...200]
		Decay	logf [0.1...5]
		Size	linf [2...100]
		Damping	logf [1000...20000]
		Lo Cut	logf [10...500]
Mix	linf [0...100]		

## Delay / Chorus



Effect Name	Parameters	Parameter Name	Type & Range
D/CR (delay / chorus)	fiffffffffff	Time	linf [1...3000]
		Pattern	enum [1/4, 1/3, 3/8, 1/2, 2/3, 3/4, 1, 1/4X, 1/3X, 3/8X, 1/2X, 2/3X, 3/4X,1X]
		Feed Hi Cut	logf [1000...20000]
		Feedback	linf [0...100]
		Cross Feed	linf [0...100]
		Balance	linf [-100...+100]
		Speed	logf [0.05...4]
		Depth	linf [0...100]
		Delay	logf [0.5...50]
		Phase	linf [0...180]
		Wave	linf [0...100]
Mix	linf [0...100]		

## Delay /Flanger



Effect Name	Parameters	Parameter Name	Type & Range
D/FL (delay / flanger)	fiffffffffff	Time	linf [1...3000]
		Pattern	enum [1/4, 1/3, 3/8, 1/2, 2/3, 3/4, 1, 1/4X, 1/3X, 3/8X, 1/2X, 2/3X, 3/4X, 1X]
		Feed Hi Cut	logf [1000...20000]
		Feedback	linf [0...100]
		Cross Feed	linf [0...100]
		Balance	linf [-100...+100]
		Speed	logf [0.05...4]
		Depth	linf [0...100]
		Delay	logf [0.5...20]
		Phase	linf [0...180]
		Feed	linf [-90...+90]
Mix	linf [0...100]		

## Chorus / Chamber



Effect Name	Parameters	Parameter Name	Type & Range
CR/R (chorus / chamber)	ffffffffffff	Speed	logf [0.05...4]
		Depth	linf [0...100]
		Delay	logf [0.5...50]
		Phase	linf [0...180]
		Wave	linf [0...100]
		Balance	linf [-100...+100]
		Pre Delay	linf [0...200]
		Decay	logf [0.1...5]
		Size	linf [2...100]
		Damping	logf [1k...20k]
		Lo Cut	logf [10...500]
Mix	linf [0...100]		



## Flanger / Chamber



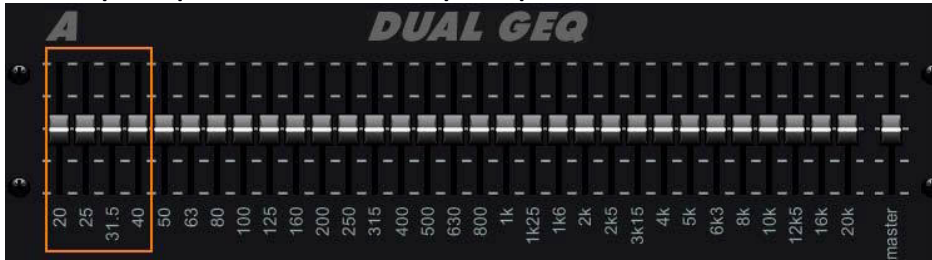
Effect Name	Parameters	Parameter Name	Type & Range
FL/R (flanger / chamber)	f f f f f f f f f f f f	Speed	logf [0.05...4]
		Depth	linf [0...100]
		Delay	logf [0.5...20]
		Phase	linf [0...180]
		Feed	linf [-90...+90]
		Balance	linf [-100...+100]
		Pre Delay	linf [0...200]
		Decay	logf [0.1...5]
		Size	linf [2...100]
		Damping	logf [1k...20k]
		Lo Cut	logf [10...500]
		Mix	linf [0...100]

## Modulation Delay



Effect Name	Parameters	Parameter Name	Type & Range
MODD (modulation delay)	f i f f f f f i i f f f f f	Time	linf [1...3000]
		Delay	enum [1, 1/2, 2/3, 3/2]
		Feed	linf [0...100]
		Lo Cut	logf [10...500]
		Hi Cut	logf [200...20k]
		Depth Rate	linf [0...100]
		Rate	logf [0.05...10]
		Setup	enum [PAR, SER]
		Type	enum [AMB, CLUB, HALL]
		Decay	linf [1...10]
		Damping	logf [1k...20k]
		Balance	linf [-100...+100]
		Mix	linf [0...100]

### Dual Graphic Equalizer / True Dual Graphic Equalizer



Effect Name	Parameters	Parameter Name	Type & Range
GEQ2 (dual graphic eq)	64 f	31 x Eq Level A	linf [-15...+15]
		Master Level A	linf [-15...+15]
TEQ2 (true dual graphic eq)		31 x Eq Level B	linf [-15...+15]
	Master Level B	linf [-15...+15]	

### Graphic Equalizer / True Graphic Equalizer



Effect Name	Parameters	Parameter Name	Type & Range
GEQ (stereo graphic eq)	32 f	31 x Eq Level L/R	linf [-15...+15]
		Master Level L/R	linf [-15...+15]
TEQ (true stereo graphic eq)			

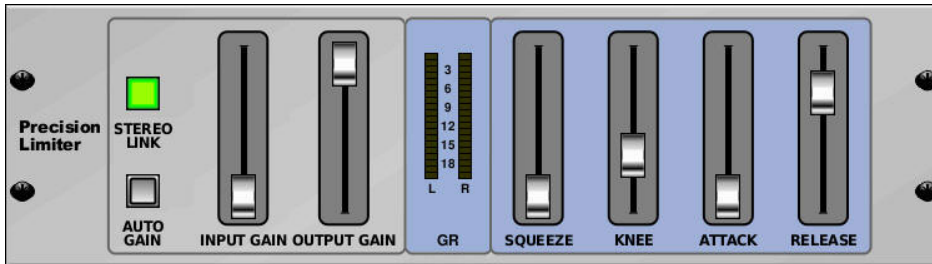
### Stereo / Dual De-Esser



Effect Name	Parameters	Parameter Name	Type & Range
DES (stereo deesser)	ffffii	Lo Band L	linf [0...50]
		Hi Band L	linf [0...50]
		Lo Band R	linf [0...50]
		Hi Band R	linf [0...50]
		Voice	enum [FEM / MALE]
		Mode	enum [ST / M/S]

Effect Name	Parameters	Parameter Name	Type & Range
DES2 (dual deesser)	ffffii	Lo Band A	linf [0...50]
		Hi Band A	linf [0...50]
		Lo Band B	linf [0...50]
		Hi Band B	linf [0...50]
		Voice A	enum [FEM / MALE]
		Voice B	enum [FEM / MALE]

### Precision Limiter



Effect Name	Parameters	Parameter Name	Type & Range
LIM (precision limiter)	ffffffii	Input Gain	linf [0...18]
		Out Gain	linf [-18...+18]
		Squeeze	linf [0...100]
		Knee	linf [0...10]
		Attack	logf 0.05...1]
		Release	logf [20...2000]
		Stereo Link	enum [OFF, ON]
		Auto Gain	enum [OFF, ON]

### Stereo / Dual Program EQ



Effect Name	Parameters	Parameter Name	Type & Range
P1A (stereo program eq)	iffiffififii	Active	enum [OFF, ON]
		Gain	linf [-12...+12]
		Lo Boost	linf [0...10]
		Lo Freq	enum [0, 30, 60, 100]
		Mid Width	linf [0...10]
		Mid Boost	linf [0...10]
		Mid Freq	enum [3k, 4k, 5k, 8k, 10k, 12k, 16k]
		Hi Attenuation	linf [0...10]
		Hi Freq	enum [5k, 10k, 20k]
Transformer	enum [OFF, ON]		

Effect Name	Parameters	Parameter Name	Type & Range
P1A2 (dual program eq)	iffiffififiiiiffiffififii	Active A	enum [OFF, ON]
		Gain A	linf [-12...+12]
		Lo Boost A	linf [0...10]
		Lo Freq A	enum [0, 30, 60, 100]
		Mid Width A	linf [0...10]
		Mid Boost A	linf [0...10]
		Mid Freq A	enum [3k, 4k, 5k, 8k, 10k, 12k, 16k]
		Hi Attenuation A	linf [0...10]
		Hi Freq A	enum [5k, 10k, 20k]
		Transformer A	enum [OFF, ON]
		Active B	enum [OFF, ON]
		Gain B	linf [-12...+12]
		Lo Boost B	linf [0...10]
		Lo Freq B	enum [0, 30, 60, 100]
		Mid Width B	linf [0...10]
		Mid Boost B	linf [0...10]
		Mid Freq B	enum [3k, 4k, 5k, 8k, 10k, 12k, 16k]
		Hi Attenuation B	linf [0...10]
Hi Freq B	enum [5k, 10k, 20k]		
Transformer B	enum [OFF, ON]		

## Stereo / Dual Midrange EQ



Effect Name	Parameters	Parameter Name	Type & Range
PQ5 (stereo midrange eq)	ififififi	Active	enum [OFF, ON]
		Gain	linf [-12...+12]
		Lo Freq	enum [200, 300, 500, 700, 1000]
		Lo Boost	linf [0...10]
		Mid Freq	enum [200, 300, 500, 700, 1k, 1k5, 2k, 3k, 4k, 5k, 7k]
		Mid Boost	linf [0...10]
		Hi Freq	enum [1k5, 2k, 3k, 4k, 5k]
		Hi Boost	linf [0...10]
		Transformer	enum [OFF, ON]

Effect Name	Parameters	Parameter Name	Type & Range
PQ5S (dual midrange eq)	ififififiififififi	Active A	enum [OFF, ON]
		Gain A	linf [-12...+12]
		Lo Freq A	enum [200, 300, 500, 700, 1000]
		Lo Boost A	linf [0...10]
		Mid Freq A	enum [200, 300, 500, 700, 1k, 1k5, 2k, 3k, 4k, 5k, 7k]
		Mid Boost A	linf [0...10]
		Hi Freq A	enum [1k5, 2k, 3k, 4k, 5k]
		Hi Boost A	linf [0...10]
		Transformer A	enum [OFF, ON]
		Active B	enum [OFF, ON]
		Gain B	linf [-12...+12]
		Lo Freq B	enum [200, 300, 500, 700, 1000]
		Lo Boost B	linf [0...10]
		Mid Freq B	enum [200, 300, 500, 700, 1k, 1k5, 2k, 3k, 4k, 5k, 7k]
		Mid Boost B	linf [0...10]
		Hi Freq B	enum [1k5, 2k, 3k, 4k, 5k]
		Hi Boost B	linf [0...10]

**Stereo / Dual Combinator**



Effect Name	Parameters	Parameter Name	Type & Range
CMB (stereo combinator)	iiiiifififiifffiffiffiffiffii	Active	enum [OFF, ON]
		Band Solo	enum [OFF, Bd1, Bd2, Bd3, Bd4, Bd5]
		Mix	linf [0...100]
		Attack	linf [0...19]
		Release	logf [20...3000]
		Autorelease	enum [OFF, ON]
		SBC speed	linf [0...10]
		SBC ON	enum [OFF, ON]
		Xover	linf [-50...+50]
		Xover Slope	enum [12, 48]
		Ratio	enum [1.1, 1.2, 1.3, 1.5, 1.7, 2, 2.5, 3, 3.5, 4, 5, 7, 10, LIM]
		Threshold	linf [-40...0]
		Gain	linf [-10...+10]
		Band 1 Threshold	linf [-10...+10]
		Band 1 Gain	linf [-10...+10]
		Band 1 Lock	enum [0, 1]
		Band 2 Threshold	linf [-10...+10]
		Band 2 Gain	linf [-10...+10]
		Band 2 Lock	enum [0, 1]
		Band 3 Threshold	linf [-10...+10]
		Band 3 Gain	linf [-10...+10]
		Band 3 Lock	enum [0, 1]
		Band 4 Threshold	linf [-10...+10]
		Band 4 Gain	linf [-10...+10]
		Band 4 Lock	enum [0, 1]
		Band 5 Threshold	linf [-10...+10]
		Band 5 Gain	linf [-10...+10]
		Band 5 Lock	enum [0, 1]
		Meter Mode	enum [GR, SBC, PEAK]

Effect Name	Parameters	Parameter Name	Type & Range
CMB2 (dual combinator)	ii f f f f i f i f i i f f f f i f f i f f i f f i f f i f f i ii f f f f i f i f i i f f f f i f f i f f i f f i f f i f f ii	Active A	enum [OFF, ON]
		Band Solo A	enum [OFF, Bd1, Bd2, Bd3, Bd4, Bd5]
		Mix A	linf [0...100]
		Attack A	linf [0...19]
		Release A	logf [20...3000]
		Autorelease A	enum [OFF, ON]
		SBC speed A	linf [0...10]
		SBC ON A	enum [OFF, ON]
		Xover A	linf [-50...+50]
		Xover Slope A	enum [12, 48]
		Ratio A	enum [1.1, 1.2, 1.3, 1.5, 1.7, 2, 2.5, 3, 3.5, 4, 5, 7, 10, LIM]
		Threshold A	linf [-40...0]
		Gain A	linf [-10...+10]
		Band 1 Threshold A	linf [-10...+10]
		Band 1 Gain A	linf [-10...+10]
		Band 1 Lock A	enum [0, 1]
		Band 2 Threshold A	linf [-10...+10]
		Band 2 Gain A	linf [-10...+10]
		Band 2 Lock A	enum [0, 1]
		Band 3 Threshold A	linf [-10...+10]
		Band 3 Gain A	linf [-10...+10]
		Band 3 Lock A	enum [0, 1]
		Band 4 Threshold A	linf [-10...+10]
		Band 4 Gain A	linf [-10...+10]
		Band 4 Lock A	enum [0, 1]
		Band 5 Threshold A	linf [-10...+10]
		Band 5 Gain A	linf [-10...+10]
		Band 5 Lock A	enum [0, 1]
		Meter Mode A	enum [GR, SBC, PEAK]
		Active B	enum [OFF, ON]
		Band Solo B	enum [OFF, Bd1, Bd2, Bd3, Bd4, Bd5]
		Mix B	linf [0...100]
		Attack B	linf [0...19]
		Release B	logf [20...3000]
		Autorelease B	enum [OFF, ON]
		SBC speed B	linf [0...10]
		SBC ON B	enum [OFF, ON]
		Xover B	linf [-50...+50]
		Xover Slope B	enum [12, 48]
		Ratio B	enum [1.1, 1.2, 1.3, 1.5, 1.7, 2, 2.5, 3, 3.5, 4, 5, 7, 10, LIM]
		Threshold B	linf [-40...0]
		Gain B	linf [-10...+10]
		Band 1 Threshold B	linf [-10...+10]
		Band 1 Gain B	linf [-10...+10]
		Band 1 Lock B	enum [0, 1]
		Band 2 Threshold B	linf [-10...+10]
		Band 2 Gain B	linf [-10...+10]
		Band 2 Lock B	enum [0, 1]

