UNOFFICIAL X32/M32
OSC REMOTE PROTOCOL

OSC protocol implementation for the
X32/M32 Digital Mixing Console families
Applies to console FW ver 4.0 and later

Initiated from version 1.01 (Oct-17-2012)
version 4.06 – 09 (Mar 17, 2022)

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.

I have 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 want to thank X32 forums well known **Paul Vannatto** for his invaluable support, generous time and advice in reviewing early versions of this document.

As you read through this document, you may like a "hands on" experience with testing OSC commands, it is recommended 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 OSC data back from X32/M32.

> ***X32_Command***[1] is a terminal based utility running on Windows, Linux, OSX and Raspberry platforms, supporting batch and interactive modes, timed commands, multi-tag parameters, and also scenes, snippets, and presets. Download it from https://sites.google.com/site/patrickmaillot/x32.
>
> ***X32 Live Toolbox***[2] 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/

With my purchase of an X32 digital mixer and as I started to find out more about OSC and ways to achieve more with the X32 via programs, I have spent quite some time designing and coding several utilities and applications for the M/X32 family of systems. Late 2015, I decided to open-source the code for the programs I wrote. These can be found at https://github.com/pmaillot/X32-Behringer. I'll continue to add programs as I finally "clean" them before publishing.

Note: Starting with X32 FW release 4.0, this manual will follow the X32 FW release numbering rather than its own numbering scheme; For example, *"version 4.02 – 01 (Jan 12, 2020)"* refers to update 01 of the document for FW 4.02, issued on Jan 12th, 2020.

Patrick-Gilles Maillot

[1] X32_Command: © 2014-2018 Patrick-Gilles Maillot
[2] X32 Live Toolbox: © 2014-2018 Paul Vannatto

Contents

DESCRIPTION

X32 & M32 represent a family of digital mixers (Core[3], Rack, Producer, Compact and Standard) which 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[4].

MIDI is also supported by the console and this is addressed later in this document, although the main focus hereafter is on OSC, using faster ethernet connections.

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.

Due to the nature of UDP communications, buffer overflows situations should be taken into consideration. A typical example of critical situation is sending large numbers of */node* requests to an X32 connected to a 2.4GHz/54Mb/s WIFI router. The 100Mb/s link between the X32 and the router will enable the X32 to send a lot more data than what can be propagated via WIFI by the router to connected clients, with possibly missing data at the client level due to UDP packets being silently lost by the router. Indeed, no errors will be reported in UDP for loss of data.

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 notification messages 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.

X32 internal variables are driving the behavior of the console. These can be read (Get) or written (Set) with OSC commands mapping variables with addressable parameters. Parameters are internally organized in logical groups, I will refer to as "**X32nodes**". X32nodes are widely used in scenes, snippets, and presets. They can be read using the */node* command presented later in this document[5]. X32nodes can be written or sent to X32 using the */* command, also presented later in this document. Parameters of an X32node can also be updated as a group (complete or not) by using the combined (multiple Type Tags) form of OSC Set commands.

[3] The X32 Core has been discontinued during the 4th quarter of 2017
[4] See Appendix for an example of communication program
[5] See chapter "X32nodes (/node & /) commands"

The list of <OSC Address Pattern> parameters commands, enabling an interactive control of all features of the X32/M32 mixer family is listed below:

<OSC Address Pattern> :=

```
/ | /-action | /add | /auxin | /batchsubscribe | /bus | /ch | /config |
/copy | /dca | /delete | /formatsubscribe | /fx | /fxrtn | /headamp |
/info | /-insert | /-libs | /load | /main/m | /main/st | /meters | /mtx |
/node | /outputs | /-prefs | /rename | /renew | /save | /-show | /showdump |
/-stat | /status | /subscribe | -undo | /unsubscribe | /-urec | /-usb |
/xinfo | /xremote
```

In this document, many commands to X32/M32 and replies from the console are shown as text in the following forms:

<command> <format> <parameters>

Or

<command>~~~<format>~~~<parameter> <parameter>…

The first form, for example: `/ch/01/mix/fader ,f [0.7]` represents a command you would type in, using the **X32_Command** utility mentioned earlier in this document.

The second form, for example: `/ch/01/mix/fader~~~~,f~~[0.7]` represents a command sent by the X32_Command utility mentioned earlier in this document. In that case, the number of ~ characters accurately represent the number of Null bytes (or \0) sent along with the command to respect the OSC protocol format.

This is further detailed in the coming pages.

Client initiated messages (client → X32 console)

Operation	OSC address	Parameters	Comments
Info request	/info /xinfo	None	Server responds with */info* message Server responds with */xinfo* message
Status request	/status	None	Server responds with */status* message
Set X32 parameter	<OSC Address Pattern>	<string | int | float | blob value>	Sets the value of a console parameter, e.g.: */ch/01/mix/fader~~~~,f~~<float>* If it exists and value is in range, the new value takes place in the X32.
Get X32 parameter	<OSC Address Pattern>	None	Requests the value of a console parameter, e.g. */ch/01/mix/fader~~~~* If it exists , the current value is echoed back by server, e.g.: */ch/01/mix/fader~~~~,f~~<float>*
Set X32 node data	/	<string>	Updates the values of a set of console parameters. A full set of X32node values can be sent to the server as plain text and matching /node formats, e.g.: */~~~,s~~-prefs/iQ/01 none Linear 0~~*
Get X32 node data	/node	<string>	Requests the values of a set of console parameters, e.g.: */node~~~,s~~-prefs/iQ/01~~~~* 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.: *node~~~,s~~*`/-prefs/iQ/01 none Linear 0\n~~~~`
Get X32 meters	/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. */meters ,s meters/1* 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 X32	/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: */subscribe ,s /-stat/solosw/01* or */subscribe ,si /-stat/solosw/01 1* Will report about 200 updates of the state of solo switch for channel 01 over the span of 10s. */subscribe ,si /-stat/solosw/01 50* Will report about 4 updates of the state of solo switch for channel 01
Subscribe to data formats	/formatsubscribe	<string>…<string><int>…<int>	Client describes to X32 server what information it is interested in receiving, e.g.: */formatsubscribe ,ssiii /mfm_c /dca/*/on 1 8 8*

from server			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.: */batchsubscribe ,ssiii /x_meters_0 /meters/0 0 69 1* Reports a blob of 70 floats for about 10s. */batchsubscribe ,ssiii /x_meters_8 /meters/8 0 5 1* Reports a blob of 6 floats for about 10s. */batchsubscribe ,ssiii /mfm_a /mix/on 0 63 8* 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~~~*
Register for updates	/xremote	None	Triggers X32 to send all parameter changes to maximum four active clients. Timeout is 10 seconds, e.g. the */xremote* command has to be renewed before this delay in order to avoid losing information from The X32 console.

Multiple client management

A single X32 can manage updates from and to several simultaneous UDP clients.

In order to keep being synchronized with changes happening at the X32 level, either from a change at the desk itself or requested by another remote client, each client must register for receiving updates from the X32. This is possible with the */xremote* command.

After receiving an */xremote* command, the X32 will update the client with changes taking place in the X32, such as fader movements, bank change requests, and screen updates. Some changes or user actions will not be reported as they do not directly affect the connected clients or result in changes that are strictly local to the X32/M32, such as pressing on one of the view buttons of the Standard X32/M32.

Registering for desk updates with a */xremote* command maintains updates for 10 seconds, after which a new */xremote* command should be issued by the client to keep the updating process alive.
Please refer to the examples given at the end of this document on how to use */xremote* in client applications (for example X32Saver.c (Linux or Windows), X32 data echo in Go)

Note: other commands such as */subscribe*, */formatsubscribe*, */batchsubscribe* also enable receiving regular updates form the server; details are available in the paragraph “Subscribing to X32/M32 Updates”.

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)
	/xinfo	<string network address> <string network name> <string console_model> <string console_version>	*/xinfo~~,ssss~~~192.168.1.62~~~~X32-02-4A-53~~~~X32~3.04~~~~*
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	<OSC Address Pattern>	<string | int | float>	If */xremote* is active, the X32 console echoes the value of a console parameter in response to a set command from another client or X32 parameter change, e.g. */-stat/solosw/01~~~~,i~~[1]* */-stat/solo~,i~~[1]* */ch/01/mix/01/pan~~~,f~~[1.0000]*

X32/M32 OSC Protocol Parameters

The table below lists the type and associated characteristics of parameters used for <OSC Address Pattern> and X32node commands.

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 padded to a multiple of 4 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 [-90.0…10.0 (+10 dB), steps]: 4 'linear' dB ranges: 0.0…0.0625 (-oo, -90…-60 dB), 0.0625…0.25 (-60…-30 dB), 0.25…0.5 (-30…-10dB) and 0.5…1.0 (-10…+10dB) *(see conversion help in appendix)*
	%int	An int corresponding to the bitwise OR of multiple bits (0 or 1)

Type rules (Get/Set parameter) and data formatting

With very few exceptions (clearly mentioned in this document when needed), the X32/M32 follow the guidelines as set by the Open Sound Control (OSC) 1.0[6], implementing the 4 basic OSC type tags for int32, float32, string, and blob.

- 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 a 4-byte padded type tag string, and if a non-empty type tag string is present, one or more 4-byte aligned/padded arguments.

The OSC 1.0 specification mentions that older implementations of OSC may omit the OSC type tag string, and OSC implementations should be robust in the case of a missing OSC type tag string, which is the case of X32/M32 systems.

Examples:

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

`/info~~~,~~~` correct format (OSC 1.0 compliant) command

The following will also work

`/info~~~` non OSC 1.0 compliant command, but accepted as older form of OSC

And the reply from different X32 or M32 systems (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~~~~`
X32 Compact: `/info~~~,ssss~~~V2.05~~~osc-server~~X32C~~2.12~~~~`
X32 Producer: `/info~~~,ssss~~~V2.05~~~osc-server~~X32P~2.12~~~~`
X32 Core: `/info~~~,ssss~~~V2.05~~~osc-server~~X32CORE~2.12~~~~`
M32 Standard: `/info~~~,ssss~~~V2.05~~~osc-server~~M32~2.12~~~~`
M32 Compact: `/info~~~,ssss~~~V2.05~~~osc-server~~M32C~2.12~~~~`
M32 Rack: `/info~~~,ssss~~~V2.05~~~osc-server~~M32R~2.12~~~~`

Note: Using UDP port 10024 (10023 for X32 family members), the XAir systems will return similar messages, as follows:

XR18: `/info~~~,ssss~~~V0.04~~~XR18-1D-DA-B4~~~XR18~~~~1.12~~~~`
XR16: `/info~~~,ssss~~~V0.04~~~XR16-1D-DA-B4~~~XR16~~~~1.12~~~~`
XR12: `/info~~~,ssss~~~V0.04~~~XR12-1D-DA-B4~~~XR12~~~~1.12~~~~`

[6] Please refer to http:// http://opensoundcontrol.org/ for further information on the OSC full spec.

An OSC command with a single type tag string and argument:

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

An OSC command with a more complex tag string and multiple arguments:[7]

```
/ch/01/eq/1 ,ifff [2] [0.2650] [0.5000] [0.4648]
```

This is equivalent to the following 4 simpler commands:

```
/ch/01/eq/1/t~~~,i~~[     2]
/ch/01/eq/1/f~~~,f~~[0.2650]
/ch/01/eq/1/g~~~,f~~[0.5000]
/ch/01/eq/1/q~~~,f~~[0.4648]
```

Or in hexadecimal for the last command:

```
 / c h / 0 1 / e q / 1 / q ~ ~ ~ , f ~ ~[0.4648]
2f6368 2f3031 2f6571 2f31 2f71 00000002 c66 0000 3eedfa44
```

Where `3eedfa44` is the hex for a 32bit float, big endian representation of `0.4648`, and where ~ stands for null character `(\0)`

Responses from X32/M32:

Sending to 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 to 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 to 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:

```
2f6678 2f34 2f70617 2f 3233 00000000 2c66 0000 3f000000
```

Special considerations for the 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]
```

in hexadecimal:

```
2f63682f30312f676174652f6d6f6465 00000000 2c73 0000 4741544500 000000
```

or

```
2f63682f30312f676174652f6d6f6465 00000000 2c69 0000 00000003
```

[7] In the case of X32/M32 "node" commands, this only applies to combinations of int or floats (`,i` or `,f`); strings (`,s`) sent to a node address (rather than a parameter address) are interpreted differently (internally used for X32-edit). As a result of such choice, the command `/ch/[01..32]/config ,siii [name] [1] [3] [1]`, although semantically correct and OSC compliant, does not work on X32/M32 when it does work fine on XAIR series.

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
```

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 point 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 value.

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, for example in the case of OSC data sent over MIDI Sysex commands[8].

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

Float type.
In “standard” OSC commands using floating point data parameters, floats are encoded as big endian 32bit floats, with a value between 0.0 and 1.0. For example, a volume variation on channel 01 to 3dB will be sent as follows:

```
/ch/01/mix/fader~~~~,f~~[0.8250]
```

or

```
2f63682f30312f6d69782f666164657200000000 2c660000 3f5334cd
```

Similarely, a pan change for channel 02 to half right will be set as follows:

```
/ch/02/mix/pan~~,f~~[0.7500]
```

or

```
2f63682f30322f6d69782f70616e0000 2c660000 3f400000
```

What is different in the case of floats is that the X32node-like commands for the commands above will enable setting parameters to their actual range, and not mapped to [0.0 ... 1.0], so for example and the case of the two examples above for volume and pan control, we can actually send the foowing:

```
/~~~,s~~/ch/01/mix/fader  3~~
```

or

```
2f000000 2c730000 2f63682f30312f6d69782f666164657220 33 0000
```

Above: setting volume fader to 3dB; volme range is [-90, +10, 1024], a pseudo log scale from -90dB to +10 dB, in 1024 steps. Below are additional examples for values 10, 0, -5, -90, and -20.5, respectively:

```
/ ~ ~ ~ , s ~ ~ / c h / 0 1 / m i x / f a d e r   1 0 ~
2f0000002c7300002f63682f30312f6d69782f666164657220313000
```

[8] See MIDI Connections chapter and appendix pages for section on sending OSC commands over MIDI SYSEX messages

```
/ ~ ~ ~ , s ~ ~ / c h / 0 1 / m i x / f a d e r   0 ~ ~
2f0000002c7300002f63682f30312f6d69782f66616465722030000 0

/ ~ ~ ~ , s ~ ~ / c h / 0 1 / m i x / f a d e r   - 5 ~
2f0000002c7300002f63682f30312f6d69782f6661646572202d3500

/ ~ ~ ~ , s ~ ~ / c h / 0 1 / m i x / f a d e r   - 9 0 ~ ~ ~ ~
2f0000002c7300002f63682f30312f6d69782f6661646572202d393000000000

/ ~ ~ ~ , s ~ ~ / c h / 0 1 / m i x / f a d e r   - 2 0 . 5 ~ ~
2f0000002c7300002f63682f30312f6d69782f6661646572202d32302e350000
```

```
/~~~,s~~/ch/02/mix/pan 50~~~
```

or

```
2f000000 2c73 0000 2f63682f30322f6d69782f70616e 20 3530 000000
```

Above: setting channel 02 pan to “right, 50% level “; pan range is a linear scale in the range of -100.0 to +100.0 in steps of 2.0

Later in this document, you will find tables of all known X32/M32 OSC commands.

These will contain the format, the parameter type(s) and value range(s) of each possible command:

- Possible *ints*, *%ints*, *enums* and *strings* values are as stated in the parameter ranges.
- For binary *floats*, the data sent to or returned from the X32/M32 is always in the range [0.0 ... 1.0], but the parameter range values will be shown as *level*, *logf*, or *linf* types with the min and max values along with the step or number of steps values, in full text format (ex: [-100.0, +100.0, 2.0]), to help in the case you would prefer to use the X32node style notation to send data to your X32/M32.

MIDI Connections[9]

This Document is all about OSC over ethernet. Nevertheless, the X32/M32 familly of devices can be connected to MIDI and send or receive commands using the MIDI protocol. Some commands are direct (using the standard MIDI controllers, channels and parameters data) and can provide a very simple way to control some of the features of the X32/M32.

Other commands use the MIDI SYSEX standard extensions; Using SYSEX, most (not all) of the X32/M32 OSC commands can be sent over MIDI; This is a major advantage for people who want to control with a finer granularity their device, yet do not have or want to invest in OSC programming. This document then becomes quite useful as OSC commands have to be coded over SYSEX to be sent as MIDI protocol.

The user should not expect the same performance when using MIDI as when using OSC. Indeed, ethernet is a much faster protocol than MIDI, but MIDI has solid advantages such as real-time control, much shorter messages for certain commands, and a very wide acceptance within the Music community, offering a large set of devices that can undertand, manipulate and send or receive MIDI.