		Band 3 Threshold B	linf [-10...+10]
		Band 3 Gain B	linf [-10...+10]
		Band 3 Lock B	enum [0, 1]
		Band 4 Threshold B	linf [-10...+10]
		Band 4 Gain B	linf [-10...+10]
		Band 4 Lock B	enum [0, 1]
		Band 5 Threshold B	linf [-10...+10]
		Band 5 Gain B	linf [-10...+10]
		Band 5 Lock B	enum [0, 1]
		Meter Mode B	enum [GR, SBC, PEAK]

### Stereo / Dual Fair Compressor



Effect Name	Parameters	Parameter Name	Type & Range
FAC (stereo fair compressor)	iffffff	Active	enum [OFF, ON]
		Input Gain	linf [-20...+20]
		Threshold	linf [0...10]
		Time	linf [0...6]
		Bias	linf [0...100]
		Gain	linf [-18...6]
		Balance	linf [-100...+100]

Effect Name	Parameters	Parameter Name	Type & Range
FAC2 (dual fair compressor)  FAC1M (m/s fair compressor)	iffffffiffffff	Active	enum [OFF, ON]
		Input Gain	linf [-20...+20]
		Threshold	linf [0...10]
		Time	linf [0...6]
		Bias	linf [0...100]
		Gain	linf [-18...6]
		Balance	linf [-100...+100]
		Active	enum [OFF, ON]
		Input Gain	linf [-20...+20]
		Threshold	linf [0...10]
		Time	linf [0...6]
		Bias	linf [0...100]
		Gain	linf [-18...6]
		Balance	linf [-100...+100]



### Stereo / Dual Leisure Compressor



Effect Name	Parameters	Parameter Name	Type & Range
LEC (stereo leisure compressor)	iffif	Active	enum [OFF, ON]
		Gain	linf [0...100]
		Peak	linf [0...100]
		Mode	enum [COMP, LIM]
		Gain	linf [-18...6]

Effect Name	Parameters	Parameter Name	Type & Range
LEC2 (dual leisure compressor)	iffiffiffif	Active A	enum [OFF, ON]
		Gain A	linf [0...100]
		Peak A	linf [0...100]
		Mode A	enum [COMP, LIM]
		Gain A	linf [-18...6]
		Active B	enum [OFF, ON]
		Gain B	linf [0...100]
		Peak B	linf [0...100]
		Mode B	enum [COMP, LIM]
		Gain B	linf [-18...6]

### Edison EX1



Effect Name	Parameters	Parameter Name	Type & Range
EDI (edison ex1)	iiiffff	Active	enum [OFF, ON]
		Stereo Input	enum [ST / M/S]
		Stereo Output	enum [ST / M/S]
		ST Spread	linf [-50...+50]
		LMF Spread	linf [-50...+50]
		Balance	linf [-50...+50]
		Center Distance	linf [-50...+50]
		Out Gain	linf [-12...+12]

### Stereo / Dual Ultimo Compressor



Effect Name	Parameters	Parameter Name	Type & Range
ULC (stereo ultimo compressor)	iffffi	Active	enum [OFF, ON]
		Input Gain	linf [-48...0]
		Out Gain	linf [-48...0]
		Attack	linf [1...7]
		Release	linf [1...7]
		Ratio	enum [4, 8, 12, 20, ALL]

Effect Name	Parameters	Parameter Name	Type & Range
ULC2 (dual ultimo compressor)	iffffiiffffi	Active A	enum [OFF, ON]
		Input Gain A	linf [-48...0]
		Out Gain A	linf [-48...0]
		Attack A	linf [1...7]
		Release A	linf [1...7]
		Ratio A	enum [4, 8, 12, 20, ALL]
		Active B	enum [OFF, ON]
		Input Gain B	linf [-48...0]
		Out Gain B	linf [-48...0]
		Attack B	linf [1...7]
		Release B	linf [1...7]
		Ratio B	enum [4, 8, 12, 20, ALL]

### Sound Maxer



Effect Name	Parameters	Parameter Name	Type & Range
SON (sound maxer)	iffiff	Active A	enum [OFF, ON]
		Lo Contour A	linf [0...10]
		Process A	linf [0...10]
		Out Gain A	linf [-12...+12]
		Active B	enum [OFF, ON]
		Lo Contour B	linf [0...10]
		Out Gain B	linf [-12...+12]

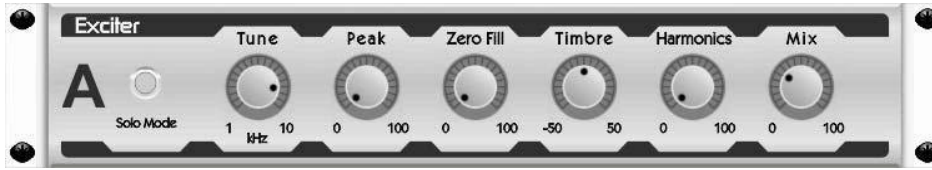
Stereo / Dual Enhancer



Effect Name	Parameters	Parameter Name	Type & Range
ENH (stereo enhancer)	f f f f f f f f f i	Out Gain	linf [-12...+12]
		Speed	linf [0...100]
		Bass Gain	linf [0...100]
		Bass Freq	linf [1...50]
		Mid Gain	linf [0...100]
		Mid Freq	linf [1...50]
		Hi Gain	linf [0...100]
		Hi Freq	linf [1...50]
		Solo	enum [OFF, ON]

Effect Name	Parameters	Parameter Name	Type & Range
ENH2 (dual enhancer)	f f f f f f f f f i f f f f f f f f f i	Out Gain A	linf [-12...+12]
		Speed A	linf [0...100]
		Bass Gain A	linf [0...100]
		Bass Freq A	linf [1...50]
		Mid Gain A	linf [0...100]
		Mid Freq A	linf [1...50]
		Hi Gain A	linf [0...100]
		Hi Freq A	linf [1...50]
		Solo A	enum [OFF, ON]
		Out Gain B	linf [-12...+12]
		Speed B	linf [0...100]
		Bass Gain B	linf [0...100]
		Bass Freq B	linf [1...50]
		Mid Gain B	linf [0...100]
		Mid Freq B	linf [1...50]
		Hi Gain B	linf [0...100]
		Hi Freq B	linf [1...50]

### Stereo / Dual Exciter



Effect Name	Parameters	Parameter Name	Type & Range
EXC (stereo exciter)	ffffffi	Tune	logf 1k...10k
		Peak	linf [0...100]
		Zero Fill	linf [0...100]
		Timbre	linf [-50...+50]
		Harmonics	linf [0...100]
		Mix	linf [0...100]
		Solo	enum [OFF, ON]

Effect Name	Parameters	Parameter Name	Type & Range
EXC2 (dual exciter)	ffffffiffffffffi	Tune A	logf 1k...10k
		Peak A	linf [0...100]
		Zero Fill A	linf [0...100]
		Timbre A	linf [-50...+50]
		Harmonics A	linf [0...100]
		Mix A	linf [0...100]
		Solo A	enum [OFF, ON]
		Tune B	logf 1k...10k
		Peak B	linf [0...100]
		Zero Fill B	linf [0...100]
		Timbre B	linf [-50...+50]
		Harmonics B	linf [0...100]
		Mix B	linf [0...100]
		Solo B	enum [OFF, ON]

### Stereo Imager



Effect Name	Parameters	Parameter Name	Type & Range
IMG (stereo imager)	fffffff	Balance	linf [-100...+100]
		Mono Pan	linf [-100...+100]
		Stereo Pan	linf [-100...+100]
		Shv Gain	linf [0...12]
		Shv Freq	logf [100...1000]
		Shv Q	logf [1...10]
		Out Gain	linf [-12...+12]

### Stereo / Dual Guitar Amp



Effect Name	Parameters	Parameter Name	Type & Range
AMP (stereo guitar amp)	ffffffffi	Preamp	linf [0...10]
		Buzz	linf [0...10]
		Punch	linf [0...10]
		Crunch	linf [0...10]
		Drive	linf [0...10]
		Low	linf [0...10]
		High	linf [0...10]
		Level	linf [0...10]
		Cabinet	enum [OFF, ON]

Effect Name	Parameters	Parameter Name	Type & Range
AMP2 (dual guitar amp)	ffffffffifffffffffi	Preamp A	linf [0...10]
		Buzz A	linf [0...10]
		Punch A	linf [0...10]
		Crunch A	linf [0...10]
		Drive A	linf [0...10]
		Low A	linf [0...10]
		High A	linf [0...10]
		Level A	linf [0...10]
		Cabinet A	enum [OFF, ON]
		Preamp B	linf [0...10]
		Buzz B	linf [0...10]
		Punch B	linf [0...10]
		Crunch B	linf [0...10]
		Drive B	linf [0...10]
		Low B	linf [0...10]
		High B	linf [0...10]
		Level B	linf [0...10]

Stereo / Dual Tube Stage



Effect Name	Parameters	Parameter Name	Type & Range
DRV (stereo tube stage)	f f f f f f f f f f	Drive	linf [0...100]
		Even Ear	linf [0...50]
		Odd Ear	linf [0...50]
		Gain	linf [-12...+12]
		Lo Cut	logf [20...200]
		Hi Cut	logf [4k...20k]
		Lo Gain	linf [-12...+12]
		Lo Freq	logf [50...400]
		Hi Gain	linf [-12...+12]
		Hi Freq	logf [1k...10k]

Effect Name	Parameters	Parameter Name	Type & Range
DRV2 (dual tube stage)	f f	Drive A	linf [0...100]
		Even Ear A	linf [0...50]
		Odd Ear A	linf [0...50]
		Gain A	linf [-12...+12]
		Lo Cut A	logf [20...200]
		Hi Cut A	logf [4k...20k]
		Lo Gain A	linf [-12...+12]
		Lo Freq A	logf [50...400]
		Hi Gain A	linf [-12...+12]
		Hi Freq A	logf [1k...10k]
		Drive B	linf [0...100]
		Even Ear B	linf [0...50]
		Odd Ear B	linf [0...50]
		Gain B	linf [-12...+12]
		Lo Cut B	logf [20...200]
		Hi Cut B	logf [4k...20k]
		Lo Gain B	linf [-12...+12]
		Lo Freq B	logf [50...400]
		Hi Gain B	linf [-12...+12]
		Hi Freq B	logf [1k...10k]

### Stereo / Dual Pitch Shifter



Effect Name	Parameters	Parameter Name	Type & Range
PIT (stereo pitch)	ffffff	Semitone	linf [-12...+12]
		Cent	linf [-50...+50]
		Delay	logf [1...100]
		Lo Cut	logf [10...500]
		Hi Cut	logf [2k...20k]
		Mix	linf [0...100]

Effect Name	Parameters	Parameter Name	Type & Range
PIT2 (dual pitch)	ffffffffffffff	Semitone A	linf [-12...+12]
		Cent A	linf [-50...+50]
		Delay A	logf [1...100]
		Lo Cut A	logf [10...500]
		Hi Cut A	logf [2k...20k]
		Mix A	linf [0...100]
		Semitone B	linf [-12...+12]
		Cent B	linf [-50...+50]
		Delay B	logf [1...100]
		Lo Cut B	logf [10...500]
		Hi Cut B	logf [2k...20k]
		Mix B	linf [0...100]

### Wave Designer

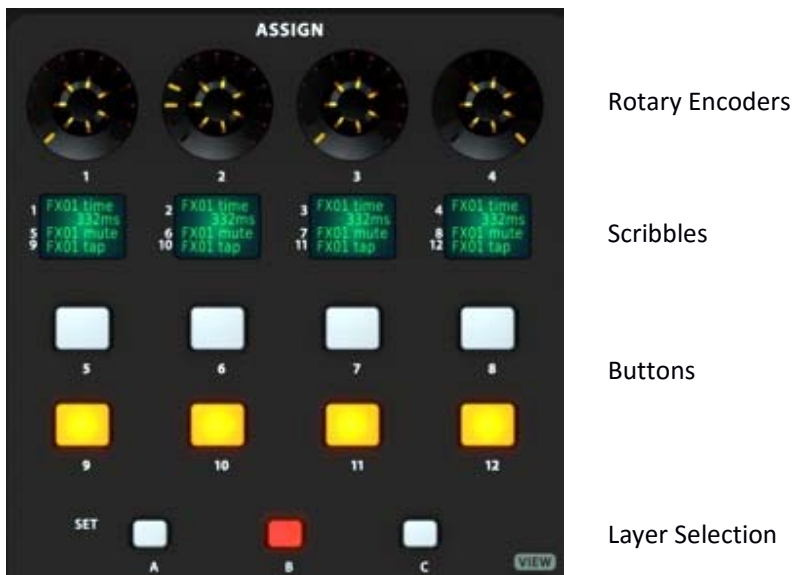


Effect Name	Parameters	Parameter Name	Type & Range
WAV (wave designer)	ffffff	Attack A	linf [-100...+100]
		Sustain A	linf [-100...+100]
		Gain A	linf [-24...+24]
		Attack B	linf [-100...+100]
		Sustain B	linf [-100...+100]
		Gain B	linf [-24...+24]

## User Definable Controls (ASSIGN Section)

This chapter describes the different settings and options linked to User Definable Controls a.k.a OSC command `/config/userctrl`.

The User Control section consists of 4 columns composed of a rotary encoder and two buttons. Encoders are numbered 1 to 4 and buttons are numbered 5 to 12. **Question: Are the 8 buttons of the X32/M32 Compact known as buttons 1 to 8 or 5 to 12, as on the X32/M32 Standard and Producer?**



A series of 3 buttons at the bottom of the section enables selecting one of 3 layers of user controls layers [A, B and C]. When addressing user controls to get or set data, the layer name and encoder or button number must be provided. A small LCD displays the functions the rotary encoders or buttons are assigned to.

### Notes:

There are no OSC commands to update the 4 scribbles of the assign section. This and the lack of full user assignable data limit what can be achieved with the assign section.

There are no OSC commands to set an OSC only mode for the assign section in the current implementation (FW 2.14).

There are no “NC” (normally closed) push-type buttons in the current implementation (FW 2.10). “NC” push buttons types can still be implemented but do require a feedback from the receiving program to simulate a normally closed push button.



## Rotary Encoders (X32/M32 Standard)

Note: TBD is this applies "as is" to Rack, Core, Compact and Producer

The data used to set encoder values is a string made of up to 7 characters. The first character (encoder assignment) selects the main functionality the encoder controls.

Target Type	Associated function
"-"	Not assigned
"F" Fader	Format "Fxx" xx: Channel/Bus "xx": 00...31: Channel 01 to Channel 32 32...39: Aux 01 to Aux 08 40...47: FX rtn 1L to FX rtn 4R 48...63: MixBus 01 to MixBus 16 64...69: Matrix 1 to Matrix 6 70 : Main LR 71 : Main M/C 72...79: DCA 1 to DCA 8
"P" Panorama	Format "Pxx" xx: Channel/Bus "xx": 00...31: Channel 01 to Channel 32 32...39: Aux 01 to Aux 08 40...47: FX rtn 1L to FX rtn 4R 48...63: MixBus 01 to MixBus 16 64...69: Matrix 1 to Matrix 6 70 : Main LR 71 : Main M/C 72...79: DCA 1 to DCA 8
"S" Send	Format "Sxxyy" xx: Channel/Bus, yy: Sends "xx": 00...31: Channel 01 to Channel 32 32...39: Aux 01 to Aux 08 40...47: FX rtn 1L to FX rtn 4R 48...63: MixBus 01 to MixBus 16 64...69: Matrix 1 to Matrix 6 70 : Main LR 71 : Main M/C 72...79: DCA 1 to DCA 8 "yy" - two characters representing a mix bus number: 00...15: MixBus 01 to MixBus 16
"X" Effect	Format "Xxyy" x: Effects Slot, yy: Paramater "x": 0...7: Effect 1 to Effect 8 "yy": 00...63: Effect parameter number 01 to 64
"M" Midi	Format "Mxyzzz" x" Message, yy: Channel, zzz: Value "x": C: Control Change N: Note P: Program Change "yy": 00...16: Midi channel number 01 to 16 "zzz": 000...127: Midi note or Midi value
"R" Remote	Format "Rxxx" xxx: Parameter "xxx" - three characters representing a remote assign:

	<p>000...007: remote 1 to remote 8  008: Jog</p>
"D" Selected Channel	<p>Format "Dx" x: Parameter  "x":</p> <ul style="list-style-type: none"> <li>@: Fader</li> <li>A: Gate threshold</li> <li>B: Gate range</li> <li>C: Gate attack</li> <li>D: Gate hold</li> <li>E: Gate release</li> <li>F: Dyn. threshold</li> <li>G: Dyn. ratio</li> <li>H: Dyn. knee</li> <li>I: Dyn. mgain</li> <li>J: Dyn. attack</li> <li>K: Dyn. hold</li> <li>L: Dyn. release</li> </ul>

## Buttons (X32/M32 Standard)

Note: TBD is this applies "as is" to Rack, Core, Compact and Producer

The data used to set buttons values is a string made of up to 7 characters. The first character (button assignment) selects the main functionality the button controls.

Button assignment	Associated function
"P" Jump to Page	Format: "Pxyyz", xx: Channel/Bus, y: Target, z: Page "y": 0: Channel "xx": 00...31: Channel 01 to Channel 32 32...39: Aux 01 to Aux 08 40...47: FX rtn 1L to FX rtn 4R 48...63: MixBus 01 to MixBus 16 64...69: Matrix 1 to Matrix 6 70 : Main LR 71 : Main M/C "z": 0: Home, 1: Config, 2: Gate, 3: Dyn, 4: EQ, 5: Mix, 6: Main, S: sends on faders 1: Meter "z": 0: Channel, 1: MixBus, 2: Aux/FX, 3: In/Out, 4: RTA 2: Route "z": 0: Home, 1: ANAOUT, 2: AUXOUT, 3: P16OUT, 4: CARDOUT, 5: AESAOOUT, 6: AESBOUT, 7: XLROUT 3: Setup "z": 0: Global, 1: Conf, 2: Remote, 3: Network, 4: Names, 5: Preamps, 6: Card 4: Lib "z": 0: Chan, 1: Effect, 2: Route 5: FX "z": 0: Home, 1: FX1, 2: FX2, 3: FX3, 4: FX4, 5: FX5, 6: FX6, 7: FX7, 8: FX8 "xx" : 00 to 04 for layer "-", 01 to layer 04 6: MON "z": 0: Monitor, 1: Talk A, 2: Talk B, 3: OSC 7: USB "z": 0: Home, 1: Config 8: Scene "z": 0: Home, 1: Scenes, 2: Bit, 3: ParSafe, 4: ChnSafe, 5: Midi 9: Assign "z": 0: Home, 1: Set A, 2: Set B, 3: Set C

Button assignment	Associated function
"O" Mutes	Format: "Oxx", xx: Channel/Bus "xx": 00...31: Channel 01 to Channel 32 32...39: Aux 01 to Aux 08 40...47: FX rtn 1L to FX rtn 4R 48...63: MixBus 01 to MixBus 16 64...69: Matrix 1 to Matrix 6 70 : Main LR 71 : Main M/C

	72...79: DCA 1 to DCA 8 80...85: Mute group 1 to 6
--	---

Button assignment	Associated function
"I" Inserts	Format: "Ixx", xx: Channel/Bus "xx": 00...31: Channel 01 to Channel 32 32...39: Aux 01 to Aux 08 40...47: FX rtn 1L to FX rtn 4R 48...63: MixBus 01 to MixBus 16 64...69: Matrix 1 to Matrix 6 70 : Main LR 71 : Main M/C

Button assignment	Associated function
"X" Effect Button	Format: "Xxyy", x: Effects Slot, yy: Parameter "x": 0...7: Effect 1 to Effect 8 Params "yy": 00...63: Parameter number

Button assignment	Associated function
"M" Midi Push	Format: "Mxyzzz", x: Message, yy: Channel, zzz: Value "x": C: Control Change N: Note P: Program Change "yy": 01...16: Channel Number "zzz": 000...127: Value

Button assignment	Associated function
"M" Midi Toggle	Format: "Mxyzzz", x: Message, yy: Channel, zzz: Value "x": c: Control Change n: Note "yy": 01...16: Channel Number "zzz": 000...127: Value

Button assignment	Associated function
"R" Remote	Format: "Rxxx", xxx: Parameter "xxx": 000...007: F1 to F8 008: Undo 009: Save 010: <Bank 011: >Bank 012: < CHN 013: >CHN 014...017: UP, DOWN, LEFT, RIGHT 018: STOP 019: PLAY 020: REC 021: FF

	022: <i>REW</i> 023: <i>MRK/RTZ</i> 024: <i>CYCLE</i> 025: <i>SCRUB</i> 026: <i>NDG/SHUT</i> 027: <i>DROP/IN</i> 028: <i>REP/OUT</i> 029: <i>CLI/OFF</i> 030: <i>READ</i> 031: <i>WRITE</i> 032: <i>TOUCH</i> 033: <i>TRIM</i> 034: <i>LATCH</i>
--	--

Button assignment	Associated function
"S" Cue Recall	Format: "S4xx", xx: Cue Number "xx": 00...99: <i>Cue number</i>

Button assignment	Associated function
"S" Scene Recall	Format: "S0xx", xx: Scene Number "xx": 00...99: <i>Scene number</i>

Button assignment	Associated function
"S" Snippet Recall	Format: "S2xx", xx: Snippet Number "xx": 00...99: <i>Snippet number</i>

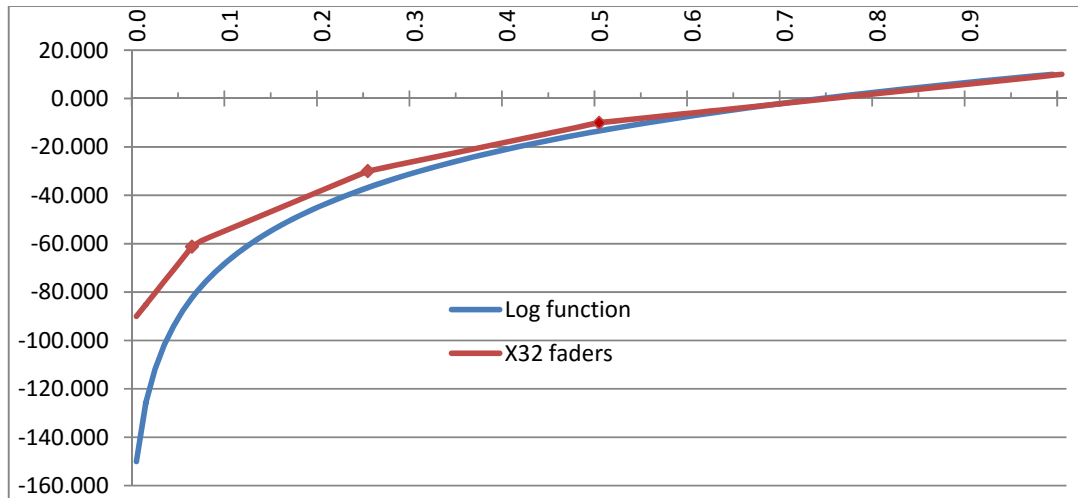
Button assignment	Associated function
"T" USB Recorder	Format: "Tx", xx: Function "x": 0: <i>Stop</i> 1: <i>Play</i> 2: <i>Record</i> 3: <i>Pause</i> 4: <i>Play/Stop</i> 5: <i>Play/Pause</i> 6: <i>Rec/Stop</i> 7: <i>Rec/Pause</i> 8: <i>Prev. Track</i> 9: <i>Next Track</i>

## Appendix – Converting X32 fader data to decibels and vice-versa

As mentioned earlier in this document, X32 faders implement a 4 linear functions approach with cross points at -10, -30, -60 dB to emulate the log function one can expect to manipulate volume data. Fader controls typically follow a  $\log_{10}$  function to match the human perception of loudness.

The volume ratio generic formula:  $dB\ value = 20 * \log (v2/v1)$  produces a response curve in blue, as below. On the other hand, X32 faders are using 4 different linear functions with increasing slopes to approximate the dB log transfer shape; the figure below shows the 4 different X32 line segments in red.

In both representations, 0db maps to 0.75 and 10dB maps to 1.0



The paragraphs below show a C-like conversion to go from float [0.0, 1.0] to dB [-90,+10], and vice-versa. This can be useful to map with other programs or tools used, or for programmers who need to match the float values returned by OSC functions to dB values in their programs.

```
// float to dB
// "f" represents OSC float data. f: [0.0, 1.0]
// "d" represents the dB float data. d:[-90, +10]
    if (f >= 0.5)          d = f * 40. - 30.;
    else if (f >= 0.25)    d = f * 80. - 50.;
    else if (f >= 0.0625) d = f * 160. - 70.;
    else if (f >= 0.0)     d = f * 480. - 90.;

// dB to float
// "d" represents the dB float data. d:[-90, +10]
// "f" represents OSC float data. f: [0.0, 1.0]
    if (d < -60.)         f = (d + 90.) / 480.;
    else if (d < -30.)    f = (d + 70.) / 160.;
    else if (d < -10.)   f = (d + 50.) / 80.;
    else if (d <= 10.)   f = (d + 30.) / 40.;
// Optionally round "f" to a X32 known value
    f = roundf(f * 1024) / 1024;
```

## Appendix – Scene data elements

The following table lists the control elements that can be found in a scene file. A scene file "*name.scn*" is typically 2070 lines (2054 in FW versions prior to 2.12) of editable data representing the working state of the X32/M32 and controlling almost all X32/M32 parameters.