This document contains several appendix pages explaining:

- The standard, direct MIDI commands understood by X32/M32 systems
- How to code OSC messages within a MIDI SYSEX command

[9] See appendix pages 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 typical format for */meters* is as follows:

/meters ,sii <meter request and parameters (see below)> *[time_factor]*

The highlighted *sii* tags are used for the meter request, comprising a string and two ints depending on the meter request type. The command is active for about 10s. Possible meter requests are given in the following pages. The last int of the command is used to control the number of times the requestor will receive meter values.

time_factor is a value between 1 and 99 setting the interval between two successive meters messages to *50ms * time_factor*. Any value of time_factor outside or [1, 99] is equivalent to 1. For a timespan of 10s, the number of updates can be calculated based on the value of *time_factor* as below:

time_factor:	<2 or >99 → 200 updates
	2 → 100 updates
	[…]
	40 → 5 updates
	80 to 99 → 3 updates

The data returned by the X32/M32 server for */meters* is an OSC-blob, an abitrary set of 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 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:
2f6d65746572732f360000002c62000000000014040000000fd1d2137fdff7f3f0000803f6ebbd534
/ m e t e r s / 6 ~ ~ ~ , b ~ ~[int1][int2][nfloat][nfloat][nfloat][nfloat]

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 a single OSC 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 a single OSC 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 a single OSC 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 a single OSC 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> 0: DCA 1-8; 1: mix bus 1-8; 2: mix bus 9-16; 3: matrixes
2 main LR
1 mono M/C
➝ returns 27 float values as a single OSC blob

/meters/6 <channel_id>

Returns meter values from **Channel Strip Meters** (post gain/trim, gate, dyn gain reduction and post-fade):
4 channel strip meters: <channel_id> channel 0…71]
➝ returns 4 float values a as single OSC 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 a single OSC blob

/meters/8

Returns meter values from **Matrix Send** meters:
6 Matrix send meters
➝ returns 6 float values (from Matrix sends 1-6) as a single OSC 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 a single OSC 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 a single OSC blob

/meters/12

Returns meter values from the **Recorder** page
➝ returns 4 float values (RecInput L, RecInput R, Playback L, Playback R) as a single OSC blob

/meters/13

Returns meter values from the **METERS** page
32 input channels
8 aux returns
4x2 st fx returns
➝ 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 a single OSC 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

/meters/16

Used for **comp** and **automix**
➝ returns 48 32bits values as a single OSC blob.

The first 44 values are 32bits values returned are representing:

- 32 channel gate gains,
- 32 channel comp gains,
- 16 bus comp gains,
- 6 matrix comp gains,
- 2 (L/R and Mono) comp gains

All data snt as *little endian coded short ints*; each short int value represents a floating-point level in the range [*0, 1.0*], by dividing the short int (converted to float) by *32767.0*.

The 4 last floats represent 8 automix (channel 01...08) gains coded on successive shorts as $Log_2(value) * 256$.

X32/M32 ↔ Client communications

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

Configuration (/config) data

<path>	<type>	<range>	<unit>
config data			
/config/chlink/1-2 /config/chlink/3-4 /config/chlink/5-6 /config/chlink/7-8 /config/chlink/9-10 /config/chlink/11-12 /config/chlink/13-14 /config/chlink/15-16 /config/chlink/17-18 /config/chlink/19-20 /config/chlink/21-22 /config/chlink/23-24 /config/chlink/25-26 /config/chlink/27-28 /config/chlink/29-30 /config/chlink/31-32	enum	*{OFF, ON}*, int with value 0 or 1 indicating whether channels pairs are linked or not	
/config/auxlink/1-2 /config/auxlink/3-4 /config/auxlink/5-6 /config/auxlink/7-8	enum	*{OFF, ON}*, int with value 0 or 1 indicating whether channels pairs are linked or not	
/config/fxlink/1-2 /config/fxlink/3-4 /config/fxlink/5-6 /config/fxlink/7-8	enum	*{OFF, ON}*, int with value 0 or 1 indicating whether channels pairs are linked or not	
/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	enum	*{OFF, ON}*, int with value 0 or 1 indicating whether channels pairs are linked or not	
/config/mtxlink/1-2 /config/mtxlink/3-4 /config/mtxlink/5-6	enum	*{OFF, ON}*, int with value 0 or 1 indicating whether channels pairs are linked or not	

/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	[-90.0…10.0 (+10 dB), 161] [10]	dB
/config/solo/source	enum	int [0…6] representing *{OFF, LR, LR+C, LRPFL, LRAFL, AUX56, AUX78}*	
/config/solo/sourcetrim	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}*, int with value 0 or 1	
/config/solo/dcamode	enum	*{PFL, AFL}*, int with value 0 or 1	
/config/solo/exclusive	enum	*{OFF, ON}*, int with value 0 or 1	
/config/solo/followsel	enum	*{OFF, ON}*, int with value 0 or 1	
/config/solo/followsolo	enum	*{OFF, ON}*, int with value 0 or 1	
/config/solo/dimatt	linf	[-40.000, 0.000, 1.000][11]	dB
/config/solo/dim	enum	*{OFF, ON}*, int with value 0 or 1	
/config/solo/mono	enum	*{OFF, ON}*, int with value 0 or 1	
/config/solo/delay	enum	*{OFF, ON}*, int with value 0 or 1	
/config/solo/delaytime	linf	[0.300, 500.000, 0.100]	ms
/config/solo/masterctrl	enum	*{OFF, ON}*, int with value 0 or 1	
/config/solo/mute	enum	*{OFF, ON}*, int with value 0 or 1	
/config/solo/dimpfl	enum	*{OFF, ON}*, int with value 0 or 1	
/config/talk/enable	enum	*{OFF, ON}*, int with value 0 or 1	
/config/talk/source	enum	*{INT, EXT}*, int with value 0 or 1	
/config/talk/A/level /config/talk/B/level	level	[-90.0…10.0 (+10 dB), 161]	dB
/config/talk/A/dim /config/talk/B/dim	enum	*{OFF, ON}*, int with value 0 or 1	
/config/talk/A/latch /config/talk/B/latch	enum	*{OFF, ON}*, int with value 0 or 1	
/config/talk/A/destmap /config/talk/B/destmap	%int	[0, 262143] (18 bits bitmap)	
/config/osc/level	level	[-90.0…10.0 (+10 dB), 161]	dB
/config/osc/f1 /config/osc/f2	logf	[20.000, 20000, 121][12]	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,*	

[10] See Appendix section for detailed values
[11] See Appendix section for detailed values
[12] See Appendix section for detailed values

		WHITE}	
/config/osc/dest	int	int with value [0…25] representing *{MixBus1…16, L, R, L+R, M/C, Matrix1…6}*	
/config/userrout/out/01…48[13]	int	int with value [0…208] representing OFF to Mon R as follows: *0 OFF* *1…32 Local In 1…32* *33…80 AES50-A 1…48* *81…128 AES50-B 1…48* *129…160 Card In 1…32* *161…166 Aux In 1…6* *167 TB Internal* *168 TB External* *169…184 Outputs 1…16* *185…200 P16 1…16* *201…206 AUX 1…6* *207 Monitor L* *208 Monitor R*	
/config/userrout/in/01…32[14]	int	int with value [0…168] representing OFF to TB ext as follows: *0 OFF* *1…32 Local In 1…32* *33…80 AES50-A 1…48* *81…128 AES50-B 1…48* *129…160 Card In 1…32* *161…166 Aux In 1…6* *167 TB Internal* *168 TB External*	
/config/routing/routswitch	enum	*{REC, PLAY}*: routing *0: Rec [default value]* *1: Playback* This command will automatically select the */config/routing/IN* or the */config/routing/PLAY* parameter blocks below, respective of the OSC parameter value *[0]* or *[1]*	
/config/routing/IN/1-8 /config/routing/IN/9-16 /config/routing/IN/17-24 /config/routing/IN/25-32	enum	int with value [0…23] representing *{AN1-8, AN9-16, AN17-24, AN25-32, A1-8, A9-16, A17-24, A25-32, A33-40, A41-48, B1-8, B9-16, B17-24, B25-32, B33-40, B41-48, CARD1-8, CARD9-16, CARD17-24, CARD25-32, UIN1-8, UIN9-16, UIN17-24, UIN25-32}*	
/config/routing/IN/AUX	enum	int with value [0…15] representing *{AUX1-4[15], AN1-2, AN1-4, AN1-6, A1-2, A1-4 A1-6, B1-2, B1-4, B1-6, CARD1-2, CARD1-4, CARD1-6, UIN1-2, UIN1-4, UIN1-6}*	
/config/routing/AES50A/1-8 /config/routing/AES50A/9-16 /config/routing/AES50A/17-24	enum	int with value [0…35] representing *{AN1-8, AN9-16, AN17-24, AN25-32, A1-8, A9-16, A17-24, A25-32, A33-40, A41-*	

[13] FW 4.0 and above

[14] FW 4.0 and above

[15] It really is AUX1-6, but needs to stay AUX1-4 for backward compatibility

/config/routing/AES50A/25-32 /config/routing/AES50A/33-40 /config/routing/AES50A/41-48 /config/routing/AES50B/1-8 /config/routing/AES50B/9-16 /config/routing/AES50B/17-24 /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		*48, B1-8, B9-16, B17-24, B25-32, B33-40, B41-48, CARD1-8, CARD9-16, CARD17-24, CARD25-32, OUT1-8, OUT9-16, P161-8, P16 9-16, AUX1-6/Mon, AuxIN1-6/TB, UOUT1-8, UOUT9-16, UOUT17-24, UOUT25-32, UOUT33-40, UOUT41-48, UIN1-8, UIN9-16, UIN17-24, UIN25-32}*	
/config/routing/OUT/1-4 /config/routing/OUT/9-12	enum	int with value [0…35] representing *{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, B33-36, B41-44, CARD1-4, CARD9-12, CARD17-20, CARD25-28, OUT1-4, OUT9-12, P161-4, P169-12, AUX/CR, AUX/TB, UOUT1-4, UOUT9-12, UOUT17-20, UOUT25-28, UOUT33-36, UOUT41-44, UIN1-4, UIN9-12, UIN17-20, UIN25-28}*	
/config/routing/OUT/5-8 /config/routing/OUT/13-16	enum	int with value [0…35] representing *{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, UOUT5-8, UOUT13-16, UOUT21-24, UOUT29-32, UOUT37-40, UOUT45-48, UIN5-8, UIN13-16, UIN21-24, UIN29-32}*	
/config/routing/PLAY/1-8 /config/routing/PLAY/9-16 /config/routing/PLAY/17-24 /config/routing/PLAY/25-32	enum	int with value [0…23] representing *{AN1-8, AN9-16, AN17-24, AN25-32, A1-8, A9-16, A17-24, A25-32, A33-40, A41-48, B1-8, B9-16, B17-24, B25-32, B33-40, B41-48, CARD1-8, CARD9-16, CARD17-24, CARD25-32, UIN1-8, UIN9-16, UIN17-24, UIN25-32}*	
/config/routing/PLAY/AUX	enum	int with value [0…15] 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, UIN1-2, UIN1-4, UIN1-6}*	
/config/userctrl/A/color /config/userctrl/B/color /config/userctrl/C/color	enum	int with value [0…15] representing *{OFF, RD, GN, YE, BL, MG, CY, WH, OFFi, RDi, GNi, YEi, BLi, MGi, CYi, WHi}*	
/config/userctrl/A/enc/1…4 /config/userctrl/B/enc/1…4 /config/userctrl/C/enc/1…4	string	string up to 7 characters representing encoder assignment and unction. See User Control Chapter for full details.	
/config/userctrl/A/btn/5…12	string	User assignable set A, B, or C: Button 5 to 12	

/config/userctrl/B/btn/5…12 /config/userctrl/C/btn/5…12		See User Control Chapter for full details.	
/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	
/config/amixenable/X…Y	enum	Automix Enable for group X or Y *{OFF, ON}*, int with value 0 or 1	
/config/dp48/scope[16]	%int	[0, 15] (4 bits bitmap): *Bit 0: Group Name* *Bit 1: Group Assign* *Bit 2: [implies bit 1] Chan Level* *Bit 3: [implies bit 1] Chan Pan*	
/config/dp48/broadcast	int	int value 0 or 1: *0: No-op* *1: Broadcast scope (TBV); the console will reply with a value of 0*	
/config/dp48/aesAB	enum	*{0, 1}* representing AESA (0) or AESB (1) for selecting the broadcast target port	
/config/dp48/link/01…24	int	int with value [0…1] representing the "link pair" status of each pair of channels [1-2], [3-4], …, [47-48]	
/config/dp48/assign/01…48	int	int with value [0…12] representing the group of to 12 being assigned	
/config/dp48/grpname/01…12	string	string up to 8 characters representing the name of the DP48 group	

[16] All dp48 related commands: FW 4.0 and above

Channel (/ch) data

channel [01…32] (channel id 0…31)			
/ch/[01…32]/config/name	string	A 12-character max string representing the input channel name	
/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[17]	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}*, int with value 0 or 1	
/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)[18]	dB
/ch/[01…32]/preamp/invert	enum	*{OFF, ON}*, int with value 0 or 1	
/ch/[01…32]/preamp/hpon	enum	*{OFF, ON}*, int with value 0 or 1; 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][19]	Hz
/ch/[01…32]/gate/on	enum	*{OFF, ON}*, int with value 0 or 1	
/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][20]	dB
/ch/[01…32]/gate/attack	linf	[0.000, 120.000, 1.000]	ms
/ch/[01…32]/gate/hold	logf	[0.020, 2000, 101][21]	ms
/ch/[01…32]/gate/release	logf	[5.000, 4000.000, 101][22]	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][23]	Hz
/ch/[01…32]/dyn/on	enum	*{OFF, ON}*, int with value 0 or 1	
/ch/[01…32]/dyn/mode	enum	*{COMP, EXP}*, int with value 0 or 1	
/ch/[01…32]/dyn/det	enum	*{PEAK, RMS}*, int with value 0 or 1	
/ch/[01…32]/dyn/env	enum	*{LIN, LOG}*, int with value 0 or 1	
/ch/[01…32]/dyn/thr	linf	[-60.000, 0.000, 0.500]	dB

[17] See /headamp chapter; X32 will return the actual headamp used as source using /-ha/xx/index.

[18] See Appendix section for detailed values

[19] See Appendix section for detailed values

[20] See Appendix section for detailed values

[21] See Appendix section for detailed values

[22] See Appendix section for detailed values

[23] See Appendix section for detailed values

/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]	
/ch/[01…32]/dyn/mgain	linf	[0.000, 24.000, 0.500][24]	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}*, int with value 0 or 1	
/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	enum	*{OFF, ON}*, int with value 0 or 1	
/ch/[01…32]/dyn/filter/on	enum	*{OFF, ON}*, int with value 0 or 1	
/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}*, int with value 0 or 1	
/ch/[01…32]/insert/pos	enum	*{PRE, POST}*, int with value 0 or 1	
/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}*, int with value 0 or 1	
/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][25]	dB
/ch/[01…32]/eq/[1…4]/q	logf	[10.000, 0.3, 72]	
/ch/[01…32]/mix/on	enum	*{OFF, ON}*, int with value 0 or 1	
/ch/[01…32]/mix/fader	level	[0.0…1.0(+10dB), 1024]	dB
/ch/[01…32]/mix/st	enum	*{OFF, ON}*, int with value 0 or 1	
/ch/[01…32]/mix/pan	linf	[-100.000, 100.000, 2.000]	
/ch/[01…32]/mix/mono	enum	*{OFF, ON}*, int with value 0 or 1	
/ch/[01…32]/mix/mlevel	level	[-90.0…10.0 (+10 dB), 161]	dB
/ch/[01…32]/mix/[01…16]/on	enum	*{OFF, ON}*, int with value 0 or 1	
/ch/[01…32]/mix/[01…16]/level	level	[-90.0…10.0 (+10 dB), 161]	dB
/ch/[01…32]/mix/01/pan /ch/[01…32]/mix/03/pan /ch/[01…32]/mix/05/pan /ch/[01…32]/mix/07/pan /ch/[01…32]/mix/09/pan	linf	[-100.000, 100.000, 2.000]	

[24] See Appendix section for detailed values

[25] See Appendix section for detailed values

/ch/[01…32]/mix/11/pan /ch/[01…32]/mix/13/pan /ch/[01…32]/mix/15/pan			
/ch/[01…32]/mix/01/type /ch/[01…32]/mix/03/type /ch/[01…32]/mix/05/type /ch/[01…32]/mix/07/type /ch/[01…32]/mix/09/type /ch/[01…32]/mix/11/type /ch/[01…32]/mix/13/type /ch/[01…32]/mix/15/type	enum	int [0…5] representing *{IN/LC, <-EQ, EQ->, PRE, POST, GRP}*	
/ch/[01…32]/mix/01/panFollow /ch/[01…32]/mix/03/panFollow /ch/[01…32]/mix/05/panFollow /ch/[01…32]/mix/07/panFollow /ch/[01…32]/mix/09/panFollow /ch/[01…32]/mix/11/panFollow /ch/[01…32]/mix/13/panFollow /ch/[01…32]/mix/15/panFollow	int	int [0…1][26]	
/ch/[01…32]/grp/dca	%int	[0, 255] (8 bits bitmap)	
/ch/[01…32]/grp/mute	%int	[0, 63] (6 bits bitmap)	
/ch/[01…32]/automix/group	enum	int [0…2] representing Channel's assignment to an Automix Group: *{OFF, X, Y}* This command is only effective on channels 01 to 08	
/ch/[01…32]/automix/weight	linf	[-12.000, 12.000, 0.500][27] This command is only effective on channels 01 to 08	dB