<i>config/chlink</i>	<i>ch/[01...32]/eq/[1...4]</i>	<i>mtx/[01...06]/eq</i>
<i>config/auxlink</i>	<i>ch/[01...32]/mix</i>	<i>mtx/[01...06]/eq[1...6]</i>
<i>config/fxlink</i>	<i>ch/[01...32]/mix/[01...16]</i>	<i>mtx/[01...06]/mix</i>
<i>config/buslink</i>	<i>ch/[01...32]/grp</i>	<i>main/st/config</i>
<i>config/mtxlink</i>	<i>auxin/[01...08]/config</i>	<i>main/st/dyn</i>
<i>config/mute</i>	<i>auxin/[01...08]/preamp</i>	<i>main/st/dyn/filter</i>
<i>config/linkcfg</i>	<i>auxin/[01...08]/eq</i>	<i>main/st/insert</i>
<i>config/mono</i>	<i>auxin/[01...08]/eq/[1...4]</i>	<i>main/st/eq</i>
<i>config/solo</i>	<i>auxin/[01...08]/mix</i>	<i>main/st/eq[1...6]</i>
<i>config/talk</i>	<i>auxin/[01...08]/mix/[01...16]</i>	<i>main/st/mix</i>
<i>config/talk/A</i>	<i>auxin/[01...08]/grp</i>	<i>main/st/mix/[01...06]</i>
<i>config/talk/B</i>	<i>fxrtn/[01...08]/config</i>	<i>main/m/config</i>
<i>config/osc</i>	<i>fxrtn/[01...08]/eq</i>	<i>main/m/dyn</i>
<i>config/routing/IN</i>	<i>fxrtn/[01...08]/eq[1...4]</i>	<i>main/m/dyn/filter</i>
<i>config/routing/AES50A</i>	<i>fxrtn/[01...08]/mix</i>	<i>main/m/insert</i>
<i>config/routing/AES50B</i>	<i>fxrtn/[01...08]/mix/[01...16]</i>	<i>main/m/eq</i>
<i>config/routing/CARD</i>	<i>fxrtn/[01...08]/grp</i>	<i>main/m/eq[1...6]</i>
<i>config/routing/OUT</i>	<i>bus/[01...16]/config</i>	<i>main/m/mix</i>
<i>config/userctrl/{A,B,C}</i>	<i>bus/[01...16]/dyn</i>	<i>main/m/mix/[01...06]</i>
<i>config/userctrl/{A,B,C}/enc</i>	<i>bus/[01...16]/dyn/filter</i>	<i>dca/[1...8]</i>
<i>config/userctrl/{A,B,C}/btn</i>	<i>bus/[01...16]/insert</i>	<i>dca/[1...8]/config</i>
<i>config/tape</i>	<i>bus/[01...16]/eq</i>	<i>fx/[1...8]</i>
<i>ch/[01...32]/config</i>	<i>bus/[01...16]/eq[1...6]</i>	<i>fx/[1...8]/source</i>
<i>ch/[01...32]/delay</i>	<i>bus/[01...16]/mix</i>	<i>fx/[1...8]/par</i>
<i>ch/[01...32]/preamp</i>	<i>bus/[01...16]/mix/[01...06]</i>	<i>outputs/main/[01...16]</i>
<i>ch/[01...32]/gate</i>	<i>bus/[01...16]/grp</i>	<i>outputs/main/[01...16]/delay</i>
<i>ch/[01...32]/gate/filter</i>	<i>mtx/[01...06]/config</i>	<i>outputs/aux/[01...06]</i>
<i>ch/[01...32]/dyn</i>	<i>mtx/[01...06]/preamp</i>	<i>outputs/p16/[01...16]</i>
<i>ch/[01...32]/dyn/filter</i>	<i>mtx/[01...06]/dyn</i>	<i>outputs/p16/[01...16]/iQ<sup>25</sup></i>
<i>ch/[01...32]/insert</i>	<i>mtx/[01...06]/dyn/filter</i>	<i>outputs/aes/[01...02]</i>
<i>ch/[01...32]/eq</i>	<i>mtx/[01...06]/insert</i>	<i>outputs/rec/[01...02]</i>
		<i>headamp/[000...127]</i>

A scene file starts with a line such as:

```
#2.1# "Scene name" "Scene note" %000000000 1
```

This first line contains the *name* and *note* associated to the scene and ends with the list of scene safes in the form of 8 [0/1] characters terminated by a 0: %000000000 and is terminated by a 1. It is then followed by (/node like commands) lines from the table above; they are followed by the parameters they control;

A line beginning with '#' is treated as a comment line.

**For example**, *ch/[01...32]/config* will be followed by 4 parameters, as in

```
/ch/01/config "Kick Drum" 3 YE 1
```

<sup>25</sup> Starting with FW version 2.12

These parameters respectively correspond to the name given to the channel (as displayed on the Channel scribble), the icon associated with the channel scribble, the channel scribble color, and the channel source. These are detailed in their order of appearance after the corresponding */node* commands, in this document and for the present case under the **Channel (/ch) data** chapter.



## Appendix – Snippet data elements

The table below lists the control elements that can be found in a snippet file. A snippet file "*name.snp*" is variable in size and made of text editable data representing a subset of X32/M32 parameters.

A snippet file starts with a line such as:

```
#2.1# "Snippet name" 31473663 1 66305 449 1
```

The 4 numerical parameters following the snippet *name* and followed by '1' are the *eventtyp*, *channels*, *auxbuses*, and *maingrps* filters respectively, saved/present in the file.

This first line is then followed by the lines (*/node* like commands) in the table below; they are followed by the parameters they control; a line beginning with '#' is treated as a comment line.

/fx/[1...8]	/outputs/aes/[01...02]	/fxrtn/[01...08]/mix/[01...16]
/fx/[1...8]/source	/headamp/[000...127]	/fxrtn/[01...08]/mix/mono
/fx/[1...8]/par	/ch/[01...32]/preamp	/fxrtn/[01...08]/mix/mlevel
/config/solo	/ch/[01...32]/delay	/bus/[01...16]/config
/config/talk	/ch/[01...32]/config	/bus/[01...16]/eq
/config/talk/A	/ch/[01...32]/eq	/bus/[01...16]/eq/[1...6]
/config/talk/B	/ch/[01...32]/eq/[1...4]	/bus/[01...16]/dyn
/config/routing/IN/1-8	/ch/[01...32]/gate	/mtx/[01...06]/config
/config/routing/IN/9-16	/ch/[01...32]/gate/filter	/mtx/[01...06]/eq
/config/routing/IN/17-24	/ch/[01...32]/dyn	/mtx/[01...06]/eq/[1...6]
/config/routing/IN/25-32	/ch/[01...32]/dyn/filter	/mtx/[01...06]/dyn
/config/routing/IN/AUX	/ch/[01...32]/insert	/mtx/[01...06]/dyn/filter
/config/routing/AES50A/1-8	/ch/[01...32]/grp	/mtx/[01...06]/insert
/config/routing/AES50A/9-16	/ch/[01...32]/mix/fader	/mtx/[01...06]/mix/fader
/config/routing/AES50A/17-24	/ch/[01...32]/mix/pan	/mtx/[01...06]/mix/on
/config/routing/AES50A/25-32	/ch/[01...32]/mix/on	/main/st/config
/config/routing/AES50A/33-40	/ch/[01...32]/mix/[01...16]	/main/st/eq
/config/routing/AES50A/41-48	/ch/[01...32]/mix/mono	/main/st/eq/[1...6]
/config/routing/AES50B/1-8	/ch/[01...32]/mix/mlevel	/main/st/dyn
/config/routing/AES50B/9-16	/auxin/[01...06]/preamp	/main/st/dyn/filter
/config/routing/AES50B/17-24	/auxin/[01...06]/config	/main/st/insert
/config/routing/AES50B/25-32	/auxin/[01...06]/eq	/main/st/mix/fader
/config/routing/AES50B/33-40	/auxin/[01...06]/eq/[1...4]	/main/st/mix/pan
/config/routing/AES50B/41-48	/auxin/[01...06]/grp	/main/st/mix/on
/config/routing/CARD/1-8	/auxin/[01...06]/mix/fader	/main/st/mix/[01...06]
/config/routing/CARD/9-16	/auxin/[01...06]/mix/pan	/main/m/config
/config/routing/CARD/17-24	/auxin/[01...06]/mix/on	/main/m/eq
/config/routing/CARD/25-32	/auxin/[01...06]/mix/[01...16]	/main/m/eq/[1...6]
/config/routing/OUT/1-4	/auxin/[01...06]/mix/mono	/main/m/dyn
/config/routing/OUT/5-8	/auxin/[01...06]/mix/mlevel	/main/m/dyn/filter
/config/routing/OUT/9-12	/fxrtn/[01...08]/config	/main/m/insert
/config/routing/OUT/13-16	/fxrtn/[01...08]/eq	/main/m/mix/fader
/outputs/main/[01...16]	/fxrtn/[01...08]/eq/[1...4]	/main/m/mix/on
/outputs/main/[01...16]/delay	/fxrtn/[01...08]/grp	/main/m/mix/[01...06]
/outputs/aux/[01...06]	/fxrtn/[01...08]/mix/fader	/dca/[1...8]/config
/outputs/p16/[01...16]	/fxrtn/[01...08]/mix/pan	/dca/[1...8]/fader
/outputs/p16/[01...16]/iQ <sup>26</sup>	/fxrtn/[01...08]/mix/on	/dca/[1...8]/on

<sup>26</sup> Starting with FW version 2.12

## Appendix – Channel, Library and Routing preset files data elements

The table below lists the control elements that can be found in a Channel, Library or Routing preset file. Preset files are variable in size and consist of subset of X32/M32 parameters in the form of editable text (*/node* like commands).

A preset file starts with a line such as:

```
#2.1# <pos> "Preset name" <type> %<flags> 1
```

Where

- <pos>: preset index value (can be chosen by the user)
- <type>: a type number:
  - Effects: used for sorting by type or by name and enable loading in FX slots [1...4] or [5...8]
  - Channels and Routing: unused
- <flags>: a list of 16 digits [0 or 1] representing:
  - Channels: if a channel section is present in the preset [*digits 0...7*] and whether it is active or not [*digits 8...15*] (see */-libs/ch[001-100]/flags* for details)
  - Effects and Routing: unused

This first line is then followed by the lines (*/node* like commands) in the table below; they are followed by the parameters they control; a line beginning with '#' is treated as a comment line.

Channel*	Effect*	Routing
<i>/config</i>	<i>/type</i>	<i>/config/routing/IN</i>
<i>/delay</i>	<i>/source</i>	<i>/config/routing/AES50A</i>
<i>/preamp</i>	<i>/par</i>	<i>/config/routing/AES50B</i>
<i>/gate</i>		<i>/config/routing/CARD</i>
<i>/gate/filter</i>		<i>/config/routing/OUT</i>
<i>/dyn</i>		<i>/outputs/main/[01...16]</i>
<i>/dyn/filter</i>		<i>/outputs/main/[01...16]/delay</i>
<i>/eq</i>		<i>/outputs/aux/[01...06]</i>
<i>/eq/[1...6]</i>		<i>/outputs/p16/[01...16]</i>
<i>/mix</i>		<i>/outputs/p16/[01...16]/iQ<sup>27</sup></i>
<i>/mix/[01...16]</i>		<i>/outputs/aes/[01...02]</i>
<i>/headamp/[000...127]</i>		

\* In the case of Channel and Effect types, the typical */node* header (i.e. */ch/nn/* or */fx/n/*) is not present as the file applies to a specific channel or effect.

<sup>27</sup> Starting with FW version 2.12

## Appendix – X32/M32 Icons

X32/M32 Icons<sup>28</sup> are numbered 01...74 and shown in the table below.

01		02		03		04		05		06	
07		08		09		10		11		12	
13		14		15		16		17		18	
19		20		21		22		23		24	
25		26		27		28		29		30	
31		32		33		34		35		36	
37		38		39		40		41		42	
43		44		45		46		47		48	
49		50		51		52		53		54	
55		56		57		58		59		60	
61		62		63		64		65		66	
67		68		69		70		71		72	
73		74									

<sup>28</sup> The icons have been extracted from the X32-edit application

## Appendix – OSC over MIDI Sysex commands

Additionally to Behringer’s document/note “X32 MIDI Implementation (06 May 2014)”<sup>29</sup> which provides an overview of the MIDI RX, TX and MIDI assignments applicable to X32/M32 systems, OSC commands can be sent to the device over MIDI, using Sysex messages. Make sure your MIDI connection or device will support sending SYSEX messages; some devices do not provide (full) SYSEX support.

The general format for sending OSC commands over MIDI Sysex is:

```
F0 00 20 32 32 <OSCtext> F7
```

with <OSCtext> being the OSC command in text hex format, and up to 39 kbytes in length.

Examples: (~ stands for the NULL character, \0; data within brackets are sent as 32 bits big endian values)

Setting channel 01 mute ON (muting the channel):

```
OSC: /ch/01/mix/on~~~,i~~[0]
```

```
OSC: /ch/01/mix/on~~~,s~~OFF
```

```
Sysex: F0 00 20 32 32 2F 63 68 2F 30 31 2F 6D 69 78 2F 6F 6E 20 4F 46 46 F7
```

Unmuting channel 01:

```
OSC: /ch/01/mix/on~~~,i~~[1]
```

```
OSC: /ch/01/mix/on~~~,s~~ON
```

```
Sysex: F0 00 20 32 32 2F 63 68 2F 30 31 2F 6D 69 78 2F 6F 6E 20 4F 4E F7
```

Setting channel 01, EQ 2 frequency to 1kHz (actually 1020Hz, due to known discrete values)

```
OSC: /ch/01/eq/2/f~~~,f~~[0.57]
```

```
Sysex: F0 00 20 32 32 2F 63 68 2F 30 31 2F 65 71 2F 32 2F 66 20 31 30 32 30 F7
```

Setting channel 01, dynamics Hold value to 100ms

```
OSC: /ch/01/dyn/hold~,f~~[0.74]
```

```
Sysex: F0 00 20 32 32 2F 63 68 2F 30 31 2F 64 79 6E 2F 68 6F 6C 64 20 31 30 30 F7
```

Setting User Assign Bank C, button 5 to send MIDI note 3 on MIDI channel 5, as a MIDI push command

```
OSC: /config/userctrl/C/btn/5~~~~,s~~MN05003~
```

```
Sysex: F0 00 20 32 32 2F 63 6F 6E 66 69 67 2F 75 73 65 72 63 74 72 6C 2F 43 2F 62 74  
6E 2F 35 20 4D 4E 30 35 30 30 33 F7
```

As a result, Bank C button 5 will generate the following two MIDI sequences: 94 03 7F ; 94 03 00

Please refer to the OSC commands descriptions in this document for command formats and applicable ranges for their respective parameters types and ranges, and tables in appendix for corresponding floating point data and X32/M32 known discrete values for different fields (EQ, Dynamics, Gate, etc).