[26] FW 4.0 and above

[27] See Appendix section for detailed values

Aux In (/auxin) data

auxin [01…08] (channel id 32…39)			
/auxin/[01…08]/config/name	string	A 12-character max string representing the aux in channel name	
/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}*, int with value 0 or 1	
/auxin/[01…08]/eq/on	enum	*{OFF, ON}*, int with value 0 or 1	
/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
/auxin/[01…08]/eq/[1…4]/g	linf	[-15.000, 15.000, 0.250]	dB
/auxin/[01…08]/eq/[1…4]/q	logf	[10.000, 0.3, 72]	
/auxin/[01…08]/mix/on	enum	*{OFF, ON}*, int with value 0 or 1	
/auxin/[01…08]/mix/fader	level	[0.0…1.0(+10dB), 1024]	
/auxin/[01…08]/mix/st	enum	*{OFF, ON}*, int with value 0 or 1	
/auxin/[01…08]/mix/pan	linf	[-100.000, 100.000, 2.000]	
/auxin/[01…08]/mix/mono	enum	*{OFF, ON}*, int with value 0 or 1	
/auxin/[01…08]/mix/mlevel	level	[-90.0…10.0 (+10 dB), 161]	
/auxin/[01…08]/mix/[01…16]/on	enum	*{OFF, ON}*, int with value 0 or 1	
/auxin/[01…08]/mix/[01…16]/level	level	[-90.0…10.0 (+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 /auxin/[01…08]/mix/05/pan /auxin/[01…08]/mix/07/pan /auxin/[01…08]/mix/09/pan /auxin/[01…08]/mix/11/pan /auxin/[01…08]/mix/13/pan /auxin/[01…08]/mix/15/pan	linf	[-100.000, 100.000, 2.000]	
/auxin/[01…08]/mix/03/type /auxin/[01…08]/mix/05/type /auxin/[01…08]/mix/07/type /auxin/[01…08]/mix/09/type /auxin/[01…08]/mix/11/type /auxin/[01…08]/mix/13/type /auxin/[01…08]/mix/15/type	enum	int [0…5] representing *{IN/LC, <-EQ, EQ->, PRE, POST, GRP}*	
/auxin/[01…08]/mix/03/panFollow	int	int [0…1][28]	

[28] FW 4.0 and above

/auxin/[01…08]/mix/05/panFollow /auxin/[01…08]/mix/07/panFollow /auxin/[01…08]/mix/09/panFollow /auxin/[01…08]/mix/11/panFollow /auxin/[01…08]/mix/13/panFollow /auxin/[01…08]/mix/15/panFollow			
/auxin/[01…08]/grp/dca	%int	[0, 255] (8 bits bitmap)	
/auxin/[01…08]/grp/mute	%int	[0, 63] (6 bits bitmap)	

FX Return (/fxrtn) data

fxrtn [01…08] (channel id 40…47)			
/fxrtn/[01…08]/config/name	string	A 12-character max string representing the fx return channel name	
/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}*, int with value 0 or 1	
/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}*, int with value 0 or 1	
/fxrtn/[01…08]/mix/fader	level	[0.0…1.0(+10dB), 1024]	dB
/fxrtn/[01…08]/mix/st	enum	*{OFF, ON}*, int with value 0 or 1	
/fxrtn/[01…08]/mix/pan	linf	[-100.000, 100.000, 2.000]	dB
/fxrtn/[01…08]/mix/mono	enum	*{OFF, ON}*, int with value 0 or 1	
/fxrtn/[01…08]/mix/mlevel	level	[-90.0…10.0 (+10 dB), 161]	dB
/fxrtn/[01…08]/mix/[01…16]/on	enum	*{OFF, ON}*, int with value 0 or 1	
/fxrtn/[01…08]/mix/[01…16]/level	level	[-90.0…10.0 (+10 dB), 161]	dB
/fxrtn /[01…08]/mix/03/pan /fxrtn /[01…08]/mix/05/pan /fxrtn /[01…08]/mix/07/pan /fxrtn /[01…08]/mix/09/pan /fxrtn /[01…08]/mix/11/pan /fxrtn /[01…08]/mix/13/pan /fxrtn /[01…08]/mix/15/pan	linf	[-100.000, 100.000, 2.000]	
/fxrtn /[01…08]/mix/03/type /fxrtn /[01…08]/mix/05/type /fxrtn /[01…08]/mix/07/type /fxrtn /[01…08]/mix/09/type /fxrtn /[01…08]/mix/11/type /fxrtn /[01…08]/mix/13/type /fxrtn /[01…08]/mix/15/type	enum	int [0…5] representing *{IN/LC, <-EQ, EQ->, PRE, POST, GRP}*	
/fxrtn /[01…08]/mix/03/panFollow /fxrtn /[01…08]/mix/05/panFollow /fxrtn /[01…08]/mix/07/panFollow /fxrtn /[01…08]/mix/09/panFollow /fxrtn /[01…08]/mix/11/panFollow /fxrtn /[01…08]/mix/13/panFollow /fxrtn /[01…08]/mix/15/panFollow	int	int [0…1][29]	
/fxrtn/[01…08]/grp/dca	%int	[0, 255] (8 bits bitmap)	
/fxrtn/[01…08]/grp/mute	%int	[0, 63] (6 bits bitmap)	

[29] FW 4.0 and above

Bus (/bus) data

bus [01…16] (channel id 48…63)			
/bus/[01…16]/config/name	string	A 12-character max string representing the bus channel name	
/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}*, int with value 0 or 1	
/bus/[01…16]/dyn/mode	enum	*{COMP, EXP}*, int with value 0 or 1	
/bus/[01…16]/dyn/det	enum	*{PEAK, RMS}*, int with value 0 or 1	
/bus/[01…16]/dyn/env	enum	*{LIN, LOG}*, int with value 0 or 1	
/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	enum	*{OFF, ON}*, int with value 0 or 1	
/bus/[01…16]/dyn/filter/on	enum	*{OFF, ON}*, int with value 0 or 1	
/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}*, int with value 0 or 1	
/bus/[01…16]/insert/pos	enum	*{PRE, POST}*, int with value 0 or 1	
/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}*, int with value 0 or 1	
/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}*, int with value 0 or 1	
/bus/[01…16]/mix/fader	level	[0.0…1.0(+10dB), 1024]	dB

/bus/[01…16]/mix/st	enum	*{OFF, ON}*, int with value 0 or 1	
/bus/[01…16]/mix/pan	linf	[-100.000, 100.000, 2.000]	
/bus/[01…16]/mix/mono	enum	*{OFF, ON}*, int with value 0 or 1	
/bus/[01…16]/mix/mlevel	level	[0.0…1.0(+10dB), 161]	dB
/bus/[01…16]/mix/[01…06]/on	enum	*{OFF, ON}*, int with value 0 or 1	
/bus/[01…16]/mix/[01…06]/level	level	[0.0…1.0(+10dB), 161]	dB
/bus/[01…16]/mix/01/pan /bus/[01…16]/mix/03/pan /bus/[01…16]/mix/05/pan	linf	[-100.000, 100.000, 2.000]	
/bus/[01…16]/mix/01/type /bus/[01…16]/mix/03/type /bus/[01…16]/mix/05/type	enum	int [0…5] representing *{IN/LC, <-EQ, EQ->, PRE, POST}*	
/bus/[01…16]/mix/01/panFollow /bus/[01…16]/mix/03/panFollow /bus/[01…16]/mix/05/panFollow	int	int [0…1][30]	
/bus/[01…16]/grp/dca	%int	[0, 255] (8bits bitmap)	
/bus/[01…16]/grp/mute	%int	[0, 63] (6 bits bitmap)	

[30] FW 4.0 and above

Matrix (/mtx) data

mtx [01…06] (channel id 64…69)			
/mtx/[01…06]/config/name	string	A 12-character max string representing the matrix name	
/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}*, int with value 0 or 1	
/mtx/[01…06]/dyn/on	enum	*{OFF, ON}*, int with value 0 or 1	
/mtx/[01…06]/dyn/mode	enum	*{COMP, EXP}*, int with value 0 or 1	
/mtx/[01…06]/dyn/det	enum	*{PEAK, RMS}*, int with value 0 or 1	
/mtx/[01…06]/dyn/env	enum	*{LIN, LOG}*, int with value 0 or 1	
/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}*, int with value 0 or 1	
/mtx/[01…06]/dyn/mix	linf	[0, 100, 5]	%
/mtx/[01…06]/dyn/auto	enum	*{OFF, ON}*, int with value 0 or 1	
/mtx/[01…06]/dyn/filter/on	enum	*{OFF, ON}*, int with value 0 or 1	
/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}*, int with value 0 or 1	
/mtx/[01…06]/insert/pos	enum	*{PRE, POST}*, int with value 0 or 1	
/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}*, int with value 0 or 1	
/mtx/[01…06]/eq/[1…6]/type	enum	int [0…13] representing *{LCut, LShv, PEQ, VEQ, HShv, HCut, BU6, BU12, BS12, LR12, BU18, BU24, BS24, LR24}*. In some cases, eq 2 and eq 5 are ignored.	
/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}*, int with value 0 or 1	
/mtx/[01…06]/mix/fader	level	[0.0…1.0(+10dB), 1024]	dB

/mtx/[01…06]/grp/dca	%int	[0, 255] (8bits bitmap)	
/mtx/[01…06]/grp/mute	%int	[0, 63] (6 bits bitmap)	

Main Stereo (/main/st) data

main stereo (channel id 70)			
/main/st/config/name	string	A 12-character max string representing the main LR channel name	
/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}*, int with value 0 or 1	
/main/st/dyn/mode	enum	*{COMP, EXP}*, int with value 0 or 1	
/main/st/dyn/det	enum	*{PEAK, RMS}*, int with value 0 or 1	
/main/st/dyn/env	enum	*{LIN, LOG}*, int with value 0 or 1	
/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}*, int with value 0 or 1	
/main/st/dyn/mix	linf	[0, 100, 5]	%
/main/st/dyn/auto	enum	*{OFF, ON}*, int with value 0 or 1	
/main/st/dyn/filter/on	enum	*{OFF, ON}*, int with value 0 or 1	
/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}*, int with value 0 or 1	
/main/st/insert/pos	enum	*{PRE, POST}*, int with value 0 or 1	
/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}*, int with value 0 or 1	
/main/st/eq/[1…6]/type	enum	int [0…13] representing *{LCut, LShv, PEQ, VEQ, HShv, HCut, BU6, BU12, BS12, LR12, BU18, BU24, BS24, LR24}*. In some cases, eq 2 and eq 5 are not available.	
/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}*, int with value 0 or 1	
/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}*, int with value 0 or 1	
/main/st/mix/[01…06]/level	level	[0.0…1.0(+10dB), 161]	dB
/main/st/mix/01/pan /main/st/mix/03/pan /main/st/mix/05/pan	linf	[-100.000, 100.000, 2.000]	
/main/st/mix/01/type /main/st/mix/03/type /main/st/mix/05/type	enum	int [0…5] representing *{IN/LC, <-EQ, EQ->, PRE, POST}*	
/main/st/mix/01/panFollow /main/st/mix/03 panFollow /main/st/mix/05 panFollow	int	int [0…1][31]	
/main/st/grp/dca	%int	[0, 255] (8bits bitmap)	
/main/st/grp/mute	%int	[0, 63] (6 bits bitmap)	

[31] FW 4.0 and above

Main Mono (/main/m) data

main mono (channel id 71)			
/main/m/config/name	string	A 12-character max string representing themain mono channel name	
/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}*, int with value 0 or 1	
/main/m/dyn/mode	enum	*{COMP, EXP}*, int with value 0 or 1	
/main/m/dyn/det	enum	*{PEAK, RMS}*, int with value 0 or 1	
/main/m/dyn/env	enum	*{LIN, LOG}*, int with value 0 or 1	
/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}*, int with value 0 or 1	
/main/m/dyn/mix	linf	[0, 100, 5]	%
/main/m/dyn/auto	enum	*{OFF, ON}*, int with value 0 or 1	
/main/m/dyn/filter/on	enum	*{OFF, ON}*, int with value 0 or 1	
/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
/main/m/insert/on	enum	*{OFF, ON}*, int with value 0 or 1	
/main/m/insert/pos	enum	*{PRE, POST}*, int with value 0 or 1	
/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}*, int with value 0 or 1	
/main/m/eq/[1…6]/type	enum	int [0…13] representing *{LCut, LShv, PEQ, VEQ, HShv, HCut, BU6, BU12, BS12, LR12, BU18, BU24, BS24, LR24}*. In some cases, eq 2 and eq 5 are not available.	
/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}*, int with value 0 or 1	
/main/m/mix/fader	level	[0.0…1.0(+10dB), 1024]	dB

/main/m/mix/[01…06]/on	enum	*{OFF, ON}*, int with value 0 or 1	
/main/m/mix/[01…06]/level	level	[0.0…1.0(+10dB), 161]	dB
/main/m/mix/01/pan /main/st/mix/03/pan /main/st/mix/05/pan	linf	[-100.000, 100.000, 2.000]	
/main/m/mix/01/type /main/m/mix/03/ type /main/m/mix/05/ type	enum	int [0…5] representing *{IN/LC, <-EQ, EQ->, PRE, POST}*	
/main/m/mix/01/panFollow /main/m/mix/03 panFollow /main/m/mix/05 panFollow	int	int [0…1][32]	
/main/m/grp/dca	%int	[0, 255] (8bits bitmap)	
/main/m/grp/mute	%int	[0, 63] (6 bits bitmap)	

[32] FW 4.0 and above

DCA groups (/dca) data

dca groups (no channel id)			
/dca/[1…8]/on	enum	*{OFF, ON}*, int with value 0 or 1	
/dca/[1…8]/fader	level	[0.0…1.0(+10dB), 1024]	dB
/dca/[1…8]/config/name	string	A 12-character max string representing the DCA name	
/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}*[33]
/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}*[34]
/fx/[5…8]/par/[01…64]	linf/logf	Up to 64 parameters, depending on selected effect type. See Effect Parameters Chapter

[33] See Appendix for table of enum/name/type
[34] See Appendix for table of enum/name/type

Output sets (/output) data

outputs main [01…16]			
/outputs/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]/invert	enum	*{OFF, ON}*, int with value 0 or 1	
/outputs/main/[01…16]/delay/on	enum	*{OFF, ON}*, int with value 0 or 1	
/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/aux/[01…06]/invert	enum	*{OFF, ON}*, int with value 0 or 1	

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]/invert	enum	*{OFF, ON}*, int with value 0 or 1	
/outputs/p16/[01…16]/iQ/group	enum	int [0…2] representing the group the iQ speaker is associated to, in the range *{OFF, A, B}* *0: OFF* *1: A* *2: B*	
/outputs/p16/[01…16]/iQ/speaker	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}* *0: none* *1: iQ8* *2: iQ10* *3: iQ12* *4: iQ15* *5: iQ15B* *6: iQ18B*	

/outputs/p16/[01…16]/iQ/eq	enum	int [0…4] representing a frequency response setting for the respective speaker. Possible values are: *{Linear, Live, Speech, Playback, User}* *0: Linear (default setting)* *1: Live (typical live sound setting)* *2: Speech (optimal speech intelligibility setting)* *3: Playback (ideal setting for music playback)* *4: User (response curve set in the iQ speaker sub-menu)*	
/outputs/p16/[01…16]/iQ/model	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: *iQ8 : [0…5]: iQ8, E8, F8+, UPJunior, PS8, NuQ8-DP* *iQ10: [0…4]: iQ10, F10+, UPJ-1P, PS10-R2, NuQ10-DP* *iQ12: [0…7]: iQ12, E12, JF29NT, ELX112P, PRX612M,F12+, UPA-1P, NuQ12-DP* *iQ15: [0…7]: iQ15, JF59NT, ELX115P, PRX615M, F15+, UPQ-1P, PS15-R2, NuQ15-DP* *iQ15B: [0…3]: iQ15B, E15X, S15+, B-15DP* *iQ18B: [0…4]: iQ18B, ELX18P, PRX6118S, S18+, B-18DP*	

outputs AES [01…02]			
/outputs/aes/[01…02]/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/aes/[01…02]/pos	enum	int [0…8] representing *{IN/LC, IN/LC+M, <-EQ, <-EQ+M, EQ->, EQ->+M, PRE, PRE+M, POST}*	
/outputs/aes/[01…02]/invert	enum	*{OFF, ON}*, int with value 0 or 1	

outputs REC [01…02]			
/outputs/rec/[01…02]/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/rec/[01…02]/pos	enum	int [0…8] representing *{IN/LC, IN/LC+M, <-EQ, <-EQ+M, EQ->, EQ->+M, PRE, PRE+M, POST}*	

Headamp (/headamp) data

headamp [000…127]			
/headamp/[000…127]/gain	linf	[-12.000, 60.000, 0.500][35] */headamp index:* *000…031: local XLR inputs* *032…079: AES50 port A connected devices* *080…127: AES50 port B connected devices*	dB
/headamp/[000…127]/phantom	enum	*{OFF, ON}*, int with value 0 or 1 */headamp index:* *000…031: local XLR inputs* *032…079: AES50 port A connected devices* *080…127: AES50 port B connected devices*	
/-ha/[00…39]/index	int	*(Read only)* returns the actual headamp used as source for a given input [00…39] represent a channel index *0…31: channel 01…32* *32…39: aux 1…8* A value of *-1* is possible and typically happens when the X32 audio engine routing changes to an internal source such as the card slot.	

Inserts (/-insert) data[36]

insert			
/-insert/fx[1-8]L	int	Channel the FX L input[1…8] is inserted into	
/-insert/fx[1-8]R	int	Channel the FX R input [1…8] is inserted into	
/-insert/aux[1…6]	int	Channel the Aux input [1…6] is inserted into	

[35] See Appendix section for detailed values

[36] */node ,s –insert* reports 22 inserts in the following order: *fx1L, fx1R, fx2L, …, fx8R, aux1, …, aux6*

Show, Cue, Scene, Snippet, and Preset Management

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

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.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" (see below) 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. Starting with FW realease 4.02, X32/M32 support a special scene file that can be automatically loaded at boot time. The file must be in the root directory of the USB drive and be named CustomBootState.scn. This can be quite useful for ensuring a proper reset of the desk to a specific set of parameters at start time, rather than relying on a manual load of scene 0 or a console reset state from the config screen.

Scene Safes: How do they work?
There's been numerous comments and questions about "Scene Safes"; How do they work? When do they actually apply? And what to?

When a scene is saved, *all* parameters that can affect the audio signal path are saved to a text file in one fe the 100 Scene slots of the X32 (and can be exported to the USB stick). Scenes are a way to quickly and efficiently save the audio-related state of your desk, so you can recall your current settings, or move them to another desk. There is not much granularity in Scenes, they are composed of rather large sections, covering thousands of parameters in an 2105 lines text file (extension in ".scn").

So, you saved a Scene and want to (re)apply it to your desk; you can overwrite all parameters or a subset of them, according to the low granularity mentioned above. This is what SAFES are for: selecting what (groups of) elements that will *not* be modified when you recall a previously saved Scene. **SAFES are only applying on recall of Scenes**.

The Scene screen SAFES are 8 groups of parameters types: *Talkback*, *Effects*, *Mix Buses*, *Chan Process*, *Configuration*, *Preamp (HA)*, *Output Patch* and *Routing I/O*. The way you set these when saving a scene has no effect at all on the saving process, and doe not change the scene flags in the file that is created (all flags in the file will always be %00000000).

SAFE parameters are actually a finer set than the 8 main groups mentioned above, and can be set directly at the desk, under the SCENES screen; the "param safe" and "chan safe" tabs can be used for you set or select the

parameters or channels you want to preserve from any change as a Scene gets loaded to the desk.

When you recall a Scene to update your desk, it is important to ensure the SAFE flags are set as your expection. Any section that has a flag set will not affect/update the desk when the Scene is recalled/applied/loaded.

Presets (/-libs)

The X32/M32 familly 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 X32node 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 not dedicated to specific channels or effect slots.

- AES/DP48 settings

A complete list of elements found in Channel, Effects, Routing and AES/DP48 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/…* commands are used to get/set elements and values related to Shows, Cues, Scenes and Snippets.

The */-libs/…* commands are used for listing and dealing with presets.

Manipulation of datasets (/add, /copy, /save, /load, /delete, /rename)

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.

Show, Cue, Scene, Snippet, and Preset Management			
/showdump	none	Requests the X32/M32 to send all Cue, Scene and Snippet related data. Cues, Scenes and Snippets data are returned using one or more X32node messages (see below). If no cues, scene, or snippets exist, only the first line is reported by the command (see below in “Notes on the use of /showdump” paragraph).	
/-show/prepos/current	int	Scene page cue, scene or snippet slot highlighted line/index is <int> value. int = [1-099] ***Note***: selection of cue/scene/snippet depends on the	

		/-prefs/show_control 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).	
/-show/showfile/show/name	string	Name of the current show ***Note/Bug***: the displayed name changes after the "utility" screen has been selected within the Scene/home screen	
/-show/showfile/show/inputs	int	Param safe page Scene safe parameters Input channels selection: (8 bits bitmap): *bit 0: Preamp (Trim, Invert, LoCut, Delay, HA Gain, 48V)* *bit 1: Config (Color, Source)* *bit 2: EQ* *bit 3: Gate & Comp* *bit 4: Insert (Position, Src/Dest)* *bit 5: Groups (DCA assign, Mute group assign)* *bit 6: Faders, Pan (L/R level, L/R pan, M/C level)* *bit 7: Mute* e.g.: <int> = *0x00000024*: EQ and Groups are safe	
/-show/showfile/show/mxsends	int	Param safe page Scene safe parameters Input channels selection (16 bits bitmap): *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: (8 bits bitmap): *bit 0: Mix Sends* *bit 1: Config (Name, Icon, Color)* *bit 2: EQ* *bit 3: Comp* *bit 4: Insert (Position, Src/Dest)* *bit 5: Groups (DCA assign, Mute group assign)* *bit 6: Faders, Pan (L/R level, L/R pan, M/C level)* *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 (7 bits bitmap): *bit 0: Configuration (TB Mic settings, A/B level&dest, Mon/Solo settings, Channel links, Mute Groups, Osc settings, User Ctrl Assign, USB recorder gain/autoplay, Automix enable)* *bit 1: Solo (Mon/Solo settings, DP48 config)* *bit 2: Routing (Routing banks, Rec/Play)* *bit 3: Outpatch (Aux Patch, P16 Patch, AES-BU assign, USB recorder source)* *bit 4: User In* *bit 5: User Out* *bit 6: Surface State (Fader bank, Bus send bank, EQ bank, Cust. Ctrl bank, Selected channel)*	

/-show/showfile/show/chan16	int	Chan safe page Chanel safe parameters selection (16 bits bitmap): *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 (16 bits bitmap): *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 (16 bits bitmap): *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 (16 bits bitmap): *bit 0: Mix 1* … *bit 15: Mix 16*	
/-show/showfile/show/lrmtxdca	int	Chan safe page Buses safe parameters selection (16 bits bitmap): *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*	
/-show/showfile/show/effects	int	Chan safe page Effects Slots safe parameters selection (8 bits bitmap): *bit 0: FX1* … *bit 7: FX8*	
/-show/showfile/cue/[000-499]/numb	int	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)	
/-show/showfile/cue/[000-499]/name	string	Name of cue at position [000-099]	
/-show/showfile/cue/[000-499]/skip	int	*0* (no Skip) or *1* (Skip) for cue at position [000-099]	
/-show/showfile/cue/[000-499]/scene	int	Associate Scene <int> with cue at position [000-099]	
/-show/showfile/cue/[000-499]/bit	int	Associate Snippet <int> with cue at position [000-099]	

/-show/showfile/cue/[000-499]/miditype	int	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*	
/-show/showfile/cue/[000-499]/midichan	int	Set MIDI channel number to <int>	
/-show/showfile/cue/[000-499]/midipara1	int	Set Midi parameter 1 value to <int>	
/-show/showfile/cue/[000-499]/midipara2	int	Set Midi parameter 2 value to <int>	
/-show/showfile/scene/[000-099]/name	string	Scene “Name” parameter for scene [000-099]	
/-show/showfile/scene/[000-099]/notes	string	Scene “Notes” parameter for scene [000-099]	
/-show/showfile/scene/[000-099]/safes	%int	Scene “Scene Safes” parameters selection for scene [000-099] *bit 1: Talkback (TB Mic settings, A/B level&dest)* *bit 2: Effects (FX engine, FX source, FX params)* *bit 3: Mix Buses (Bus Channels data)* *bit 4: Chan Process (Chan 1-32 data, Aux 1-8 data, FxRtn channels data)* *bit 5: Configuration (Mon/Solo settings, DP48 config, Channel links, Mute groups, Osc settings, User Ctrl Assign, Usb recorder gain/autoplay, Automix enable)* *bit 6: Preamp (HA) (Ch HPF, Ch Trim, Ch Invert, HA gains, 48V)* *bit 7: Output Patch (User slots assigns, Out Patch, Aux Patch, P16 Patch, AES-BU assign, USB recorder source)* *bit 8: Routing I/O (Routing banks, Rec/Play)* e.g.: <int> = *0x00000106*: Routing I/O, Talkback and Effects are safe	
/-show/showfile/scene/[000-099]/hasdata	int	Scene at position [000-099] has valid data *0: No* *1: Yes*	
/-show/showfile/snippet/[000-099]/name	string	Snippet “Name” parameter for Snippet [000-099]	
/-show/showfile/snippet/[000-099]/eventtyp	%int	Parameter Filters & Effects affected by snippet in the form of bitwise operation: *bit 0: Preamp HA* *bit 13: FX 1* *bit 1: Config* *bit 14: FX 2* *bit 2: EQ* *bit 15: FX 3* *bit 3: Gate & Comp* *bit 16: FX 4* *bit 4: Insert* *bit 17: FX 5* *bit 5: Groups* *bit 18: FX 6* *bit 6: Fader, Pan* *bit 19: FX 7* *bit 7: Mute* *bit 20: FX 8* *bit 8: Send 1-8* *bit 9: Send 9-12* *bit 21: Config* *bit 10: Send 13-16* *bit 22: Solo* *bit 11: Send M/C – LR_SW* *bit 23: Routing*	

		bit 12: Send Matrix Bit 24: Out Patch	
/-show/showfile/snippet/[000-099]/channels	%int	Channels affected by snippet in the form of bitwise operation: *bit 0: channel 1* *…* *bit 31: channel 32*	
/-show/showfile/snippet/[000-099]/auxbuses	%int	Returns and Buses affected by snippet in the form of bitwise operation: *bit 0: Aux 1* *…* *Bit 15: FX 4R* *bit 16: Mix 1* *…* *bit 31: Mix 16*	
/-show/showfile/snippet/[000-099]/maingrps	%int	Main/Matrix/Group affected by snippet in the form of bitwise operation: *bit 0: Matrix 1* *…* *bit 15: DCA Group 8*	
/-show/showfile/snippet/[000-099]/hasdata	int	Snippet at position [000-099] has valid data *0: No* *1: Yes*	
/-libs/ch/[001-100]/pos	int	The position of the channel preset number [001-100]	
/-libs/ch/[001-100]/name	string	Name of the channel preset	
/-libs/ch/[001-100]/type	int	Type of the channel preset	
/-libs/ch/[001-100]/flags	%int	Lists the scope elements for the channel preset index [001-100] in the form of bitwise operation: *bit 0: preamp phantom ON* *bit 1: config: delay ON* *bit 2: LoCut ON* *bit 3: Gate ON* *bit 4: EQ ON* *bit 5: Dyn ON* *bit 6: 0* *bit 7: 0* *bit 8: preset has a preamp section* *bit 9: preset has a config section* *bit 10: preset has a LoCut section* *bit 11: preset has a Gate section* *bit 12: preset has a EQ section* *bit 13: preset has a Dyn section* *bit 14: 0* *bit 15: 0*	
/-libs/ch/[001-100]/hasdata	int	*{0, 1}* depending on the validity of the channel preset.	
/-libs/fx/[001-100]/pos	int	The position of the effect preset number [001-100]	
/-libs/fx/[001-100]/name	string	Name of the effect preset	
/-libs/fx/[001-100]/type	int	Type of the effect preset	
/-libs/fx/[001-100]/flags	%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. [37] ***Note**: int values do not match with FX enums!*	

[37] See Appendix for table of enum/name/type

/-libs/fx/[001-100]/hasdata	int	*{0, 1}* depending on the validity of the effect preset.	
/-libs/r/[001-100]/pos	int	The position of the routing preset number [001-100]	
/-libs/r/[001-100]/name	string	Name of the routing preset	
/-libs/r/[001-100]/type	int	Type of the routing preset	
/-libs/r/[001-100]/flags	%int	Unused (all *0*).	
/-libs/r/[001-100]/hasdata	int	*{0, 1}* depending on the validity of the routing preset.	
/-libs/mon/[001-100]/pos	int	The position of the AES/ DP48 preset number [001-100]	
/-libs/mon/[001-100]/name	string	Name of the AES/ DP48 preset	
/-libs/mon/[001-100]/type	int	Type of the AES/ DP48 preset	
/-libs/mon/[001-100]/flags	%int	Unused (all *0*).	
/-libs/mon/[001-100]/hasdata	int	*{0, 1}* depending on the validity of the AES/DP48 preset.	
/copy	string, int, int	Copies an X32/M32 internal set to another. The type of internal set is listed with the first <*string*> parameter and can be *scene*, *libchan*, *libfx*, *librout, or libmon*[38] for scene, channel, effect, routing or AES/ DP48 presets respectively. The next <*int*> is the source index, and is followed by another <*int*> representing the destination index. ***Index values start at 0.*** The server returns a status[39] indicating if the operation failed [0] or was successful [1], e.g.: *->X: /copy ,sii libchan 45 48* *X->: /copy~~~,si~libchan~[1]*	

/add	string, int, string	Adds a cue element to the current show in the X32/M32 internal memory.
		
		The first parameter is a string: *cue*
		The second parameter is an <*int*> representing the cue index and subindex. For example:
		cue index data *1.0.0* will have int=*100* for value;
		cue index data *12.5.2* will have int=*1252* for value;
		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, int, [int |	Saves or updates in the X32/M32 internal memory a scene, snippet or preset at a given index with information specific to the object saved;	

[38] */copy* does not enable copying snippets; */load* and */save* should be used instead

[39] The copy operation is not necessarily fully completed when the status is returned by the server

	string, …]	The first parameter is a string representing the type of element to save to internal memory. 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. **AES/DP48 presets**: the first parameter is followed by <*int*>, <*string*> representing respectively the AES/DP48 preset position **index** in the range [000-099], and the **name** of the preset. The server returns a status[40] indicating if the operation failed [0] or was successful [1], e.g.: *->X: /save ,siss scene 45 test note*	

[40] The save operation is not necessarily fully completed when the status is returned by the server

		X->: /save~~~,si~scene~~~[1]	
/load	string, int [,int[, %int]]	Loads from the X32/M32 internal memory a scene, snippet or a preset listed at a given index with information specific to the state/audio engine; The first parameter is a string representing the type of element to load from internal X32 memory. It can be one of: *scene*, *snippet, libchan, libfx, librout* or *libmon* for loading a scene, a snippet, a channel preset, an effect preset, a routing preset, or an AES/DP48 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 **AES/DP48:** No additional parameters The server returns a status[41] 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,	

[41] The load operation is not necessarily fully completed when the status is returned by the server

	int, string	snippet or a preset listed at a given index; The first parameter is a string representing the type of element to save. It can be one of: *scene*, *snippet*, *libchan*, *libfx*, *librout* or *libmon* for loading a scene, a snippet, a channel preset, an effect preset, a routing preset, or and AES/DP48 preset, 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. The server returns a status indicating if the operation failed [0] or was successful [1], e.g.: *->X: /rename~,sis~~~~scene~~~[99]myScene~* *X->: /rename~,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*, *librout*, or *libmon*, giving the type of element to delete. The second parameter is an **index** of the element to delete in the range [000-099]. 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]*	

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 OSC data resulting from a */node* 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.

Note: the preset name *libmon*, corresponding to AES/DP48 settings, only exists starting with FW 4.0. In 4.0 FW, it takes a manual screen refresh to show the result of /save, /load, /copy, /delete, and /rename commands above

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, 54Mbits/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 100Mbis/s wired connection.