<sup>29</sup> Available on Behringer web site, download section – MIDI Protocol

## Appendix – Frequency Table – 201 log scale frequency values – [20 Hz, 20 kHz]

The data is presented as [float, node value] couples

0.0000	20.0	0.2050	82.4	0.4100	339.6	0.6150	1k39	0.8200	5k76
0.0050	20.7	0.2100	85.3	0.4150	351.6	0.6200	1k44	0.8250	5k97
0.0100	21.4	0.2150	88.3	0.4200	363.9	0.6250	1k49	0.8300	6k18
0.0150	22.2	0.2200	91.4	0.4250	376.7	0.6300	1k55	0.8350	6k39
0.0200	23.0	0.2250	94.6	0.4300	390.0	0.6350	1k60	0.8400	6k62
0.0250	23.8	0.2300	98.0	0.4350	403.7	0.6400	1k66	0.8450	6k85
0.0300	24.6	0.2350	101.4	0.4400	417.9	0.6450	1k72	0.8500	7k09
0.0350	25.5	0.2400	105.0	0.4450	432.5	0.6500	1k78	0.8550	7k34
0.0400	26.4	0.2450	108.7	0.4500	447.7	0.6550	1k84	0.8600	7k60
0.0450	27.3	0.2500	112.5	0.4550	463.5	0.6600	1k91	0.8650	7k87
0.0500	28.3	0.2550	116.4	0.4600	479.8	0.6650	1k97	0.8700	8k14
0.0550	29.2	0.2600	120.5	0.4650	496.6	0.6700	2k04	0.8750	8k43
0.0600	30.3	0.2650	124.7	0.4700	514.1	0.6750	2k11	0.8800	8k73
0.0650	31.3	0.2700	129.1	0.4750	532.1	0.6800	2k19	0.8850	9k03
0.0700	32.4	0.2750	133.7	0.4800	550.8	0.6850	2k27	0.8900	9k35
0.0750	33.6	0.2800	138.4	0.4850	570.2	0.6900	2k34	0.8950	9k68
0.0800	34.8	0.2850	143.2	0.4900	590.2	0.6950	2k43	0.9000	10k02
0.0850	36.0	0.2900	148.3	0.4950	611.0	0.7000	2k51	0.9050	10k37
0.0900	37.2	0.2950	153.5	0.5000	632.5	0.7050	2k60	0.9100	10k74
0.0950	38.6	0.3000	158.9	0.5050	654.7	0.7100	2k69	0.9150	11k11
0.1000	39.9	0.3050	164.4	0.5100	677.7	0.7150	2k79	0.9200	11k50
0.1050	41.3	0.3100	170.2	0.5150	701.5	0.7200	2k89	0.9250	11k91
0.1100	42.8	0.3150	176.2	0.5200	726.2	0.7250	2k99	0.9300	12k33
0.1150	44.3	0.3200	182.4	0.5250	751.7	0.7300	3k09	0.9350	12k76
0.1200	45.8	0.3250	188.8	0.5300	778.1	0.7350	3k20	0.9400	13k21
0.1250	47.4	0.3300	195.4	0.5350	805.4	0.7400	3k31	0.9450	13k67
0.1300	49.1	0.3350	202.3	0.5400	833.7	0.7450	3k43	0.9500	14k15
0.1350	50.8	0.3400	209.4	0.5450	863.0	0.7500	3k55	0.9550	14k65
0.1400	52.6	0.3450	216.8	0.5500	893.4	0.7550	3k68	0.9600	15k17
0.1450	54.5	0.3500	224.4	0.5550	924.8	0.7600	3k81	0.9650	15k70
0.1500	56.4	0.3550	232.3	0.5600	957.3	0.7650	3k94	0.9700	16k25
0.1550	58.3	0.3600	240.5	0.5650	990.9	0.7700	4k08	0.9750	16k82
0.1600	60.4	0.3650	248.9	0.5700	1k02	0.7750	4k22	0.9800	17k41
0.1650	62.5	0.3700	257.6	0.5750	1k06	0.7800	4k37	0.9850	18k03
0.1700	64.7	0.3750	266.7	0.5800	1k09	0.7850	4k52	0.9900	18k66
0.1750	67.0	0.3800	276.1	0.5850	1k13	0.7900	4k68	0.9950	19k32
0.1800	69.3	0.3850	285.8	0.5900	1k17	0.7950	4k85	1.0000	20k00
0.1850	71.8	0.3900	295.8	0.5950	1k21	0.8000	5k02		
0.1900	74.3	0.3950	306.2	0.6000	1k26	0.8050	5k20		
0.1950	76.9	0.4000	317.0	0.6050	1k30	0.8100	5k38		
0.2000	79.6	0.4050	328.1	0.6100	1k35	0.8150	5k57		

## Appendix – Frequency Table – 121 log scale frequency values – [20 Hz, 20 kHz]

The data is presented as [float, node value] couples

0.0000	20.0	0.2333	100.2	0.4667	502.4	0.7000	2k51	0.9333	12k61
0.0083	21.2	0.2417	106.2	0.4750	532.1	0.7083	2k66	0.9417	13k36
0.0167	22.4	0.2500	112.5	0.4833	563.7	0.7167	2k82	0.9500	14k15
0.0250	23.8	0.2583	119.1	0.4917	597.1	0.7250	2k99	0.9583	14k99
0.0333	25.2	0.2667	126.2	0.5000	632.5	0.7333	3k16	0.9667	15k88
0.0417	26.7	0.2750	133.7	0.5083	669.9	0.7417	3k35	0.9750	16k82
0.0500	28.3	0.2833	141.6	0.5167	709.6	0.7500	3k55	0.9833	17k82
0.0583	29.9	0.2917	150.0	0.5250	751.7	0.7583	3k76	0.9917	18k88
0.0667	31.7	0.3000	158.9	0.5333	796.2	0.7667	3k99	1.0000	20k00
0.0750	33.6	0.3083	168.3	0.5417	843.4	0.7750	4k22		
0.0833	35.6	0.3167	178.3	0.5500	893.4	0.7833	4k47		
0.0917	37.7	0.3250	188.8	0.5583	946.3	0.7917	4k74		
0.1000	39.9	0.3333	200.0	0.5667	1k00	0.8000	5k02		
0.1083	42.3	0.3417	211.9	0.5750	1k06	0.8083	5k32		
0.1167	44.8	0.3500	224.4	0.5833	1k12	0.8167	5k63		
0.1250	47.4	0.3583	237.7	0.5917	1k19	0.8250	5k97		
0.1333	50.2	0.3667	251.8	0.6000	1k26	0.8333	6k32		
0.1417	53.2	0.3750	266.7	0.6083	1k33	0.8417	6k69		
0.1500	56.4	0.3833	282.5	0.6167	1k41	0.8500	7k09		
0.1583	59.7	0.3917	299.2	0.6250	1k49	0.8583	7k51		
0.1667	63.2	0.4000	317.0	0.6333	1k58	0.8667	7k96		
0.1750	67.0	0.4083	335.8	0.6417	1k68	0.8750	8k43		
0.1833	71.0	0.4167	355.7	0.6500	1k78	0.8833	8k93		
0.1917	75.2	0.4250	376.7	0.6583	1k88	0.8917	9k46		
0.2000	79.6	0.4333	399.1	0.6667	2k00	0.9000	10k02		
0.2083	84.3	0.4417	422.7	0.6750	2k11	0.9083	10k61		
0.2167	89.3	0.4500	447.7	0.6833	2k24	0.9167	11k24		
0.2250	94.6	0.4583	474.3	0.6917	2k37	0.9250	11k91		

**Appendix – Frequency Table – 101 log scale frequency values – [20 Hz, 400 Hz]**

The data is presented as [float, node value] couples

0.0000	20	0.3000	49	0.6000	121	0.9000	296
0.0100	21	0.3100	51	0.6100	124	0.9100	305
0.0200	21	0.3200	52	0.6200	128	0.9200	315
0.0300	22	0.3300	54	0.6300	132	0.9300	324
0.0400	23	0.3400	55	0.6400	136	0.9400	334
0.0500	23	0.3500	57	0.6500	140	0.9500	344
0.0600	24	0.3600	59	0.6600	144	0.9600	355
0.0700	25	0.3700	61	0.6700	149	0.9700	366
0.0800	25	0.3800	62	0.6800	153	0.9800	377
0.0900	26	0.3900	64	0.6900	158	0.9900	388
0.1000	27	0.4000	66	0.7000	163	1.0000	400
0.1100	28	0.4100	68	0.7100	168		
0.1200	29	0.4200	70	0.7200	173		
0.1300	30	0.4300	73	0.7300	178		
0.1400	30	0.4400	75	0.7400	184		
0.1500	31	0.4500	77	0.7500	189		
0.1600	32	0.4600	79	0.7600	195		
0.1700	33	0.4700	82	0.7700	201		
0.1800	34	0.4800	84	0.7800	207		
0.1900	35	0.4900	87	0.7900	213		
0.2000	36	0.5000	89	0.8000	220		
0.2100	38	0.5100	92	0.8100	226		
0.2200	39	0.5200	95	0.8200	233		
0.2300	40	0.5300	98	0.8300	240		
0.2400	41	0.5400	101	0.8400	248		
0.2500	42	0.5500	104	0.8500	255		
0.2600	44	0.5600	107	0.8600	263		
0.2700	45	0.5700	110	0.8700	271		
0.2800	46	0.5800	114	0.8800	279		
0.2900	48	0.5900	117	0.8900	288		

**Appendix – Q Factor Table – 72 log scale Q values – [10.0, 0.3, 72]**

The data is presented as [float, node value] couples

0.0000	10	0.3521	2.9	0.7042	0.8
0.0141	9.5	0.3662	2.8	0.7183	0.8
0.0282	9.1	0.3803	2.6	0.7324	0.8
0.0423	8.6	0.3944	2.5	0.7465	0.7
0.0563	8.2	0.4085	2.4	0.7606	0.7
0.0704	7.8	0.4225	2.3	0.7746	0.7
0.0845	7.4	0.4366	2.2	0.7887	0.6
0.0986	7.1	0.4507	2.1	0.8028	0.6
0.1127	6.7	0.4648	2.0	0.8169	0.6
0.1268	6.4	0.4789	1.9	0.8310	0.5
0.1408	6.1	0.4930	1.8	0.8451	0.5
0.1549	5.8	0.5070	1.7	0.8592	0.5
0.1690	5.5	0.5211	1.6	0.8732	0.5
0.1831	5.3	0.5352	1.5	0.8873	0.4
0.1972	5.0	0.5493	1.5	0.9014	0.4
0.2113	4.8	0.5634	1.4	0.9155	0.4
0.2254	4.5	0.5775	1.3	0.9296	0.4
0.2394	4.3	0.5915	1.3	0.9437	0.4
0.2535	4.1	0.6056	1.2	0.9577	0.3
0.2676	3.9	0.6197	1.1	0.9718	0.3
0.2817	3.7	0.6338	1.1	0.9859	0.3
0.2958	3.5	0.6479	1.0	1.0000	0.3
0.3099	3.4	0.6620	1.0		
0.3239	3.2	0.6761	0.9		
0.3380	3.1	0.6901	0.9		



**Appendix – Hold Table – 101 log scale Hold values – [0.02, 2000.00, 101]**

The data is presented as [float, node value] couples

0.0000	0.02	0.3000	0.63	0.6000	20.0	0.9000	632
0.0100	0.02	0.3100	0.71	0.6100	22.4	0.9100	709
0.0200	0.03	0.3200	0.80	0.6200	25.1	0.9200	796
0.0300	0.03	0.3300	0.89	0.6300	28.2	0.9300	893
0.0400	0.03	0.3400	1.00	0.6400	31.7	0.9400	1002
0.0500	0.04	0.3500	1.12	0.6500	35.5	0.9500	1124
0.0600	0.04	0.3600	1.26	0.6600	39.9	0.9600	1261
0.0700	0.04	0.3700	1.42	0.6700	44.7	0.9700	1415
0.0800	0.05	0.3800	1.59	0.6800	50.2	0.9800	1588
0.0900	0.06	0.3900	1.78	0.6900	56.3	0.9900	1782
0.1000	0.06	0.4000	2.00	0.7000	63.2	1.0000	2000
0.1100	0.07	0.4100	2.24	0.7100	70.9		
0.1200	0.08	0.4200	2.52	0.7200	79.6		
0.1300	0.09	0.4300	2.83	0.7300	89.3		
0.1400	0.10	0.4400	3.17	0.7400	100		
0.1500	0.11	0.4500	3.56	0.7500	112		
0.1600	0.13	0.4600	3.99	0.7600	126		
0.1700	0.14	0.4700	4.48	0.7700	141		
0.1800	0.16	0.4800	5.02	0.7800	158		
0.1900	0.18	0.4900	5.64	0.7900	178		
0.2000	0.20	0.5000	6.32	0.8000	200		
0.2100	0.22	0.5100	7.10	0.8100	224		
0.2200	0.25	0.5200	7.96	0.8200	251		
0.2300	0.28	0.5300	8.93	0.8300	282		
0.2400	0.32	0.5400	10.0	0.8400	316		
0.2500	0.36	0.5500	11.2	0.8500	355		
0.2600	0.40	0.5600	12.6	0.8600	399		
0.2700	0.45	0.5700	14.1	0.8700	447		
0.2800	0.50	0.5800	15.8	0.8800	502		
0.2900	0.56	0.5900	17.8	0.8900	563		