Replies to client are formatted as per X32node commands 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 Pach 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:

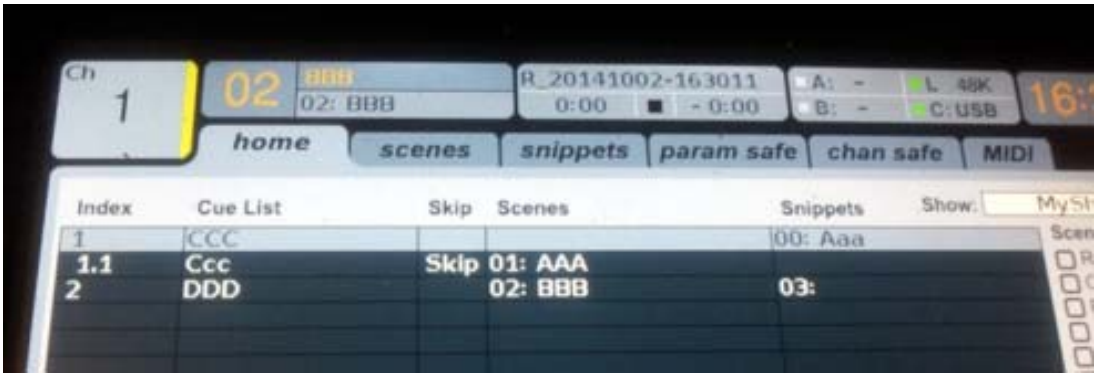
```
node~~~~,s~~/-show/showfile/show “MyShow” 0 0 0 0 0 0 0 0 0 0 “2.08”
node~~~~,s~~/-show/showfile/cue/000 100 “CCC” 0 -1 -1 0 1 0 0
node~~~~,s~~/-show/showfile/cue/001 110 “Ccc” 0 1 -1 0 1 0 0
node~~~~,s~~/-show/showfile/scene/001 “AAA” “aaa” %111111110 1
node~~~~,s~~/-show/showfile/scene/002 “BBB”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

Selecting “skip” on cue Ccc, the answer to the X32node command 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 X32node command 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
```

X32/M32 console status commands

Preferences (/-prefs) data

Preferences data			
/-prefs/style	string	Name given to your prefs ex: “Patrick”. Will be “ablesque” after factory reset	
/-prefs/bright	linf	[10., 100., 5.], Main LCD Brightness	
/-prefs/lcdcont	linf	[0., 100., 2.], Channel LCD Contrast	
/-prefs/ledbright	linf	[10., 100., 5.], LED Brightness	
/-prefs/lamp	float	[10., 100., 10.], Lamp Dim value	
/-prefs/lampon	enum	*{OFF, ON,* int [0, 1] representing the state of lamp power. Lamp is : *0: off* *1: on*	
/-prefs/clockrate	enum	*{48K, 44K1},* Int [0, 1] representing the global “Sample Rate” (in Global screen) *0: 48K* *1: 44K1*	
/-prefs/clocksource	enum	*{INT, AES50A, AES50B,…},* int [0…3] representing clock synchronization (in Global screen) *0: INT* *1: AES50A* *2: AES50B* *3: Exp. Card*	
/-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/viewrtn	enum	*{OFF, ON}* “Return to Last” in Config->View Preferences	
/-prefs/selfollowsbank	enum	*{OFF, ON}* “Sel follows Bank” in Config->View Preferences	
/-prefs/scene_advance	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[42]	enum	*{OFF, ON}* “Auto Select” in Config->View Preferences; Enables a near touch-sensitive fader selection on X32; Channel Select follows the last ‘touched’ fader.	
/-prefs/show_control	enum	int [0…2] representing “Show Control” in Config *0: CUES* *1: SCENES* *2: SNIPPETS*	
/-prefs/clockmode	enum	*{24h, 12h}* “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/invertmutes	enum	*{NORM, INV}* “Invert Leds” in Config->Mute System	
/-prefs/name	string	Name of device; The default value varies with the device, for example: “*X32-02-4A-53*” and can be changed to your	

[42] True for all channels except L/R. DCA faders will also not be monitored by the /-prefs/autosel function.

		liking. The name is also reported by the */xinfo* command.	
/-prefs/rec_control	enum	*{USB, XLIVE}* Recorder type displayed in status line: *0: USB recorder* *1: X-Live! recorder*	
/-prefs/fastFaders	int	MR32[R} only? Int [0, 1]representing the update of faders at startup	
/-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!*	
/-prefs/remote/enable	enum	*{OFF, ON}* set or report X32/M32 remote enable state	
/-prefs/remote/protocol	enum	*{MC, HUI, CC}* Protocol type for X32/M32 Remote *0: Mackie HCU [MC]* *1: Mackie HUI [HUI]* *2: Generic CC [CC]*	
/-prefs/remote/port	enum	*{MIDI, CARD, RTP}* Port used for MIDI remote *0: MIDI in/Out [MIDI]* *1: Card MIDI [CARD]* *2: RTP MIDI [RTP]*	
/-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 Prog change* *bit 4: Tx Prog change* *bit 5: Rx Fader pos CC* *bit 6: TX Fader pos CC* *bit 7: Rx Ch Mute CC* *bit 8: Tx Ch Mute CC* *bit 9: Rx Ch Pan CC* *bit 10: Tx Ch Pan CC* *bit 11: OSC over MIDI Sysex* *bit 12: XTouch over MIDI* *bit 13: XTouch over Ethernet*	
/-prefs/card/UFifc	enum	*{FW, USB, . .}*, Int [0…] representing the card interface type of the card present in the extension slot *0: FW* *1: USB* *2: … tbd*	
/-prefs/card/UFmode	enum	*{32/32, 16/16, 32/8, 8/32}*, X-UF card settings *0: 32in/32out* *1: 16in/16out* *2: 32in/8out* *3: 8in/32out*	
/-prefs/card/USBmode	enum	*{32/32, 16/16, 32/8, 8/32, 8/8, 2/2}*, X-USB card settings *0: 32in/32out* *1: 16in/16out* *2: 32in/8out* *3: 8in/32out*	

		4: 8in/8out *5: 2in/2out*	
/-prefs/card/ADATwc	enum	*{IN, OUT}*, int with value 0 or 1	
/-prefs/card/ADATsync	enum	*{WC, ADAT1, ADAT2, ADAT3, ADAT4}*, int with value [0..4]	
/-prefs/card/MADImode	enum	*{56, 64}*, int with value 0 or 1	
/-prefs/card/MADIin	enum	*{1-32, 9-40, 17-48, 25-56, 33-64}*, int with value [0..4]	
/-prefs/card/MADIout	enum	*{OFF, 1-32, 9-40, 17-48, 25-56, 33-64}*, int with value [0..5]	
/-prefs/card/MADIsrc	enum	*{OFF, OPT, COAX, BOTH}*, int with value [0..3]	
/-prefs/card/URECsdsel	enum	*{SD1, SD2}*, int with value 0 or 1, select active sdcard *0: SD1* *1: SD2*	
/-prefs/card/URECtracks	enum	*{32Ch, 16Ch, 8Ch}*, int with value [0..2] Select number of recorded tracks *0: 32 tracks* *1: 16 tracks* *2: 8 tracks*	
/-prefs/card/URECplayb	enum	*{SD, USB}*, int with value 0 or 1 Select playback device *0: SD* *1: USB*	
/-prefs/card/URECrout	enum	*{REC, PLAY, AUTO}*, int with value [0..2] X-Live! routing *0: Rec* *1: Play* *2: Auto*	
/-prefs/rta/visibility	enum	int [0…12] representing RTA EQ Overlay value *{OFF, 25%, 30%, 35%, 40%, 45%, 50%, 55%, 60%, 65%, 70%, 75%, 80%}*	%
/-prefs/rta/gain	linf	[0.0, 60.0, 6] RTA gain value (steps of 6dB)	dB
/-prefs/rta/autogain	enum	*{OFF, ON}* RTA autogain *0: disabled (OFF)* *1: set/enabled (ON)*	
/-prefs/rta/source	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/rtasource)*	
/-prefs/rta/pos	enum	*{PRE, POST}*, int with value 0 or 1 RTA chain position *0: Pre EQ* *1: Post EQ*	
/-prefs/rta/mode	enum	*{BAR, SPEC}*, int with value 0 or 1 RTA display mode selection *0: Bar[graph]*	

		1: Spec[trograph]	
/-prefs/rta/options	%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}* , int with value 0 or 1, RTA detector selection	
/-prefs/rta/decay	logf	[0.25, 16, 19] RTA adjustable decay time[43]	
/-prefs/rta/peakhold	enum	*{OFF, 1…8}* , int with value 0 or 1, RTA peak hold	
/-prefs/iQ/[01-16]/iQmodel	enum	int [0…6] representing a type of Turbosound iQ speakers *0: none* *1: iQ8* *2: iQ10* *3: iQ12* *4: iQ15* *5: iQ15B* *6: iQ18B*	
/-prefs/iQ/[01-16]/iQeqset	enum	int [0…4] representing an EQ for Turbosound iQ speakers *0: Linear* *1: Live* *2: Speech* *3: Playback* *4: User*	
/-prefs/iQ/[01-16]/iQsound	int	int representing the emulated sound profile for the attached iQ model: *iQ8 : [0…5]: iQ8, E8, F8+, UPJunior, PS8, NuQ8-DP* *iQ10: [0…4]: iQ10, F10+, UPJ-1P, PS10-R2, NuQ10-DP* *iQ12: [0…7]: iQ12, E12, JF29NT, ELX112P, PRX612M, F12+, UPA-1P, NuQ12-DP* *iQ15: [0…7]: iQ15, JF59NT, ELX115P, PRX615M, F15+, UPQ-1P, PS15-R2, NuQ15-DP* *iQ15B:[0…3]: iQ15B, E15X, S15+, B-15DP* *iQ18B:[0…4]: iQ18B, ELX18P, PRX6118S, S18+, B-18DP*	
/-prefs/key/layout	enum	int [0…3] representing a keyboard type *0: QWERTY* *1: QWERTZ* *2: AZERTY* *3: ABCDEF*	
/-prefs/key/00…99[44]	string	12-character strings: keyboard history entries.	

[43] See Appendix section for detailed values

[44] FW 4.0 and above

USB (/-usb) data

This section enables accessing and setting some of the parameters of the USB stick.
Note: all options may not be enabled or documented.

USB (/-usb)			
/-usb/path	string	Name of the current directory, e.g.: */-usb/path~~,s~~Dblues 48kHz~~~~*	
/-usb/title	string	Name of a file in the current directory of USB stick, e.g.: */-usb/title~,s~~Candy-DB~~~~*	
/-usb/dir/dirpos	int	Current directory entry number	
/-usb/dir/maxpos	int	Number of entries of the current directory in USB stick, e.g.: */-usb/dir/maxpos~~~~,i~~<int=16>*	
/-usb/dir/001…999/type	string	The type of file at position 000…999 of current directory in USB stick, e.g.: */-usb/dir/006/type~~,s~~~~~~*	
/-usb/dir/001…999/name	string	The name of file at position 000…999 of current directory in USB stick, e.g.: */-usb/dir/006/name~~,s~~Candy.wav~~* *The file "candy.wav" is at position 6 in the current directory* Can also return the name of a directory in the usb stick, e.g.: */-usb/dir/001/name~~,s~~[…]~~~~* */-usb/dir/002/name~~,s~~[DBlues 44.1Khz]~~* */-usb/dir/003/name~~,s~~[Dblues 48kHz]~~*	

Status (/-stat) data

Status data (screen, tape, fader groups, solo, etc.)			
/-stat/selidx[45]	enum	*{Ch01…Ch32, Aux1…Aux8, Fx1L…FX4R, Bus1…Mtx1…Mtx6, LR, M/C}*, int with value [0..71], Select channel index *0-31: Ch 1-32* *32-39: Aux in 1-8* *40-47: FxRtn 1-8* *48-63: Bus master* *64-69: Matrix 1-6* *70: L/R* *71: Mono/Center*	
/-stat/chfaderbank	int	Select Main channel fader bank: *0: CH 1-16* *1: CH 17-32* *2: Aux in / USB / FX returns* *3: Bus masters*	
/-stat/grpfaderbank	int	Select Group channel fader bank: *0: DCA 1-8* *1: BUS 1-8* *2: BUS 9-16* *3: Matrix 1-6, Main C* *4: TBD* *5: Channels 1-16 on X32/M32 compact or producer*[46]	
/-stat/sendsonfader	enum	*{OFF, ON}*, int with value 0 or 1, state of Sends on Faders	
/-stat/bussendbank	int	Select Bus Sends bank: *0: Rotary buttons map to Bus 1-4* *1: Rotary buttons map to Bus 5-8* *2: Rotary buttons map to Bus 9-12* *3: Rotary buttons map to Bus 13-16*	
/-stat/eqband	int	Select EQ band (in the HOME->EQ screens) *0: Low* *1: Low2* *2: Lo-Mid* *3: Hi-Mid* *4: High2* *5: High* *Low2* and *High2* are only available with 6 band equilizers, such as used in Bus, Matrix, M, and L/R strips.	
/-stat/solo	enum	*{OFF, ON}*, int with value 0 or 1 (read only), state of CLEAR SOLO button *0: No SOLO selected* *1: At least one SOLO selected*	
/-stat/keysolo	enum	*{OFF, ON}*	
/-stat/userbank	int	Display User ASIGN bank settings on X32/M32 (pressing on SET A/B/C buttons): *0: User bank A* *1: User bank B*	

[45] /-stat/selidx will be generated by X32 for all channels except L/R when the /-prefs/autosel function is ON. DCA faders will also not be monitored by the /-prefs/autosel function.

[46] In order to ensure 16 channels can spread over the left and right banks of X32/M32 Compact or Producer, the chfaderbank parameter should be set to 0 [chi 01-16].

		2: User bank C	
/-stat/autosave	enum	*{OFF, ON}*, int with value 0 or 1, X32/M32 saves automatically its state (every 2mns?)	
/-stat/lock[47]	int	X32/M32 Lock status: *0: Unlocked* *1: Locked* *2: Shutdown*	
/-stat/usbmounted	enum	*{OFF, ON}*, int with value 0 or 1 USB drive mount status: *0: Not mounted* *1: Mounted*	
/-stat/remote	enum	*{OFF, ON}*, int with value 0 or 1 Remote mode: *0: X32 in Audio Console mode* *1: X32 in DAW mode*	
/-stat/rtamodeeq	enum	*{BAR, SPEC}*, int with value 0 or 1 RTA display mode for channel EQ *0: Bar[graph]* *1: Spec[trograph]*	
/-stat/rtamodegeq	enum	*{BAR, SPEC}*, int with value 0 or 1 RTA display mode for GEQ, Dual EQ, True EQ effect *0: Bar[graph]* *1: Spec[trograph]*	
/-stat/rtaeqpre	enum	*{OFF, ON}*, int with value 0 or 1 RTA chain position for channel EQ *0: Off* *1: On*	
/-stat/rtageqpost	enum	*{OFF, ON}*, int with value 0 or 1 RTA chain position for GEQ, Dual EQ, TrueEQ effect *0: Pre* *1: Post*	
/-stat/rtasource	int	RTA source: *0…31: Channel 01…32, PRE-EQ* *32…39: Aux 01…08, PRE-EQ* *40…47: Fxrtn 1L…4R, PRE-EQ* *48…63: Bus 01…16, PRE-EQ* *64…69: Matrix 01…06, PRE-EQ* *70: L/R, PRE-EQ* *71: Mono, PRE-EQ* *72: Monitor, PRE-EQ* … *98…129: Channel 01…32, POST-EQ* *130…137: Aux 01…08, POST-EQ* *138…145: Fxrtn 1L…4R, POST-EQ* *146…161: Bus 01…16, POST-EQ* *162…167: Matrix 01…06, POST-EQ* *168: L/R, POST-EQ* *169: Mono, POST-EQ* *170: Monitor, POST-EQ* !! after Console Reset, the value of RTA source may not reflect the *METERS/RTA* screen settings *(see also /-prefs/rta/source)*	

[47] See appendixes for a description of custom boot and lock screens

/-stat/xcardtype	int	Type of card installed (seems to be informative only) Valid values: *0: None* *1: X-UF 32in/32out Firewire/USB Card* *2: X-USB 32in/32out USB Card* *3: X-DANTE 32in/32out Dante Card* *4: X-ADAT 4in/4out 32ch ADAT Card* *5: X-MADI 32ch MADI Card* *6: DN32-USB 32in/32out USB Card* *7: DN32-DANTE 32in/32out Dante Card* *8: DN32-ADAT 4in/4out 32ch ADAT Card* *9: DN32-MADI 32ch MADI Card* *10: X-Live! X-USB & sdcard recording* *11: WAVE X-WSG card*	
/-stat/xcardsync	enum	*{OFF, ON}*, int with value 0 or 1 Sync state of the expansion card	
/-stat/geqonfdr	enum	*{OFF, ON}*, int with value 0 or 1, EQ on faders: *0: Off* *1: On*	
/-stat/geqpos	int	EQ on faders window position Bitwise OR between the FX number and the 8 band window start position: <FX#> | <start pos>, e.g.: *0x100…0x800 | 0x00…0x17* */-stat/geqpos~~~,i~~<0x00000105>* *Means EQ on faders for effect slot #1, fader window starting at fader 5, covering bands 50…250Hz*	
/-stat/dcaspill	int	Int with a value [0…8] representing the spill-split value for DCA spill[48]. A value of 0 means no DCA spill ongoing	
/-stat/screen/screen	enum	*{HOME, METERS, ROUTING, SETUP, LIBRARY, EFFECTS, MONITOR, USB, SCENES, ASSIGN, LOCK}*: X32/M32 LCD active screen: *0: HOME screen* *1: METERS screen* *2: ROUTING screen* *3: SETUP screen* *4: LIBRARY screen* *5: EFFECTS screen* *6: MONITOR screen* *7: USB RECORDER screen* *8: SCENES screen* *9: ASSIGN screen* *10: LOCK screen* (get only, can only be set via */-stat/lock* command)	
/-stat/screen/mutegrp	enum	*{OFF, ON}*, int with value 0 or 1 *0: Turn off mutegrp screen* *1: Turn on mutegrp screen*	
/-stat/screen/utils	enum	*{OFF, ON}*, int with value 0 or 1 *0: Turn off utils screen* *1: Turn on utils screen*	