**Appendix – Release Table – 101 log scale Release values – [5.00, 4000.00, 101]**

The data is presented as [float, node value] couples

0.0000	5	0.3000	37	0.6000	276	0.9000	2050
0.0100	5	0.3100	40	0.6100	295	0.9100	2192
0.0200	6	0.3200	42	0.6200	315	0.9200	2343
0.0300	6	0.3300	45	0.6300	337	0.9300	2505
0.0400	7	0.3400	49	0.6400	361	0.9400	2678
0.0500	7	0.3500	52	0.6500	385	0.9500	2864
0.0600	7	0.3600	55	0.6600	412	0.9600	3062
0.0700	8	0.3700	59	0.6700	441	0.9700	3273
0.0800	9	0.3800	63	0.6800	471	0.9800	3499
0.0900	9	0.3900	68	0.6900	504	0.9900	3741
0.1000	10	0.4000	72	0.7000	538	1.0000	4000
0.1100	10	0.4100	77	0.7100	576		
0.1200	11	0.4200	83	0.7200	615		
0.1300	12	0.4300	89	0.7300	658		
0.1400	13	0.4400	95	0.7400	703		
0.1500	14	0.4500	101	0.7500	752		
0.1600	15	0.4600	108	0.7600	804		
0.1700	16	0.4700	116	0.7700	860		
0.1800	17	0.4800	124	0.7800	919		
0.1900	18	0.4900	132	0.7900	983		
0.2000	19	0.5000	141	0.8000	1051		
0.2100	20	0.5100	151	0.8100	1123		
0.2200	22	0.5200	162	0.8200	1201		
0.2300	23	0.5300	173	0.8300	1284		
0.2400	25	0.5400	185	0.8400	1373		
0.2500	27	0.5500	198	0.8500	1468		
0.2600	28	0.5600	211	0.8600	1569		
0.2700	30	0.5700	226	0.8700	1677		
0.2800	32	0.5800	241	0.8800	1793		
0.2900	35	0.5900	258	0.8900	1917		

**Appendix – Level Table – 161 pseudo-log scale Level values – [-∞, +10, 161]**

The data is presented as [float, node value] couples

0.0000	-∞	0.2688	-28.5	0.5375	-8.5	0.8062	+2.3
0.0063	-87.0	0.2750	-28.0	0.5437	-8.3	0.8125	+2.5
0.0125	-84.0	0.2813	-27.5	0.5500	-8.0	0.8188	+2.8
0.0188	-81.0	0.2875	-27.0	0.5562	-7.8	0.8250	+3.0
0.0250	-78.0	0.2937	-26.5	0.5625	-7.5	0.8313	+3.3
0.0313	-75.0	0.3000	-26.0	0.5688	-7.3	0.8375	+3.5
0.0375	-72.0	0.3063	-25.5	0.5750	-7.0	0.8438	+3.8
0.0437	-69.0	0.3125	-25.0	0.5813	-6.8	0.8500	+4.0
0.0500	-66.0	0.3187	-24.5	0.5875	-6.5	0.8562	+4.3
0.0562	-63.0	0.3250	-24.0	0.5938	-6.3	0.8625	+4.5
0.0625	-60.0	0.3313	-23.5	0.6000	-6.0	0.8687	+4.8
0.0688	-59.0	0.3375	-23.0	0.6062	-5.8	0.8750	+5.0
0.0750	-58.0	0.3438	-22.5	0.6125	-5.5	0.8813	+5.3
0.0812	-57.0	0.3500	-22.0	0.6187	-5.3	0.8875	+5.5
0.0875	-56.0	0.3562	-21.5	0.6250	-5.0	0.8938	+5.8
0.0938	-55.0	0.3625	-21.0	0.6313	-4.8	0.9000	+6.0
0.1000	-54.0	0.3688	-20.5	0.6375	-4.5	0.9063	+6.3
0.1063	-53.0	0.3750	-20.0	0.6438	-4.3	0.9125	+6.5
0.1125	-52.0	0.3812	-19.5	0.6500	-4.0	0.9187	+6.8
0.1187	-51.0	0.3875	-19.0	0.6563	-3.8	0.9250	+7.0
0.1250	-50.0	0.3938	-18.5	0.6625	-3.5	0.9312	+7.3
0.1312	-49.0	0.4000	-18.0	0.6687	-3.3	0.9375	+7.5
0.1375	-48.0	0.4063	-17.5	0.6750	-3.0	0.9438	+7.8
0.1437	-47.0	0.4125	-17.0	0.6812	-2.8	0.9500	+8.0
0.1500	-46.0	0.4187	-16.5	0.6875	-2.5	0.9563	+8.3
0.1563	-45.0	0.4250	-16.0	0.6938	-2.3	0.9625	+8.5
0.1625	-44.0	0.4313	-15.5	0.7000	-2.0	0.9688	+8.8
0.1688	-43.0	0.4375	-15.0	0.7063	-1.8	0.9750	+9.0
0.1750	-42.0	0.4437	-14.5	0.7125	-1.5	0.9812	+9.3
0.1813	-41.0	0.4500	-14.0	0.7188	-1.3	0.9875	+9.5
0.1875	-40.0	0.4563	-13.5	0.7250	-1.0	0.9937	+9.8
0.1937	-39.0	0.4625	-13.0	0.7312	-0.8	1.0000	+10.0
0.2000	-38.0	0.4688	-12.5	0.7375	-0.5		
0.2062	-37.0	0.4750	-12.0	0.7437	-0.3		
0.2125	-36.0	0.4812	-11.5	0.7500	+0.0		
0.2188	-35.0	0.4875	-11.0	0.7563	+0.3		
0.2250	-34.0	0.4938	-10.5	0.7625	+0.5		
0.2313	-33.0	0.5000	-10.0	0.7688	+0.8		
0.2375	-32.0	0.5063	-9.8	0.7750	+1.0		
0.2438	-31.0	0.5125	-9.5	0.7813	+1.3		
0.2500	-30.0	0.5188	-9.3	0.7875	+1.5		
0.2562	-29.5	0.5250	-9.0	0.7937	+1.8		
0.2625	-29.0	0.5313	-8.8	0.8000	+2.0		

**Appendix – RTA Decay Table – 19 log scale Decay values – [0.25, 16, 19]**

The data is presented as [float, node value] couples

0.0000 0.25	0.5556 2.52			
0.0556 0.31	0.6111 3.17			
0.1111 0.40	0.6667 4.00			
0.1667 0.50	0.7222 5.04			
0.2222 0.63	0.7778 6.35			
0.2778 0.79	0.8333 8.00			
0.3333 1.00	0.8889 10.08			
0.3889 1.26	0.9444 12.70			
0.4444 1.59	1.0000 16.00			
0.5000 2.00				

## Appendix – Effects enums, names and preset names table

The data is presented as [enum value, enum name, preset flags, preset name] quadruplets

### FX1...FX4:

0	"HALL"	%000000	Hall Reverb	30	"TEQ"	%011001	Strereo TrueEQ
1	"AMBI"	%000101	Ambiance	31	"DES2"	%101011	Dual DeEsser
2	"RPLT"	%000011	Rich Plate Reverb	32	"DES"	%101010	Stereo DeEsser
3	"ROOM"	%000010	Room Reverb	33	"P1A"	%101100	Stereo Xtec EQ1
4	"CHAM"	%000001	Chamber Reverb	34	"P1A2"	%101101	Dual Xtec EQ1
5	"PLAT"	%000100	Plate Reverb	35	"PQ5"	%101110	Stereo Xtec EQ5
6	"VREV"	%001001	Vintage Reverb	36	"PQ5S"	%101111	Dual Xtec EQ5
7	"VRM"	%001000	Vintage room	37	"WAVD"	%011101	Wave Designer
8	"GATE"	%000110	Gated Reverb	38	"LIM"	%011110	Precision Limiter
9	"RVRS"	%000111	Reverse Reverb	39	"CMB"	%111011	Combinator
10	"DLY"	%010100	Stereo Delay	40	"CMB2"	%111100	Dual Combinator
11	"3TAP"	%010101	3-Tap Delay	41	"FAC"	%110000	Fair Comp
12	"4TAP"	%010110	Rhythm Delay	42	"FAC1M"	%110001	M/S Fair Comp
13	"CRS"	%001010	Stereo Chorus	43	"FAC2"	%110010	Dual Fair Comp
14	"FLNG"	%001011	Stereo Flanger	44	"LEC"	%110011	Leisure Comp
15	"PHAS"	%011011	Stereo Phaser	45	"LEC2"	%110100	Dual Leisure Comp
16	"DIMC"	%111010	Dimension-C	46	"ULC"	%110101	Ultimo Comp
17	"FILT"	%101001	Mood Filter	47	"ULC2"	%110110	Dual Ultimo Comp
18	"ROTA"	%011100	Rotary Speaker	48	"ENH2"	%100000	Dual Enhancer
19	"PAN"	%101000	Tremolo/Panner	49	"ENH"	%011111	Stereo Enhancer
20	"SUB"	%111001	Suboctaver	50	"EXC2"	%100010	Dual Exciter
21	"D/RV"	%010000	Delay+Chamber	51	"EXC"	%100001	Stereo Exciter
22	"CR/R"	%001110	Chorus+Chamber	52	"IMG"	%100111	Stereo Imager
23	"FL/R"	%001111	Flanger+Chamber	53	"EDI"	%111000	Edison EX1
24	"D/CR"	%010001	Delay+Chorus	54	"SON"	%110111	Sound Maxer
25	"D/FL"	%010010	Delay+Flanger	55	"AMP2"	%100100	Dual Guitar Amp
26	"MODD"	%010011	Modulation Delay	56	"AMP"	%100011	Stereo Guitar Amp
27	"GEQ2"	%011000	Dual Graphic EQ	57	"DRV2"	%100110	Dual Tube Stage
28	"GEQ"	%010111	Stereo Graphic EQ	58	"DRV"	%100101	Stereo Tube Stage
29	"TEQ2"	%011010	Dual TrueEQ	59	"PIT2"	%001101	Dual Pitch Shifter
				60	"PIT"	%001100	Stereo Pitch

### FX5...FX8:

0	"GEQ2"	%011000	Dual Graphic EQ	16	"ULC"	%110101	Ultimo Comp
1	"GEQ"	%010111	Stereo Graphic EQ	17	"ULC2"	%110110	Dual Ultimo Comp
2	"TEQ2"	%011010	Dual TrueEQ	18	"ENH2"	%100000	Dual Enhancer
3	"TEQ"	%011001	Strereo TrueEQ	19	"ENH"	%011111	Stereo Enhancer
4	"DES2"	%101011	Dual DeEsser	20	"EXC2"	%100010	Dual Exciter
5	"DES"	%101010	Stereo DeEsser	21	"EXC"	%100001	Stereo Exciter
6	"P1A"	%101100	Stereo Xtec EQ1	22	"IMG"	%100111	Stereo Imager
7	"P1A2"	%101101	Dual Xtec EQ1	23	"EDI"	%111000	Edison EX1
8	"PQ5"	%101110	Stereo Xtec EQ5	24	"SON"	%110111	Sound Maxer
9	"PQ5S"	%101111	Dual Xtec EQ5	25	"AMP2"	%100100	Dual Guitar Amp
10	"WAVD"	%011101	Wave Designer	26	"AMP"	%100011	Stereo Guitar Amp
11	"LIM"	%011110	Precision Limiter	27	"DRV2"	%100110	Dual Tube Stage
12	"FAC"	%110000	Fair Comp	28	"DRV"	%100101	Stereo Tube Stage
13	"FAC1M"	%110001	M/S Fair Comp	29	"PIT2"	%001101	Dual Pitch Shifter
14	"LEC"	%110011	Leisure Comp	30	"PIT"	%001100	Stereo Pitch
15	"LEC2"	%110100	Dual Leisure Comp				

## Appendix – Programming Examples

Would you take on starting programming for the X32/M32 series, below is a small “Hello World” C program to open a communication stream with X32/M32, send a simple command, and receive an answer back from X32/M32.

### HelloX32 (Unix)

```
//
// HelloX32.c
//
// Simple example of communication setup with an X32/M32
// Uses UDP protocol
// X32/M32 should be set at IP = 192.168.0.64
// Communication takes place on port 10023
//
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>

int
main(int argc, char **argv)
{
char          Xip_str[20], Xport_str[8]; // X32/M32 IP and Port
struct sockaddr_in Xip;
struct sockaddr* Xip_addr = (struct sockaddr *)&Xip;
socklen_t      Xip_len = sizeof(Xip); // length of addresses
int            Xfd; // X32/M32 socket
int            rec_len, i;
char          rec_buf[256]; // receive buffer
char          info_buf[8] = "/info"; // zeroes are automatically added
//
// Initialize communication with X32/M32 server at IP ip and PORT port
// Set default values to match your X32/M32 desk
    strcpy (Xip_str, "192.168.0.64");
    strcpy (Xport_str, "10023");
// Load the X32/M32 address we connect to; we're a client to X32/M32, keep it simple.
// Create UDP socket
    if ((Xfd = socket (PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0)
        exit (EXIT_FAILURE);
// Construct server sockaddr_in structure
    memset (&Xip, 0, sizeof(Xip)); // Clear struct
    Xip.sin_family = AF_INET; // Internet/IP
    Xip.sin_addr.s_addr = inet_addr(Xip_str); // IP address
    Xip.sin_port = htons(atoi(Xport_str)); // server port
// All done. Let's establish connection with X32/M32 server
    printf(" HelloX32 - v0.9 - 2014 Patrick-Gilles Maillot\n\n");
    printf("Connecting to Console\n");
    if (sendto (Xfd, info_buf, sizeof(info_buf), 0, Xip_addr, Xip_len) < 0)
        exit (EXIT_FAILURE);
// Receive answer back from X32/M32
    if ((rec_len = recvfrom (Xfd, rec_buf, sizeof(rec_buf), 0, 0, 0)) <= 0)
        exit (EXIT_FAILURE);
    printf("Buffer data from Console: %d bytes,\n", rec_len);
    i = 0;
    while(rec_len--) {
        if (rec_buf[i] == 0) rec_buf[i] = '~'; // handle non-printable chars
        putchar(rec_buf[i++]);
    }
    putchar('\n');
// All done!
    exit(0);
}
```

### X32 Connect, Send and Receive (Unix)

A set of 3 programs to connect to X32, send data to X32 and receive data from X32, with a timeout. This set of programs was initially created to help provide a set of UDP functions for handling communication with X32 from a Lazarus (pascal) environment as can be ran on a Raspberry Pi or even maybe on a Mac.

Lazarus indeed interfaces easily with C functions which can be compiled separately, and used at link time.

```
/*
 * X32UDP.c
 *   (POSIX compliant version, Linux...)
 *   Created on: June 2, 2015
 *   Author: Patrick-Gilles Maillot
 *
 *   Copyright 2015, Patrick-Gilles Maillot
 *   This software is distributed under the GNU GENERAL PUBLIC LICENSE.
 *
 *   This software allows connecting to a remote X32 or XAIR system using
 *   UDP protocol; It provides a set of connect, send and receive functions.
 *   The receive mode is non-blocking, i.e. a timeout enables returning from
 *   the call even if no response is obtained by the server.
 *
 *   Send and Receive buffers are provided by the caller. No provision is
 *   made in this package to keep or buffer data for deferred action or
 *   transfers.
 */

#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <poll.h>
//
#define BSIZE 512          // MAX receive buffer size
//
struct sockaddr_in  Xip;
struct sockaddr*    Xip_addr = (struct sockaddr *)&Xip;
socklen_t          Xip_len = sizeof(Xip);          // length of addresses
int                Xfd;                          // X32 socket
struct pollfd      ufds;
//
int                r_len, p_status;              // length and status for receiving
//
//
int X32Connect(char *Xip_str, char *Xport_str) {
//
// Initialize communication with X32 server at IP ip and PORT port//
// Load the X32 address we connect to; we're a client to X32, keep it simple.
//
// Input: String - Pointer to IP in the form "123.123.123.123"
// Input: String - Pointer to destination port in the form "12345"
//
// Returns int:
// -3 : Error on Sending data
// -2 : Socket creation error
// -1 : Error on polling for data
// 0 : No error, no connection (timeout)
// 1 : Connected (connection validated with X32)
//
char  r_buf[128];          // receive buffer for /info command test
char  Info[8] = "/info";  // testing connection with /info request (X32, M32)
//char Info[8] = "/xinfo"; // testing connection with /xinfo request (XR series)
//
// Create UDP socket
    if ((Xfd = socket (PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0) {
```

```

        return -2; // An error occurred on socket creation
    }
// Server sockaddr_in structure
    memset (&Xip, 0, sizeof(Xip));           // Clear structure
    Xip.sin_family = AF_INET;                // Internet/IP
    Xip.sin_addr.s_addr = inet_addr(Xip_str); // IP address
    Xip.sin_port = htons(atoi(Xport_str));  // server port
//
// Prepare for poll() on receiving data
    ufds.fd = Xfd;
    ufds.events = POLLIN; //Check for normal data
//
// Validate connection by sending a /info command
    if (sendto (Xfd, Info, 8, 0, Xip_addr, Xip_len) < 0) {
        return (-3);
    }
    if ((p_status = poll (&ufds, 1, 100)) > 0) { // X32 sent something?
        r_len = recvfrom(Xfd, r_buf, 128, 0, 0, 0); // Get answer and
        if ((strncmp(r_buf, Info, 5)) == 0) {      // test data (5 bytes)
            return 1; // Connected
        }
    } else if (p_status < 0) {
        return -1; // Error on polling (not connected)
    }
// Not connected on timeout
    return 0;
}

int X32Send(char *buffer, int length) {
//
// Sends data to X32 server at IP and PORT previously set
// with X32Connect()
//
// Input: String - Pointer to data buffer to send
// Input: integer - Length of data in bytes
//
// Returns int:
// -1 : Error on Sending data
// >= 0 : Actual length of data sent, no error
//
// Just sending data
    return (sendto (Xfd, buffer, length, 0, Xip_addr, Xip_len));
}

int X32Recv(char *buffer, int timeout) {
//
// Receives data from X32 server
//
// Input: String - Pointer to buffer to save data to,
//           should be capable of receiving 512 bytes
// Input: integer - Timeout (in ms) for receiving data:
//           - a timeout value of 0 will always return no data
//           - a negative value for timeout means "infinite time"
//           - depending on systems capabilities, positive and non-zero
//             values may also result in no data (start with a value of 10ms)
//
// Returns int:
// -1 : Error on polling for data
// 0 : No error, timeout reached or no data
// >0 : data length in the buffer
//
    if ((p_status = poll (&ufds, 1, timeout)) > 0) { // Data in?
        return recvfrom(Xfd, buffer, BSIZE, 0, 0, 0); // return length
    } else if (p_status < 0) {
        return -1; //An error occurred on polling
    }
}

```



```

    }
    return 0; // No error, timeout
}

////
//// Test purpose only - comment when linking the package to an application
////
#include <stdio.h>
int main() {
//
//     char   r_buf[512];
//     char   s_buf[] = {"/status\0"};
//     int     r_len = 0;
//     int     s_len = 0;
//     int     status;
//
//     status = X32Connect("192.168.1.67", "10023");
//     printf ("Connection status: %d\n", status);
//
//     if (status) {
//         s_len = X32Send(s_buf, 8);
//         printf ("Send status: %d\n", s_len);
//         if (s_len) {
//             r_len = X32Recv(r_buf, 10);
//             printf ("Recv status: %d\n", r_len);
//             s_len = 0;
//             while (r_len-- > 0) {
//                 if (r_buf[s_len] < ' ') putchar('~');
//                 else putchar(r_buf[s_len]);
//                 s_len++;
//             }
//         }
//     }
//     return 0;
//}

```

**Compile and link with:**

```
gcc -c -O3 -Wall -fmessage-length=0 X32UDP.c
```

## X32Saver (Unix)

A slightly more complex example, where the program registers to receive updates from the Server. The program sets a screen saver mode, lowering the power of main LCD and LEDs of the X32/M32 after a given delay without activity.

Two versions are proposed below, the first one is aimed at POSIX compliant systems; the second version is for Windows environments.

```
/*
 * X32Ssaver.c
 *   (POSIX compliant version, Linux...)
 *   Created on: May 7, 2015
 *   Author: Patrick-Gilles Maillot
 *
 *   Copyright 2015, Patrick-Gilles Maillot
 *   This software is distributed under the GNU GENERAL PUBLIC LICENSE.
 *   Changelog:
 *   Use of select() rather than poll() for unblocking IO
 */

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
//
#include <sys/socket.h>
#include <sys/select.h>
#include <arpa/inet.h>
//
#define BSIZE                512           // receive buffer size
#define XREMOTE_TIMEOUT      9           // time-out set to 9 seconds
#define TRUE                 1
#define FALSE                0
//
// Macros:
//
#define RPOLL                 \
do {                          \
    FD_ZERO (&ufds);         \
    FD_SET (Xfd, &ufds);     \
    p_status = select(Xfd+1, &ufds, NULL, NULL, &timeout); \
} while (0);
//
#define RECV                 \
do {                          \
    r_len = recvfrom(Xfd, r_buf, BSIZE, 0, 0, 0); \
} while (0);
//
#define SEND(a,l)           \
do {                        \
    if (sendto (Xfd, a, l, 0, Xip_addr, Xip_len) < 0) { \
        perror ("Error while sending data"); \
        exit (1); \
    } \
} while(0);
//
int
main(int argc, char **argv)
{
    struct sockaddr_in    Xip;
    struct sockaddr*      Xip_addr = (struct sockaddr *)&Xip;
    int                   Xfd;           // X32 socket
    char                  Xip_str[20], Xport_str[8]; // X32 IP and port
    socklen_t             Xip_len = sizeof(Xip); // length of addresses
```

```

fd_set          ufds;
struct timeval  timeout;
//
int             r_len, p_status;          // length and status for receiving
char           r_buf[BFSIZE];           // receive buffer
//
char           xremote[12] = "/xremote"; // automatic trailing zeroes
int           X32ssdelay = 5;           // Default Ssaver time-out value
int           keep_on = 1;              // Loop flag
int           ssave_on = 0;             // Screen Saver ON state
time_t        X32Rmte_bfr, X32Rmte_now; // For /xremote timer
time_t        X32Ssav_bfr, X32Ssav_now; // For Screen Saver timer
char          LcdOldBright[24];         // value of X32 screen brightness
char          LedOldBright[28];         // value of X32 LED brightness
//
char          LcdLowBright[] = "/-prefs/bright\0\0,f\0\0\0\0\0\0";
char          LedLowBright[] = "/-prefs/ledbright\0\0\0,f\0\0\0\0\0\0";
//
//
// Initialize communication with X32 server at IP ip and PORT port
// Set default values to match your X32 desk
strcpy (Xip_str, "192.168.0.64");
strcpy (Xport_str, "10023");
//
// Manage arguments
while ((*r_buf = getopt(argc, argv, "i:d:h")) != (char)-1) {
    switch (*r_buf) {
        case 'i':
            strcpy(Xip_str, optarg );
            break;
        case 'd':
            sscanf(optarg, "%d", &X32ssdelay);
            break;
        default:
        case 'h':
            printf("usage: X32Ssaver [-i X32 console ipv4 address [192.168.0.64]\n");
            printf("          [-d delay(s) [5], delay before entering Screen
Saver]\n");
            return(0);
            break;
    }
}
//
// Load the X32 address we connect to; we're a client to X32, keep it simple.
// Create UDP socket
if ((Xfd = socket (PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0) {
    perror ("failed to create X32 socket");
    exit (1);
}
// Server sockaddr_in structure
memset (&Xip, 0, sizeof(Xip));          // Clear structure
Xip.sin_family = AF_INET;               // Internet/IP
Xip.sin_addr.s_addr = inet_addr(Xip_str); // IP address
Xip.sin_port = htons(atoi(Xport_str)); // server port/
//
// Prepare for select() call on receiving data
timeout.tv_sec = 0;
timeout.tv_usec = 100000; //Set timeout for non blocking recvfrom(): 100ms
FD_ZERO(&ufds);
FD_SET(Xfd, &ufds);
//
// Establish connection with X32 server
printf(" X32Ssaver - v0.10 - (c)2015 Patrick-Gilles Maillot\n\n");
//
keep_on = 1;                            // Run forever, or until an error occurs
X32Rmte_bfr = 0;

```

```

X32Ssav_bfr = time(NULL);
while (keep_on) {
    X32Rmte_now = time(NULL);    // register for X32 data echo
    if (X32Rmte_now > X32Rmte_bfr + XREMOTE_TIMEOUT) {
        if (sendto (Xfd, xremote, 12, 0, Xip_addr, Xip_len) < 0)
            exit (1);
        X32Rmte_bfr = X32Rmte_now;
    }
    RPOLL                // X32 sent something?
    if ((p_status = FD_ISSET(Xfd, &ufds)) > 0) {
        RECV              // Exit screen saver if needed
        if (r_len && ssave_on) { // Restore main LCD and LED brightness
            SEND(LcdOldBright, 24)
            SEND(LedOldBright, 28)
            ssave_on = FALSE;    // S-saver mode is OFF
        }
        X32Ssav_bfr = time(NULL);    // remember time
        //
    } else if (p_status == 0) {      // no data back from X32, enter screen saver?
        X32Ssav_now = time(NULL);
        if (X32Ssav_now > X32Ssav_bfr + X32ssdelay) {
            if (!ssave_on) {        // No need to enter saver mode if already ON
                SEND(LcdLowBright, 16)
                RPOLL
                if ((p_status = FD_ISSET(Xfd, &ufds)) > 0) {
                    RECV            // expected: /-prefs/bright...[float]
                    memcpy(LcdOldBright, r_buf, 24);
                } // main screen brightness saved, ignore errors (p_status < 0)
                SEND(LedLowBright, 20)
                RPOLL
                if ((p_status = FD_ISSET(Xfd, &ufds)) > 0) {
                    RECV            // expected: /-prefs/ledbright...[float]
                    memcpy(LedOldBright, r_buf, 28);
                } // Leds and Scribbles brightness saved, ignore errors (p_status < 0)
                // Set LCD screen and LEDs to their lowest values
                SEND(LcdLowBright, 24)
                SEND(LedLowBright, 28)
                ssave_on = TRUE;    // S-saver is ON
            }
        }
    }
    } else keep_on = 0; // Exit on error (p_status < 0)
}
close(Xfd);
return 0;
}