[48] In FW 4.06, it seems the node data returned for -stat is not consistent with other node data returned by X32; indeed, the string returned ends with a \0 byte right after the geqpos parameter, followed by the string /-stat <n>\0 corresponding to the dcaspill parameter

/-stat/screen/CHAN/page	enum	X32/M32 page in “Home” screen *0: Home [HOME]* *1: Config [CONFIG]* *2: Gate [GATE]* *3: Dyn [DYN]* *4: Eq [EQ]* *5: Sends [MIX]* *6: Main [MAIN]*	
/-stat/screen/METER/page	enum	X32/M32 page in “Meters” screen *0: Channel [CHANNEL]* *1: Mixbus [MIXBUS]* *2: Aux/fx AUX/FX]* *3: In/out [IN/OUT]* *4: Rta [RTA]* *5: Automix [AMIX]*	
/-stat/screen/ROUTE/page	enum	X32/M32 page in “Routing” screen *0: Block Input [HOME]* *1: Block AES-A [AES50A]* *2: Block AES-B [AES50A]* *3: Block Card [CARDOUT]* *4: Block XLR [XLROUT]* *5: Patch Out [ANAOUT]* *6: Patch Aux [AUXOUT]* *7: Patch P16 [P16OUT]* *8: Patch User [USER]*	
/-stat/screen/SETUP/page	enum	X32/M32 page in “Setup” screen *0: Global [GLOB]* *1: Config [CONF]* *2: Remote [REMOTE]* *3: Network [NETW]* *4: Name/Icon [NAMES]* *5: Preamps [PREAMPS]* *6: Card [CARD]*	
/-stat/screen/LIB/page	enum	X32/M32 page in “Library” screen, loading presets and options is translated into individual settings. *0: Channel [CHAN]* *1: Effects [EFFECT]* *2: Routing [ROUTE]* *3: AES50 [MONITOR]*	
/-stat/screen/FX/page	enum	X32/M32 page in “Effects” screen *0: Home [HOME]* *1: Fx1 [FX1]* *2: Fx2 [FX2]* *…* *7: Fx7 [FX7]* *8: Fx8 [FX8]*	
/-stat/screen/MON/page	enum	X32/M32 page in “Monitor” screen *0: Monitor [MONITOR]* *1: Talkback A [TALKA]* *2: Talkback B [TALKB]* *3: Ascillator [OSC]*	
/-stat/screen/USB/page	enum	X32/M32 page in “USB Recorder” screen *0: Home [HOME]* *1: Config [CONFIG]*	
/-stat/screen/SCENE/page	enum	X32/M32 page in “Scene” screen *0: Cues [HOME]*	

		1: Scenes {SCENES] *2: Snippets [BITS]* *3: Param safe [PARSAFE]* *4: Chan safe [CHNSAFE]* *5: MIDI [MIDI]*	
/-stat/screen/ASSIGN/page	enum	X32/M32 page in “Assign” screen *0: Home [HOME]* *1: Set A [SETA]* *2: Set B [SETB]* *3: Set C [SETC]*	
/-stat/aes50/state	%int	Describes AES50 state: *bit 0 = A Audio ERR* *bit 1 = B Audio ERR* *bit 2 = A Aux Err* *bit 3 = B Aux Err* *bit 4 = Lock*	
/-stat/aes50/[A, B]	string	Detected box chain and preamps: string[4] of 'A'...'M': *A: S16 B: X32C* *C: X32 D: DL251* *E: DL251HA F: S16B* *G: Z32 H: T8* *I: X32P J: X32RACK* *K: X32CORE L: M32* *M: M32R N: DL16* *O: DL16B P: SD16* *Q: SD16B R: SD8* *S: SD8B T: DL15X* *U: DL15XHA V: DL231* *W: S32 X: S32B* *Y: DL32 Z: DL32B* *a: M32C* Followed by 6 chars preamp type '0'...'4' *0: digital 1: 8chin_A* *2: 8chin_C 3: DL251* *4: Z32*	
/-stat/solosw/[id]	enum	*{OFF, ON}*, int with value 0 or 1 0/1 for on/off of solo switch *[id]: 01-32: Ch 01-32* *33-40: Auxin 1-8* *41-48: FxRtn 1-8* *49-64: Bus master 01-16* *65-70: Matrix 1-6* *71: L/R* *72: Mono/Center* *73-80: DCA 1-8*	
/-stat/talk/[A...B]	enum	Talkback *{OFF, ON}*, int with value 0 or 1	
/-stat/osc/on	enum	*{OFF, ON}*, int with value 0 or 1 0/1 for on/off of oscillator generation	
/-stat/tape/state	enum	Tape state: *0: Stop* *1: Pause*	

		2: Play *3: Pause Record* *4: Record* *5: FF* *6: REW*	
/-stat/tape/file	string	File path, ex: *"/dir000/R_20130105-205752.wav"*	
/-stat/tape/etime	int	Elapsed time in seconds during playback or recording file (every second)	
/-stat/tape/rtime	int	Remaining time in seconds of playing media or file (every second) Also reports the number of dropouts during recording as -(dropouts + 1)	
/-stat/userpar/[id]/value	int	This command is used by X32/M32 to return to the application program (client) the value of a button or encoder actioned by the user, when a MIDI note, ctrl or pgrm changes are selected for the user control element. It can also be used to set the state of an encoder or button. *Encoder A1: id = 25 value: 0…127* *Encoder A2: id = 26 value: 0…127* *Encoder A3: id = 27 value: 0…127* *Encoder A4: id = 28 value: 0…127* *Encoder B1: id = 29 value: 0…127* *Encoder B2: id = 30 value: 0…127* *Encoder B3: id = 31 value: 0…127* *Encoder B4: id = 32 value: 0…127* *Encoder C1: id = 33 value: 0…127* *Encoder C2: id = 34 value: 0…127* *Encoder C3: id = 35 value: 0…127* *Encoder C4: id = 36 value: 0…127* *Button A5: id = 01 value: 0 or 127* *Button A6: id = 02 value: 0 or 127* *Button A7: id = 03 value: 0 or 127* *Button A8: id = 04 value: 0 or 127* *Button A9: id = 05 value: 0 or 127* *Button A10: id = 06 value: 0 or 127* *Button A11: id = 07 value: 0 or 127* *Button A12: id = 08 value: 0 or 127* *Button B5: id = 09 value: 0 or 127* *Button B6: id = 10 value: 0 or 127* *Button B7: id = 11 value: 0 or 127* *Button B8: id = 12 value: 0 or 127* *Button B9: id = 13 value: 0 or 127* *Button B10: id = 14 value: 0 or 127* *Button B11: id = 15 value: 0 or 127* *Button B12: id = 16 value: 0 or 127* *Button C5: id = 17 value: 0 or 127* *Button C6: id = 18 value: 0 or 127* *Button C7: id = 19 value: 0 or 127*	

		Button C8: id = 20 value: 0 or 127 *Button C9: id = 21 value: 0 or 127* *Button C10: id = 22 value: 0 or 127* *Button C11: id = 23 value: 0 or 127* *Button C12: id = 24 value: 0 or 127*	
/-stat/urec/state	enum	*{STOP, PPAUSE, PLAY, REC}*, int with value [0..3] X-Live! (optional) extension card state *0: STOP* *1: PPAUSE* *2: PLAY* *3: REC* Note: When recording, the card will send regular updates on the elapsed and remaining times on the card, approximately every 80ms. The example below shows what happens when lauching a recording session: */-stat/urec/state ,i 3* *->X, 28 B: /-stat/urec/state~~~,i~~[3]* *X->, 28 B: /-urec/sessionpos~~~,i~~[0]* *X->, 28 B: /-stat/urec/rtime~~~,i~~[0]* *X->, 28 B: /-urec/sessionlen~~~,i~~[0]* *X->, 28 B: /-stat/urec/rtime~~~,i~~[2615103]* *X->, 28 B: /-stat/urec/etime~~~,i~~[869]* *X->, 28 B: /-stat/urec/etime~~~,i~~[949]* *X->, 28 B: /-stat/urec/etime~~~,i~~[1029]* *X->, 28 B: /-stat/urec/etime~~~,i~~[1052]* *X->, 28 B: /-stat/urec/etime~~~,i~~[1109]* *X->, 40 B: /-urec/sd1info~~,s~~4 GB - 43m, 34s ~~~~* *X->, 28 B: /-stat/urec/rtime~~~,i~~[2614143]* *X->, 28 B: /-stat/urec/etime~~~,i~~[1189]* *X->, 28 B: /-stat/urec/etime~~~,i~~[1269]* *…..* *X->, 28 B: /-stat/urec/etime~~~,i~~[4709]* *X->, 28 B: /-stat/urec/etime~~~,i~~[4789]* *X->, 28 B: /-stat/urec/state~~~,i~~[0]* *X->, 40 B: /-urec/sd1info~~,s~~4 GB - 43m, 30s ~~~~* *X->, 48 B: /-urec/session/002/name~,s~~17-09-09 12:15:14~~~* *X->, 28 B: /-urec/sessionmax~~~,i~~[2]* *X->, 28 B: /-stat/urec/rtime~~~,i~~[0]* *X->, 28 B: /-stat/urec/etime~~~,i~~[0]* *X->, 28 B: /-stat/urec/rtime~~~,i~~[4863]* *X->, 28 B: /-urec/sessionlen~~~,i~~[4863]* *X->, 28 B: /-urec/sessionpos~~~,i~~[2]* Note: When palying back a session, the card will send regular updates on the elapsed and remaining times on the card, approximately every 350ms. The example below shows what happens when lauching the playback of the recorded session above: *->X, 28 B: /-stat/urec/state~~~,i~~[2]* *X->, 36 B: /config/routing/routswitch~~,i~~[1]* *X->, 36 B: /config/routing/PLAY/1-8~~~~,i~~[0]* *X->, 36 B: /config/routing/PLAY/9-16~~~,i~~[1]* *X->, 36 B: /config/routing/PLAY/17-24~~,i~~[2]* *X->, 36 B: /config/routing/PLAY/25-32~~,i~~[3]* *X->, 36 B: /config/routing/PLAY/AUX~~~~,i~~[0]* *X->, 28 B: /-stat/urec/rtime~~~,i~~[4522]* *X->, 28 B: /-stat/urec/etime~~~,i~~[341]* *X->, 28 B: /-stat/urec/rtime~~~,i~~[4181]* *X->, 28 B: /-stat/urec/etime~~~,i~~[682]* *…..* *X->, 28 B: /-stat/urec/rtime~~~,i~~[64]* *X->, 28 B: /-stat/urec/etime~~~,i~~[4799]* *X->, 28 B: /-stat/urec/rtime~~~,i~~[4863]* *X->, 28 B: /-stat/urec/etime~~~,i~~[0]* *X->, 36 B: /config/routing/routswitch~~,i~~[0]*	

		X->, 28 B: /-stat/urec/state~~~,i~~[0] *X->, 32 B: /config/routing/IN/1-8~~,i~~[0]* *X->, 32 B: /config/routing/IN/9-16~,i~~[1]* *X->, 36 B: /config/routing/IN/17-24~~~~,i~~[2]* *X->, 36 B: /config/routing/IN/25-32~~~~,i~~[3]* *X->, 32 B: /config/routing/IN/AUX~~,i~~[0]*	
/-stat/urec/etime	int	Elapsed time in seconds during playback or recording file (in ms) *[0…86399999]*	ms
/-stat/urec/rtime	int	Remaining time in seconds during playback or recording file (in ms) *[0…86399999]*	ms

Action (/-action) data & Undo (/-undo)

Action data			
/-action/setip	int	0 by default; changing to 1 resets Network parameters. *Use with Caution!*	
/-action/setclock	string	Set clock value	
/-action/initall	int	Initialize X32 Console, 0 by default *0: No-op* *1: Init console*	
/-action/initlib	int	Initialize X32 Libraries, 0 by default *0: No-op* *1: Init libraries*	
/-action/initshow	int	Initialize X32 Show data, 0 by default *0: No-op* *1: Init show data*	
/-action/savestate	int	Save X32/M32 state *0: No-op* *1: Save state (before power off)*	
/-action/undopt	int	Creates checkpoint to get back to upon issuing an undo command. The time of the undo point will then be set in */-undo/time*, replacing any value already there	
/-action/doundo	int	*0: No-op* *1: Undo* Perfroms an undo (see */-action/undopt* and */-undo/time*) if parameter is 1	
/-action/playtrack	int	Plays track from USB recorder, 0 by default *-1: Previous track* *0: Not playing* *1: Next track*	
/-action/newscreen	int	Renew LCD screen display *0: No* *>0: Yes*	
/-action/clearsolo	int	Clear all solo buttons *0: No-op* *1: Clear solo (as if pressing the CLEAR SOLO button)*	
/-action/setprebus	int	0	
/-action/setsrate	int	Select sampling rate: *0: 48kHz* *1: 44.1KHz*	
/-action/setrtasrc	int	Selects the source used for RTA display: <int> represents the channel # *0-31: Ch 1-32* *32-63: Ch 33-64* *64-47: Aux in /USB* *48-63: Bus master* *64-69: Matrix 1-6* *70: L/R* *71: Mono/Center* *72: Monitor*	
/-action/recselect	int	Select and execute record <int> in the current directory. Records are numbered 1…n	
/-action/gocue	int	Loading a saved cue;	

		the Cue number to load is given as an int parameter ranging from 0 to 99	
/-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/selsession	int	Select X-Live! sdcard record session index *[1…100]*
		When a valid session is selected, the card provides various information about the session, and lists the markers for that session. Below is an example of selecting session #6, which contains 3 markers:
		`/-action/selsession ,i 6`
		`->X,  28 B: /-action/selsession~,i~~[  6]`
		`X->,  28 B: /-action/selsession~,i~~[  0]`
		`X->,  28 B: /-stat/urec/rtime~~~,i~~[  4970]`
		`X->,  28 B: /-urec/sessionlen~~~,i~~[  4970]`
		`X->,  24 B: /-urec/srate~~~~,i~~[  0]`
		`X->,  28 B: /-urec/sessionpos~~~,i~~[  6]`
		…
		`X->,  32 B: /-urec/marker/001/time~~,i~~[  1]`
		`X->,  32 B: /-urec/marker/002/time~~,i~~[  99]`
		`X->,  32 B: /-urec/marker/003/time~~,i~~[  199]`
		`X->,  28 B: /-urec/markermax~~~~,i~~[  3]`	

/-action/delsession	int	Delete X-Live! sdcard record session index *[1…100]*. This command is replied with a *[0]*, indicating the command has been processed	
/-action/selmarker	int	Select X-Live! marker index *[1…100]*. This command is replied with a *[0]*, indicating the command has been processed	
/-action/delmarker	int	Delete X-Live! marker index *[1…100]*. This command is replied with a *[0]*, indicating the command has been processed	
/-action/savemarker	int	Save X-Live! sdcard position at marker index *[1…100]*. This command is replied with a *[0]*, indicating the command has been processed	
/-action/addmarker	int	Add X-Live! Marker *[0,1]*	
/-action/setposition	int	Set X-Live! sdcard position, in milliseconds. The position is a 32bit value *[0…86399999]*, corresponding to 24h of recording.	ms

/-action/clearalert	int	Clear X-Live! alert status *[0,1]*
		`0: No-op`
		`1: Clear alert`	

/-action/formatcard	int	Format active sdcard[49]
		Upon formatting, the following messages will be returned with *sd1* or *sd2* depending on the selected card:
		`X->, 28 B: /-action/formatcard~,i~~[  0]`
		`X->, 24 B: /-urec/sd1state~,i~~[  3]`
		`X->, 44 B: /-urec/sd1info~~,s~~Formatting SD Card..~~~~`
		`X->, 24 B: /-urec/sd1state~,i~~[  1]`
		`X->, 40 B: /-urec/sd1info~~,s~~4 GB - 44m,  1s ~~~~`	

[49] See the */prefs/card/URECsdsel* command

		The last two messages will be returned after the formatting of the SDcard is completed. The content of the */-urec/sd1info* message obviously depends on the SDcard.	
/-undo/time **Note**: There's only 1 undo step in X32	string	Displays the most recent time value recorded for changes, for example in selecting a scene. Time is coded as string, e.g. *18:36:54* Setting of the value is dependent on console changes or set by the */-undo/time* command *If string is empty: there are no more undo steps available*	

X-Live! sdcard recording (/-urec)

These commands are read-only and report the status of the X-Live! (optional) extension card or its components' state.