```

#### Compile and link with:

```
gcc -O3 -Wall -fmessage-length=0 -o X32Ssaver.exe X32Ssaver.c
```

## X32Saver (Windows)

Below is the Windows-only version of the same program. There are small differences in the way UDP commands are handled. It is important to note that Windows does not offer as a precise timing control as Linux does. This may have an impact on some programs relying on precise timings.

```
/*
 * X32Ssaver.c
 *   (Windows environment version)
 *   Created on: May 7, 2015
 *   Author: Patrick-Gilles Maillot
 *
 *   Copyright 2015, Patrick-Gilles Maillot
 *   This software is distributed under the GNU GENERAL PUBLIC LICENSE.
 *   Changelog:
 *   Use of select() rather than poll() for unblocking IO
 */

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
//
#include <winsock2.h>
//
#define BSIZE          512          // receive buffer size
#define XREMOTE_TIMEOUT 9          // time-out set to 9 seconds
#define TRUE          1
#define FALSE         0
//
// Macros:
//
#define R POLL                    \
do {                               \
    FD_ZERO (&ufds);              \
    FD_SET (Xfd, &ufds);          \
    p_status = select(Xfd+1, &ufds, NULL, NULL, &timeout); \
} while (0);
//
#define RECV                    \
do {                               \
    r_len = recvfrom(Xfd, r_buf, BSIZE, 0, 0, 0); \
} while (0);
//
#define SEND(a,l)                \
do {                               \
    if (sendto (Xfd, a, l, 0, Xip_addr, Xip_len) < 0) { \
        perror ("Error while sending data"); \
        exit (1); \
    } \
} while(0);
//
int
main(int argc, char **argv)
{
    struct sockaddr_in Xip;
    struct sockaddr* Xip_addr = (struct sockaddr *)&Xip;
    int Xfd; // X32 socket
    char Xip_str[20], Xport_str[8]; // X32 IP and port
    WSADATA wsa;
    int Xip_len = sizeof(Xip); // length of addresses
    struct timeval timeout;
    fd_set ufds;
    //
```

```

int          r_len, p_status;          // length and status for receiving
char        r_buf[BFSIZE];           // receive buffer
//
char        xremote[12] = "/xremote"; // automatic trailing zeroes
int         X32ssdelay = 5;          // Default Ssaver time-out value
int         keep_on = 1;             // Loop flag
int         ssave_on = 0;            // Screen Saver ON state
time_t      X32Rmte_bfr, X32Rmte_now; // For /xremote timer
time_t      X32Ssav_bfr, X32Ssav_now; // For Screen Saver timer
char        LcdOldBright[24];        // value of X32 screen brightness
char        LedOldBright[28];        // value of X32 LED brightness
//
char        LcdLowBright[] = "/-prefs/bright\0\0,f\0\0\0\0\0\0";
char        LedLowBright[] = "/-prefs/ledbright\0\0\0,f\0\0\0\0\0\0";
//
//
// Initialize communication with X32 server at IP ip and PORT port
// Set default values to match your X32 desk
strcpy (Xip_str, "192.168.1.13");
strcpy (Xport_str, "10023");
//
// Manage arguments
while ((*r_buf = getopt(argc, argv, "i:d:h")) != (char)-1) {
    switch (*r_buf) {
        case 'i':
            strcpy(Xip_str, optarg );
            break;
        case 'd':
            sscanf(optarg, "%d", &X32ssdelay);
            break;
        default:
        case 'h':
            printf("usage: X32Ssaver [-i X32 console ipv4 address [192.168.0.64]\n");
            printf("          [-d delay(s) [5], delay before entering Screen
Saver]\n");
            return(0);
            break;
    }
}
//Initialize winsock
if (WSAStartup (MAKEWORD( 2, 2), &wsa) != 0) {
    printf ("Failed. Error Code : %d", WSAGetLastError());
    exit (EXIT_FAILURE);
}
//
// Load the X32 address we connect to; we're a client to X32, keep it simple.
// Create UDP socket
if ((Xfd = socket (PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0) {
    perror ("failed to create X32 socket");
    exit (1);
}
// Server sockaddr_in structure
memset (&Xip, 0, sizeof(Xip));          // Clear structure
Xip.sin_family = AF_INET;               // Internet/IP
Xip.sin_addr.s_addr = inet_addr(Xip_str); // IP address
Xip.sin_port = htons(atoi(Xport_str)); // server port/
//
// Prepare for select() call on receiving data
timeout.tv_sec = 0;
timeout.tv_usec = 100000; //Set timeout for non blocking recvfrom(): 100ms
ufds.fd_array[0] = Xfd;
ufds.fd_count = 1;
//
// Establish connection with X32 server
printf(" X32Ssaver - v0.10 - (c)2015 Patrick-Gilles Maillot\n\n");
//

```

```

keep_on = 1; // Run forever, or until an error occurs
X32Rmte_bfr = 0;
X32Ssav_bfr = time(NULL);
while (keep_on) {
    X32Rmte_now = time(NULL); // register for X32 data echo
    if (X32Rmte_now > X32Rmte_bfr + XREMOTE_TIMEOUT) {
        if (sendto (Xfd, xremote, 12, 0, Xip_addr, Xip_len) < 0)
            exit (1);
        X32Rmte_bfr = X32Rmte_now;
    }
    RPOLL // X32 sent something?
    if ((p_status = FD_ISSET(Xfd, &ufds)) > 0) {
        RECV // Exit screen saver if needed
        if (r_len && ssave_on) { // Restore main LCD and LED brightness
            SEND(LcdOldBright, 24)
            SEND(LedOldBright, 28)
            ssave_on = FALSE; // S-saver mode is OFF
        }
        X32Ssav_bfr = time(NULL); // remember time
    }
    //
    } else if (p_status == 0) { // no data back from X32, enter screen saver?
        X32Ssav_now = time(NULL);
        if (X32Ssav_now > X32Ssav_bfr + X32ssdelay) {
            if (!ssave_on) { // No need to enter saver mode if already ON
                SEND(LcdLowBright, 16)
                RPOLL
                if ((p_status = FD_ISSET(Xfd, &ufds)) > 0) {
                    RECV // expected: /-prefs/bright...[float]
                    memcpy(LcdOldBright, r_buf, 24);
                } // main screen brightness saved, ignore errors (p_status < 0)
                SEND(LedLowBright, 20)
                RPOLL
                if ((p_status = FD_ISSET(Xfd, &ufds)) > 0) {
                    RECV // expected: /-prefs/ledbright...[float]
                    memcpy(LedOldBright, r_buf, 28);
                } // Leds and Scribbles brightness saved, ignore errors (p_status < 0)
                // Set LCD screen and LEDs to their lowest values
                SEND(LcdLowBright, 24)
                SEND(LedLowBright, 28)
                ssave_on = TRUE; // S-saver is ON
            }
        }
    } else keep_on = 0; // Exit on error (p_status < 0)
}
close(Xfd);
return 0;
}

```

#### Compile and link with:

```

gcc -O3 -Wall -c -fmessage-length=0 -o X32Ssaver.o X32Ssaver.c
gcc -o X32Ssaver.exe X32Ssaver.o -lws2_32

```

## X32 data echo in Go

Below is a program written in Go (see <https://golang.org/>) running on Mac, Unix, Windows, etc.; Go is an open source programming language. This small example establishes a UDP connection with X32 and echoing data changed by X32 while maintaining a /xremote request active every 9 seconds. In an infinite loop, UDP reads are non-blocking with a timeout of 100µs. (Use Go v 1.5 at a minimum).

```
package main

import (
    "fmt"
    "net"
    "time"
)

// Simple print function to echo X32 data
func prints(n int, p []byte) {
    fmt.Printf("%3d bytes - ", n)
    for i := 0; i < n; i++ {
        fmt.Printf("%c", p[i])    // Go replaces \0 chars with spaces
    }
    fmt.Printf("\n")
}

// Main
func main() {
    var n int
    var timeold = time.Now().Add(-10000000000)    // sets time to 10s before start
    var timenow = timeold
    var timeout = time.Duration(100000)    // ReadFrom() timeout set to 100 microseconds
    p := make([]byte, 1024)    // 1024 bytes buffer
    //
    // Change IP and port as needed below
    sAddr, err := net.ResolveUDPAddr("udp", "192.168.1.64:10023")
    // Validate UDP connection
    conn, err := net.DialUDP("udp", nil, sAddr)
    if err != nil {
        fmt.Printf("Connexion Error %v", err)
        return
    }
    defer conn.Close()
    // Dialog with X32
    _, err = conn.Write([]byte("/info"))    // /info request
    n, _, err = conn.ReadFromUDP(p)    // Read info data ignoring UDP address
    if err == nil {
        prints(n, p)
    } else {
        fmt.Printf("Error %v\n", err)
    }
    _, err = conn.Write([]byte("/status"))    // /status request
    n, _, err = conn.ReadFromUDP(p)    // Read X32 status
    if err == nil {
        prints(n, p)
    } else {
        fmt.Printf("Error %v\n", err)
    }
}
for { // Loop forever echoing X32 changes
    timenow = time.Now()
    if (timenow.Sub(timeold)).Seconds() > 9 {    // every 9 seconds...
        conn.Write([]byte("/xremote"))    // Maintain X32 notifications
        timeold = timenow    // update time (use time.Now())
        // if need more accuracy
    }
    conn.SetReadDeadline(timenow.Add(timeout))    // time.Now() could replace
```



```
                                // timenow
n, _, _ = conn.ReadFromUDP(p)    // Non-blocking reads; Ignore errors (timeouts)
if n > 0 {                       // check number of bytes
    prints(n, p)
}
}
```

**Compile with:**

```
$GOROOT\bin\go.exe install -v -gcflags "-N -l"
```

## Appendix – Miscellaneous

### Floating point data representation

An example of fader set command to change fader for channel 1 with a value of 0.9

```
/ch/01/mix/fader~~~~f~~?ffh  
2f63682f30312f6d69782f6661646572000000002c6600003f666668
```

Some floating point values converted to big endian data as sent by/to the X32/M32 console

0.0	00000000
0.1	3dcccccd
0.2	3e4ccccd
0.3	3e99999a
0.4	3ecccccd
0.5	3f000000
0.6	3f19999a
0.7	3f333334
0.8	3f4cccce
0.9	3f666668
1.0	3f800000

### Notes Features or Messages to be validated or needing further testing

```
/batchsubscribe/mfm_a /mix/on 0 63 8  
/formatsubscribe/mfm_c /dca*/on 1 8 8  
/subscribe/-stat/solosw/07 3  
/batchsubscribe/x_meters_0 /meters/0 0 69 1
```

```
/formatsubscribe~~~~ssssssssssiii~fsub_cross_MBlk~~~~/config/buslink/1-  
2~/config/buslink/3-4~/config/buslink/5-6~/config/buslink/7-  
8~/config/buslink/9-10~~~~/config/buslink/11-12~~~/config/buslink/13-  
14~~~/config/buslink/15-16~~~/config/linkcfg/fdrmute~[ 0][ 0][ 10]
```

The line above comes from X32-Q. It is the first command sent following a `/info` command to establish connection with the X32 server.

The first string of the command seems to be the name assigned to the batch format request the other strings being the elements the request is for. The command will run for about 10s (as a `/xremote` client update command) and will have to be renewed in order for the requesting client to continue receiving information from X32/M32. The last int represents the update frequency factor within the 10 seconds.

The result of `batchsubscribe` for `/config/name` contains a blob with exactly 32 bytes:

Code:

```
00000000h: 2F 62 2F 63 6F 6E 66 69 67 6E 61 6D 65 00 00 00 ; /b/configname...  
00000010h: 2C 62 00 00 00 00 00 24 24 00 00 00 4B 69 63 6B ; b.....$$...Kick  
00000020h: 00 4D 00 E3 7D 9C 40 C5 6D C7 E9 0E 84 65 E4 2E ; .M.ã}œ@ÅmÇé.„eä.  
00000030h: BD A0 80 BF 7D F7 D1 58 F9 D7 51 1A ; ½ €;}÷ÑXù×Q.
```

The OSC command `/renew` is used to keep existing subscriptions alive

The OSC command `/unsubscribe` is used to stop all subscriptions when a client application decides to exit/quit.

From X32/M32 while loading scene over another:

```
/-undo/time~s~17:11:30~
```

... Scene loads a bunch of individual settings (`/auxin /config/routing /ch...`)

```
/-ha/[00...31]index~~i~-[-1]
```

X32-edit on a PC saving a snippet to the X32 at position 001 the following are sent by X32/M32 to other clients:

```
/-show/showfile/snippet/001/eventtyp~~~~i~~<int>  
/-show/showfile/snippet/001/channels~~~~i~~<int>  
/-show/showfile/snippet/001/auxbuses~~~~i~~<int>  
/-show/showfile/snippet/001/maingrps~~~~i~~<int>  
/-show/showfile/snippet/001/name~~~~s~~<name>  
/-show/showfile/snippet/001/hasdata~i~~[ 1]
```

An update with X32-edit on a PC affecting eventtyp and auxbuses will translate into the following sent by X32/M32 to other clients (if registererd with `/xremote`):

```
/-show/showfile/snippet/001/eventtyp~~~~i~~<int>  
/-show/showfile/snippet/001/auxbuses~~~~i~~<int>
```

X32-edit on a PC deleting the snippet at position 001 within X32-edit will translate into the following sent by X32/M32 to other clients (if registererd with `/xremote`):

```
/-show/showfile/snippet/001/name~~~~s~~~~~ (nil string)  
/-show/showfile/snippet/001/eventtyp~~~~i~~[ 0]  
/-show/showfile/snippet/001/channels~~~~i~~[ 0]  
/-show/showfile/snippet/001/auxbuses~~~~i~~[ 0]  
/-show/showfile/snippet/001/maingrps~~~~i~~[ 0]  
/-show/showfile/snippet/001/hasdata~i~~[ 0]
```

X32-edit on a PC loading snippet at position 002 within X32-edit will translate into the following sent by X32/M32 to other clients(if registererd with `/xremote`):

```
/-action/newscreen~~i~~[ 0]
```

...But no data is actually loaded to the X32-edit program. X32-edit screen is not updated. X32 and X32-edit screen are different!

When using X32-edit on a PC selecting RTA will generate the following data sent to X32/M32:

```
/batchsubscribe~ssiii~~meters/15~~~/meters/15~~[ 0][ 0][ 1]  
/renew~~s~~meters/15~~
```

Need to find information on the following commands (not all may exist!):

```
/-prefs/iQ/01-16  
/hidden/prefs  
/hidden/solo  
/hidden/states  
/usb/dirnames
```

**Bug in FW 2.08 and 2.10:** A Channel preset created from a matrix (mtx) channel does not report the right items: the preset reports a "LowCut" section flag instead of reporting a "Dyn" section. The `/preamp` section (which is not reported with a presence flag) is present but incomplete.

For further information about the OSC protocol please visit <http://opensoundcontrol.org>

Some text data and Effects pictures in this document: © 2012-2015 MUSIC Group IP Ltd. All rights reserved.

Additional data, tests, program examples and text by Patrick-Gilles Maillot. © 2014-2015

All information in this document is subject to change without any further notice.