X-Live data			
/-urec/sessionmax	int	X-Live! record session max index *[0…100]* A value of 0 means there are no sessions in the sdcard (such as after formatting for example)	
/-urec/markermax	int	X-Live! marker max index *[0…100]* A value of 0 means there are no markers in the session	
/-urec/sessionlen	int	X-Live! record session length in milliseconds; a 32bit value *[0…86399999]*, corresponding to 24h max of recording.	ms
/-urec/sessionpos	int	X-Live! record session current index *[0…100]* A value of 0 means there is no session selected	
/-urec/markerpos	int	X-Live! marker current index *[0…100]* A value of 0 means there is no marker selected	
/-urec/batterystate	enum	*{NONE, GOOD, LOW}*, int with value [0..2] X-Live! internal battery state *0: NONE* *1: GOOD* *2: LOW*	
/-urec/srate	int	X-Live! record session sampling rate: *0: 44.1kHz* *1: 48kHz*	
/-urec/tracks	int	X-Live! recording type: *0: None* *8: 8 tracks* *16: 16 tracks* *32: 32 tracks*	
/-urec/sessionspan	int	Indicates if recording spans from one sdcard to the next *[0…3]* *0: No spanning* *1: Span 1 of 2* *2: Span 2 of 2* *3: Jump to 2/2*	
/-urec/sessionoffs	int	Denotes the end point of the first part of an X-Live! spanned session in milliseconds; a 32bit value *[0…86399999]*	
/-urec/sd1state	enum	*{NONE, READY, PROTECT, ERROR}*, int with value [0..3] X-Live! sdcard 1 state *0: NONE* *1: READY (ready for use)* *2: PROTECT (protected card)* *3: ERROR (error state)*	
/-urec/sd2state	enum	*{NONE, READY, PROTECT, ERROR}*, int with value [0..3] X-Live! sdcard 2 state *0: NONE* *1: READY (ready for use)* *2: PROTECT (protected card)* *3: ERROR (error state)*	
/-urec/sd1info	string	32 chars string giving information for sdcard 1; for	

		example: */-urec/sd1info* *->X, 16 B: /-urec/sd1info~~* *X->, 40 B: /-urec/sd1info~~,s~~4 GB - 43m, 54s ~~~~*	
/-urec/sd2info	string	32 chars string giving information for sdcard 2; for example: */-urec/sd2info* *->X, 16 B: /-urec/sd2info~~* *X->, 36 B: /-urec/sd2info~~,s~~Insert SD Card.~*	
/-urec/errormessage	string	X-Live! error description string (see below)	
/-urec/errorcode	int	X-Live! error code *[0…23]* *Warning: the list below is not verified*	

Error number & code	UI message
0: OK	
1: SDCARD_ERROR	Card not ready!
2: NO_COM_PROC	
3: SDCARD_BUSY	
4: SESSION_NO_OPEN	No Session open. Select Session
5: MARKER_ERROR	
6: SYS_ERROR	
7: LOG_ERROR	Please re-insert the SD card
8: UNKNOWN_COM	
9: COM_ON_PROC	
10: SD_FULL_ERROR	Max capacity of SD card reached
11: SD_MAX_SIZE	Maximum session length 24 hours
12: SD_SLOW_ERROR	Drop outs detected. SD card slow
13: WRONG_WAV_ERROR	WAV unsupported or inconsistent
14: PROTOCOL_ERROR	
15: HARDWARE_ERROR	SD card may be damaged
16: FAT_ERROR 2	
17: SD_NOT_READY	
18: FILE_NO_FOUND	File in session folder missing
19: DIR_NO_FOUND	Second part of session not found
20: FAT_ERROR 6	
21: FAT_SYS_FULL	SD card full
22: FAT_ERROR 8	
23: INVALID_OBJECT	

/-urec/session/[001…100]/name	string	X-Live! record session name; 19 char max string[50] representing the session name	
/-urec/marker/[001…100]/time	int	X-Live! marker time in milliseconds; a 32bit value *[0…86399999]*, corresponding to 24h of recording.	ms

[50] This seems incorrect; only 16 or 17 characters can be dispayed and it seems this is the real/actual limit.

X-Live! recording data

Range Maximum values				
32-bit address range	hours	minutes	seconds	milliseconds
4.294.967.296	24	1440	86,400	86,399,999

Locator Resolution				
Channels	8-ch	16-ch	32-ch	Sample rate
32kB cluster	4.00k	2.00k	1.00k	
8kS @44k1	23.2ms	11.6ms	5.8ms	@44.1kHz
8kS @48k	21.3ms	10.7ms	5.3ms	@48kHz

Recording data format

X-Live! SD-card interface is optimized for write speed ensuring long 32 channel recordings of 48 kHz / 32-bit PCM data, with minimal risk for audio drop-outs on a large variety of SD or SDHC cards. Class-10 cards (guaranteed 10MB/s write speed) are recommended for use with X-Live!

In order to achieve optimum write performance, all tracks (8, 16 or 32) are written into the same file. The file format is 32-bit PCM multi-channel WAV. Supported card file system is FAT32 (royalty free). The individual file size with 32-bit formatting is 4GB.

Recording 32 tracks of 48 kHz uncompressed 32-bit audio requires about 340 MB of memory per minute. Hence, a 4GB file may not be longer than 11.7 minutes at maximum audio bitrate. In order to allow for longer consistent recording time, X-Live! creates a so-called Session, i.e. a folder containing one or more files (or *takes*) of up to 4GB each.

Separating recorded sessions into individual wave files, or creating multi-channel sessions from individual files for playback on X-Live! require the use of external utilities[51] or can be managed directly from DAW software (for separating sessions info individual files).

Recording Session

The X-Lve! card will auto-create a subfolder underneath "X-LIVE", named by the 32-bit timestamp of the recording start as an 8-character hex-string, e.g. "4ACE72B1". The Console will read the folder name and display the corresponding timestamp as the session name, unless it was given another name (see below).
The Session name coding is done on 32 bits and displays as a string of 8 hexadecimal characters. The format is as such:

Y	Y	Y	Y	Y	Y	Y	M	M	M	M	D	D	D	D	D	H	H	H	H	H	M	M	M	M	M	M	S	S	S	S	S
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

Years are counted starting at 1980
Seconds are divided by 2

[51] X32Xlive_Wav and X32Wav_Xlive (https://sites.google.com/site/patrickmaillot/x32) provide both ways conversions. XLive SD Splitter (https://sites.google.com/view/x32-stuff-here/home), or Wave Agent (https://www.sounddevices.com/) provide splitting capability.

Session folders include a binary session log file "SE_LOG.bin", and 1 or more takes as multichannel wave files, up to 4GB each. The name of these takes is 8-characters long and in the range of "00000001.wav" to "00000128.wav" exactly.
The maximum recording time is 24h, no matter if 8, 16 or 32 channels are recorded. When spanning a recording from one SD-card to the other, the session's folder name on the other card will be the same as on the first card, and the takes inside will start being numbered from 00000001.wav upwards again.

Preproduced Session (for playback)
The SD card will require a subfolder "X-LIVE" in the root directory, which is to contain the Session folder that may be named with up to 19 ASCII characters[52]. The Session folder needs to contain multichannel WAV files, up to 4GB each, all 32-bit PCM takes which names must be 8-characters long and in the range of "00000001.wav" to "00000128.wav" exactly.
For a consistent WAV header structure, all of these files should be created using the same DAW/editor tool, at the same 44.1 or 48 kHz sample rate and with the same channel count, i.e. 8, 16 or 32 channels

If not provided, the Console will create a session log file automatically, read the optional user-given 19-character folder name and copy it into the log file, before renaming the folder with the creation time stamp. When possible, the Console displays the session name, rather than the timestamp. Markers will be read if available and the binary session log file "SE_LOG.bin" will be updated with any Markers applied during playback.

Drop Outs
Drop outs are caused by record/playback buffer under-runs due to insufficient SD card data transfer performance. X-Live! provides a 10s audio buffer for recording, however, there are old/cheap/slow cards that cannot meet the required average data rate, or present available free memory that is so fragmented that allocating/finding suitable space for write operations takes too much time. If possible , formatting the card before use is always a good idea.

Fragmentation
When a freshly formatted card is used for recording, then all data will be written in consistent address ranges successively, and there is no fragmentation. File fragmentation happens when previous cut/delete operations were performed. The deleted file areas will be recognized as "free memory" in the file system, and when a recording (write pointer) reaches the end of the SD memory, it is redirected to the next free space, i.e. a formerly deleted area. Finding the next free memory area might take too long, especially when using extremely large SD memory (i.e. several hundred of GB).

[52] This seems incorrect; only 16 or 17 characters can be dispayed and it seems this is the real/actual limit.

Subscribing to X32/M32 Updates

There may be situations when you (or the application you write) may not want to receive all data sent by the X32/M32 resulting of mainaining a */xremote* command active, as this may represent a lot of data to parse.

Besides the */xremote* command which enables clients to receive pretty much all X32/M32 changes or updates resulting from an <OSC Address Pattern> Set parameter command sent from a different client, there are a series of commands a client can use to manage specific updates from X32/M32: */subscribe*, */formatsubscribe*, */batchsubscribe*, */renew*, and */unsubscribe*

If not renewed within 10 seconds, the */subscribe*, */formatsubscribe*, */batchsubscribe* commands names and attributes are forgotten and lost. Indeed, an attempt to renew one of the above commands received past the 10s delay will have no effect.

The */subscribe* command enables getting regular updates for a single <OSC Address Pattern> command. A typical use would be: */subscribe ,si <command> [tf]*, where

<*command*> is an X32/M32 <OSC Address Pattern> parameter command, for example: */ch/01/mix/on*

[tf] is an integer which sets the number of updates received over a 10 seconds period:
[tf]: 0 → 200 updates
2 → 100 updates
[…]
40 → 5 updates
80 to 99 → 3 updates, values outside of the range [0…99] are considered 0

The */formatsubscribe ,ss[s…]iii <name> <command> [<command>…] [i0][i1][tf]* goes one step further and enables receiving regular updates for a series of commands, optionnaly using wildcard '*' characters to represent variable ranges.

<*name*> is an alias (string) given to the command, and can be later used for requesting specific renews for additional rounds of updates. As the alias will be used as an OSC address pattern, it is a good idea to start the alias name with a leading *'/'*.

<*command*> is an <OSC Address Pattern> command. There can be several commands in a single */formatsubscribe.* Some X32/M32 commands refer to range attributes, such as in a channel number: [01-32]. Range data character digits can be replaced by '*' characters. For example */dca/[1-8]/on* is replaced by */dca/*/on*, */ch/[01-32]/mix/on* will be replaced by */ch/**/mix/on*, and so on.

[i0] and *[i1]* are integers to represent the start and end range numbers, respectively.

[tf] as previously, is an integer affecting the number of updates received over a 10 seconds' period.

```
X32_Command - v1.29 - (c)2014-15 Patrick-Gilles Maillot

Connecting to X32. Done!
/formatsubscribe ,ssiii /testme /ch/**/mix/on 6 9 80
X->,   36 B: 2f746573746d65002c6200000000001414000000010000000100000001000000001000000
             / t e s t m e ~ , b ~ ~20 chrs:
X->,   36 B: 2f746573746d65002c6200000000001414000000010000000100000001000000001000000
             / t e s t m e ~ , b ~ ~20 chrs:
X->,   36 B: 2f746573746d65002c6200000000001414000000010000000100000001000000001000000
             / t e s t m e ~ , b ~ ~20 chrs:
```

(Each response from X32 above is spaced by about 3 seconds)

The previous example asks for receiving during the next 10s about 3 updates of the states of mutes for channel 06 to channel 09 inclusive. The command is aliased "/testme" and can be renewed using a */renew ,s /testme* command sent within the 10s following the call to */formatsubscribe*. The 4 values of channel mutes are returned as an OSC blob, as shown above with the hex dump, the responses are made of the name[53] of the command: */testme*, a blob tag *,b* followed by a 32bit big endian integer with value 20 representing the number of chars in the OSC blob payload. The blob itself consists of 32bit little endian integers; the first int is the blob length in bytes (20 again), from which the total number of ints of the blob can be computed (20/4 = 5), meaning there are 4 ints following the first one. If the channel [06-09] mute states were to change during the effective time of the command, the values of the respective ints (all '1' here) would have been changed to '0'.

The next example below starts with Bus 01 & 02 linked, and Channels 09 to 13 being muted. With */xremote* being maintained active, a */formatsubscribe* command is issued, requesting 10 updates for buslink/1-2, and channel [10-12] mutes updates. As the command executes, Bus/1-2 is unlinked, then channels 09 to 13 are successively unmuted. The command does repeatedly report mute status only for channels 10 to 12, as requested.

For easier reading, the X32 data resulting of user action reported thanks to the */xremote* command being active is displayed in red, while the data resulting from the */formatsubscribe* command is displayed in blue.

```
X32_Command - v1.29 - (c)2014-15 Patrick-Gilles Maillot

Connecting to X32. Done!
xremote on
/formatsubscribe ,sssiii /www /config/buslink/1-2 /ch/**/mix/on 10 12 20
X->,    36 B: /www~~~~,b~~20 chrs:
X->,    36 B: /www~~~~,b~~20 chrs:
X->,    28 B: /config/buslink/1-2~,i~~[     0]
X->,    36 B: /www~~~~,b~~20 chrs:
X->,    24 B: /ch/09/mix/on~~~,i~~[     1]
X->,    36 B: /www~~~~,b~~20 chrs:
X->,    24 B: /ch/10/mix/on~~~,i~~[     1]
X->,    36 B: /www~~~~,b~~20 chrs:
X->,    24 B: /ch/11/mix/on~~~,i~~[     1]
X->,    36 B: /www~~~~,b~~20 chrs:
X->,    24 B: /ch/12/mix/on~~~,i~~[     1]
X->,    36 B: /www~~~~,b~~20 chrs:
X->,    24 B: /ch/13/mix/on~~~,i~~[     1]
X->,    36 B: /www~~~~,b~~20 chrs:
X->,    36 B: /www~~~~,b~~20 chrs:
X->,    36 B: /www~~~~,b~~20 chrs:
```

(Each response in blue from X32 above is spaced by about 1 second)

[53] It is not mandatory but very wise to use a leading *'/'* character for the *<name>* parameter; indeed, the data sent back by the X32 will use *<name>* as OSC address pattern for the command and most OSC libraries will need this to be OSC protocol compliant, so it must begin with a *'/'*.

/batchsubscribe is a command to display meter data only [TBV]. The format is close to */formatsubscribe*: The command is aliased to a name and accepts a single meter command followed by two ints for the meter command parameters (ints are 0 if the meter command does not take arguments); as for the other commands, the last int represents a time factor.

```
X32_Command - v1.29 - (c)2014-15 Patrick-Gilles Maillot

Connecting to X32. Done!
/batchsubscribe ,ssiii /yy /meters/6 0 0 40
X->,    32 B: /yy~,b~~4 flts: 000.00 001.00 001.00 000.00
X->,    32 B: /yy~,b~~4 flts: 000.00 001.00 001.00 000.00
X->,    32 B: /yy~,b~~4 flts: 000.00 001.00 001.00 000.00
X->,    32 B: /yy~,b~~4 flts: 000.00 001.00 001.00 000.00
X->,    32 B: /yy~,b~~4 flts: 000.00 001.00 001.00 000.00
```

```
X32_Command - v1.29 - (c)2014-15 Patrick-Gilles Maillot

Connecting to X32. Done!
/batchsubscribe ,ssiii /rr /meters/5 3 1 40
->X,    52 B: /batchsubscribe~,ssiii~~/rr~/meters/5~~~[     3][     1][    40]
X->,   124 B: /rr~,b~~27 flts: 000.00 000.00 000.00 000.00 000.00 000.00 000.00 000.00
000.00 000.00 000.00 000.00 000.00 000.00 000.00 000.00 000.00 000.00 000.00 000.00
000.00 000.00 000.00 000.00 000.00 000.00

[…]

X->,   124 B: /rr~,b~~27 flts: 000.00 000.00 000.00 000.00 000.00 000.00 000.00 000.00
000.00 000.00 000.00 000.00 000.00 000.00 000.00 000.00 000.00 000.00 000.00 000.00
000.00 000.00 000.00 000.00 000.00 000.00
```

Refer to the */meter/5* meters command for the meaning of the two arguments *[3]* and *[1]*.

As already mentioned, the above subscription commands are valid for 10s. In order to keep receiving data from the X32/M32, subscriptions have to be renewed with the */renew* command. The command takes one optional argument, a string type to specify the subscription to renew. This will be either ne name of the actual command or the name of the command alias for renewing */formatsubscribe* and */batchsubscribe* commands. It is possible to renew all active subscriptions by not providing any name to the */renew* command.

```
X32_Command - v1.29 - (c)2014-15 Patrick-Gilles Maillot

Connecting to X32. Done!
/subscribe ,si /ch/01/mix/on 2
/formatsubscribe ,sssiii /AA /config/buslink/1-2 /ch/01/mix/fader 0 0 5
X->,    24 B: /ch/01/mix/on~~~,i~~[     1]
X->,    24 B: /ch/01/mix/on~~~,i~~[     1]
/batchsubscribe ,ssiii /BB /meters/6 0 0 10
X->,    24 B: /AA~,b~~12 chrs:              Ç ?
X->,    24 B: /ch/01/mix/on~~~,i~~[     1]
X->,    32 B: /BB~,b~~4 flts: 000.00 001.00 001.00 000.00
X->,    24 B: /ch/01/mix/on~~~,i~~[     1]
X->,    24 B: /ch/01/mix/on~~~,i~~[     1]
X->,    24 B: /AA~,b~~12 chrs:              Ç ?
[…]
X->,    24 B: /ch/01/mix/on~~~,i~~[     1]
X->,    24 B: /ch/01/mix/on~~~,i~~[     1]
X->,    24 B: /AA~,b~~12 chrs:              Ç ?
X->,    24 B: /ch/01/mix/on~~~,i~~[     1]
X->,    32 B: /BB~,b~~4 flts: 000.00 001.00 001.00 000.00
X->,    24 B: /ch/01/mix/on~~~,i~~[     1]
X->,    24 B: /ch/01/mix/on~~~,i~~[     1]
```

Managing multiple subscriptions

As shown above, the X32/M32 can manage multiple subscriptions. Data from different subscriptions will be mixed. Shown below, 3 different subscription requests are made for a period of 10s. Commands are in black and the X32 replies are in 3 different colors for easier reading.

At anytime, subscriptions can be stopped using the */unsubscribe* command. As several subscriptions can be active at one time, the command takes a string argument to select which subscription should be stopped. An */unsubscribe* command with no argument will stop all active subscriptions.

Subscribing to data updates

Subscribe to data			
/subscribe	string int	String: an X32/M32 command, int is a time_factor	
/formatsubscribe	string string [string…] int int int	The first string is the alias name for the command; the scond string is one or more commands with possible wildcards. The two first ints represent the wildcards range, and the last int is the time_factor	
/batchsubscribe	string string int int int	The first string is the alias name for the meter command; the scond string the meter commands. The two first ints can be used for the meter command arguments if needed, and the last int is the time_factor	
/renew	[string]	Element to be renewed (can be an alias of a command). Absence of parameter means renew “all” active subscriptions	
/unsubscribe	[string]	Subscription to be stopped (can be an alias of a command). Absence of parameter means: stop “all” active subscriptions	

Note: Strings, when returned in subscribe commands blobs are 32byte fixed length, padded with \0 if necessary.

X32node (/node, /) commands

X32/M32 nodes are collections of parameters grouped in logical sets. They enable sending or receiving complex commands by grouping several parameters, optimizing the communication between X32/M32 and its clients (less I/O operations and less data to transmit). For some of them they also serve as the base to scene and snippet files through the use of */node* or */* commands, explained below.

The */* command is used to send X32node formatted commands (i.e. similar formatting as of a */node* command) to the X32/M32, thus updating several or all parameters of a node at once and using clear text data. The command follows the standard OSC specification and takes a single string as argument. The data to send should conform to X32/M32 formats and known values, but the X32 will keep the closest value to the one sent if that is not the case; for example, sending */ ,s "ch/01/mix/fader -85.4"* will be kept as *-85.3*, as *-85.4* is not one of the 1024 "known values" for faders[54].
The leading *'/'* of the command to be sent to X32 is not mandatory, i.e. sending */ ,s "/ch/01/mix/fader -85.4"* is the same as sending */ ,s "ch/01/mix/fader -85.4"*.

Data sent with */* commands is in the form of a variable list of arguments; i.e. the list of data does not have to be complete for the command to be valid and accepted. Of course, parameters have to be provided in the order they are expected to by X32, and only those which are sent will provide updated values, for example:

```
/~~~,s~~ch/01 name 30
```

Will only set the two first items of the */ch/01/config* node which is composed of name, icon, color and source. A complete for of the command would for example be:

```
/~~~,s~~ch/01 newname 10 CY 1
```

The X32/M32 will echo back the *"/"* commands it receives, enabling a better control of the flow of data and helping avoid overruns by ensuring an application does read the UDP buffer before sending the next command.

The */node* command can be used by clients to request values and data for the X32node 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.

X32node commands		
/	string	Send X32node data passed as a string argument to X32/M32. Example: */~~~,s~~-prefs/iQ/01 none Linear 0~~* *or* */~~~,s~~/-prefs/iQ/01 none Linear 0~* Will set the Turbosound iQ speaker parameters at address 01 with the settings listed with the command. The */* command works for all X32nodes. X32node commands sent this way will be echoed back by the X32/M32.
/node	string	Request the X32/M32 to return the data associated with the X32node given in argument.

[54] See Appendix on fader float values

		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 X32node parameters. All the items below must 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/userrout/out* *config/userrout/in* *config/routing/routswitch* *config/routing/IN* *config/routing/AES50A* *config/routing/AES50B* *config/routing/CARD* *config/routing/OUT* *config/routing/PLAY* *config/userctrl/{A,B,C}* *config/userctrl/{A,B,C}/enc* *config/userctrl/{A,B,C}/btn* *config/tape* *config/amixenable* *config/dp48* *config/dp48/assign* *config/dp48/link* *config/dp48/grpname* *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* *ch/[01…32]/automix* *ch/[01…32]/automix/group* *ch/[01…32]/automix/weight*

		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]* *-insert* *-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/lrmtxdca* *-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 -libs/mon -libs/mon/[001-100] -libs/mon/[001-100]/pos -libs/mon/[001-100]/name -libs/mon/[001-100]/flags -libs/mon/[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/selfollowsbank -prefs/scene_advance -prefs/safe_masterlevels -prefs/haflags -prefs/autosel -prefs/show_control -prefs/clockmode -prefs/hardmute -prefs/dcamute -prefs/invertmutes -prefs/name -prefs/rec_control -prefs/fastFaders -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/card* *-prefs/card/UFifc* *-prefs/card/UFmode* *-prefs/card/USBmode* *-prefs/card/ADATwc* *-prefs/card/ADATsync* *-prefs/card/MADImode* *-prefs/card/MADIin* *-prefs/card/MADIout* *-prefs/card/MADIsrc* *-prefs/card/URECtracks* *-prefs/card/URECplayb* *-prefs/card/URECrout* *-prefs/card/URECsdsel* *-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/iQ* *-prefs/iQ/[01-16]* *-prefs/iQ/[01-16]/iQmodel* *-prefs/iQ/[01-16]/iQeqset* *-prefs/iQ/[01-16]/iQsound* *-prefs/key* *-prefs/key/layout* *-prefs/key/[00..99]* *-usb* *-usb/path* *-usb/title* *-usb/dir* *-usb/dirpos* *-usb/maxpos* *-usb/dir/[000…999]* *-usb/dir/[000…999]/name* *-stat* *-stat/selidx* *-stat/chfaderbank* *-stat/grpfaderbank* *-stat/sendsonfader*

		-stat/bussendbank *-stat/eqband* *-stat/keysolo* *-stat/userbank* *-stat/autosave* *-stat/lock* *-stat/usbmounted* *-stat/remote* *-stat/rtamodeeq* *-stat/rtamodegeq* *-stat/rtaeqpre* *-stat/rtageqpost* *-stat/rtasource* *-stat/xcardtype* *-stat/xcardsync* *-stat/geqonfdr* *-stat/geqpos* *-stat/screen* *-stat/screen/screen* *-stat/screen/mutegrp* *-stat/screen/utils* *-stat/screen/CHAN* *-stat/screen/METER* *-stat/screen/ROUTE* *-stat/screen/SETUP* *-stat/screen/LIB* *-stat/screen/FX* *-stat/screen/MON* *-stat/screen/USB* *-stat/screen/SCENE* *-stat/screen/ASSIGN* *-stat/aes50* *-stat/aes50/state* *-stat/aes50/stats/[A…B]* *-stat/solosw* *-stat/solosw/[01…80]* *-stat/talk* *-stat/talk/[A…B]* *-stat/osc* *-stat/osc/on* *-stat/tape* *-stat/tape/state* *-stat/tape/file* *-stat/tape/etime* *-stat/tape/rtime* *-stat/urec/state* *-stat/urec/etime* *-stat/urec/rtime* *-action* *-action/setip*

		-action/setclock *-action/initall* *-action/initlib* *-action/initshow* *-action/savestate* *-action/undopt* *-action/doundo* *-action/playtrack* *-action/newscreen* *-action/clearsolo* *-action/setprebus* *-action/setsrate* *-action/setrtasrc* *-action/newscreen* *-action/recselect* *-action/gocue* *-action/goscene* *-action/undopt* *-action/gosnippet* *-action/selsession* *-action/delsession* *-action/selmarker* *-action/delmarker* *-action/savemarker* *-action/addmarker* *-action/selposition* *-action/clearalert* *-action/formatcard* *-urec/sessionmax* *-urec/markermax* *-urec/sessionlen* *-urec/sessionpos* *-urec/markerpos* *-urec/batterystate* *-urec/srate* *-urec/tracks* *-urec/sessionspan* *-urec/sessionoffs* *-urec/sd1state* *-urec/sd2state* *-urec/sd1info* *-urec/sd2info* *-urec/errormessage* *-urec/errorcode* *-urec/session/[001..100]/name* *-urec/marker/[001..100]/time*

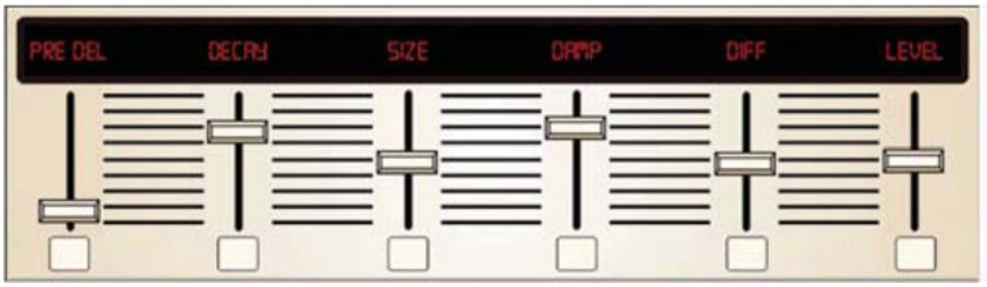
Note/bug: the response from the Server is "*node…"* and not *"/node…"* as one could expect. This is not OSC compliant.

EFFECTS
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

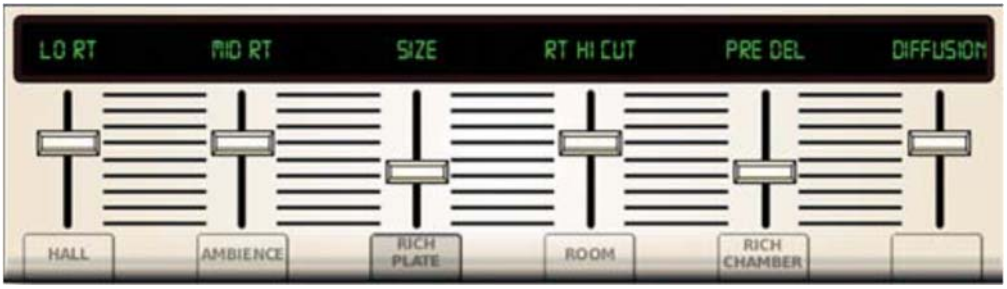


Effect Name	Parameters	Parameter Name	Type & Range	Par #
HALL (hall reverb)	fffffffffff	Pre Delay	linf [0…200]	1
		Decay	logf [0.2…5]	2
		Size	linf [2…100]	3
		Damping	logf [1k…20k]	4
		Diffuse	linf [1…30]	5
		Level	linf [-12…+12]	6
		Lo Cut	logf [10…500]	7
		Hi Cut	logf [200…20k]	8
		Bass Multi	logf [0.5…2]	9
		Spread	linf [0…50]	10
		Shape	linf [0…250]	11
		Mod Speed	linf [0…100]	12

Plate Reverb

Effect Name	Parameters	Parameter Name	Type & Range	Par #
PLAT (plate reverb)	fffffffffff	Pre Delay	linf [0…200]	1
		Decay	logf [0.5…10]	2
		Size	linf [2…100]	3
		Damping	logf [1k…20k]	4
		Diffuse	linf [1…30]	5
		Level	linf [-12…+12]	6
		Lo Cut	logf [10…500]	7
		Hi Cut	logf [200…20k]	8
		Bass Multi	logf [0.5…2]	9
		Xover	logf [10…500]	10
		Mod Depth	linf [1…50]	11
		Mod Speed	linf [0…100]	12

Ambiance Reverb



Effect Name	Parameters	Parameter Name	Type & Range	Par #
AMBI (ambiance reverb)	ffffffffff	Pre Delay	linf [0…200]	1
		Decay	logf [0.2…7.3]	2
		Size	linf [2…100]	3
		Damping	logf [1k…20k]	4
		Diffuse	linf [1…30]	5
		Level	linf [-12…+12]	6
		Lo Cut	logf [10…500]	7
		Hi Cut	logf [200…20k]	8
		Modulate	linf [0…100]	9
		Tail Gain	linf [0…100]	10

Rich Plate Reverb

Effect Name	Parameters	Parameter Name	Type & Range	Par #
RPLT (rich plate reverb)	ffffffffffffffff	Pre Delay	linf [0…200]	1
		Decay	logf [0.3…29]	2
		Size	linf [4…39]	3
		Damping	logf [1k…20k]	4
		Diffuse	linf [0…100]	5
		Level	linf [-12…+12]	6
		Lo Cut	logf [10…500]	7
		Hi Cut	logf [200…20k]	8
		Bass Multi	logf [0.25…4]	9
		Spread	linf [0…50]	10
		Attack	linf [0…100]	11
		Spin	linf [0…100]	12
		Echo L	linf [0…1200]	13
		Echo R	linf [0…1200]	14
		Echo Feed L	linf [-100…+100]	15
		Echo Feed L	linf [-100…+100]	16

Room Reverb

Effect Name	Parameters	Parameter Name	Type & Range	Par #
ROOM (room reverb)	ffffffffffffffff	Pre Delay	linf [0…200]	1
		Decay	logf [0.3…29]	2
		Size	linf [4…76]	3
		Damping	logf [1k…20k]	4
		Diffuse	linf [0…100]	5
		Level	linf [-12…+12]	6
		Lo Cut	logf [10…500]	7
		Hi Cut	logf [200…20k]	8
		Bass Multi	logf [0.25…4]	9
		Spread	linf [0…50]	10
		Shape	linf [0…250]	11
		Spin	linf [0…100]	12
		Echo L	linf [0…1200]	13
		Echo R	linf [0…1200]	14
		Echo Feed L	linf [-100…+100]	15
		Echo Feed L	linf [-100…+100]	16

Chamber Reverb

Effect Name	Parameters	Parameter Name	Type & Range	Par #
CHAM (chamber reverb)	ffffffffffffff	Pre Delay	linf [0…200]	1
		Decay	logf [0.3…29]	2
		Size	linf [4…72]	3
		Damping	logf [1k…20k]	4
		Diffuse	linf [0…100]	5
		Level	linf [-12…+12]	6
		Lo Cut	logf [10…500]	7
		Hi Cut	logf [200…20k]	8
		Bass Multi	logf [0.25…4]	9
		Spread	linf [0…50]	10
		Shape	linf [0…250]	11
		Spin	linf [0…100]	12
		Reflection L	linf [0…500]	13
		Reflection R	linf [0…500]	14
		Reflection Gain L	linf [0…100]	15
		Reflection Gain L	linf [0…100]	16

4-Tap Delay

Effect Name	Parameters	Parameter Name	Type & Range	Par #
4TAP (4-tap delay)	fffffifififiii	Time	linf [1…3000]	1
		Gain Base	linf [0…100]	2
		Feedback	linf [0…100]	3
		Lo Cut	logf [10…500]	4
		Hi Cut	logf [200…20k]	5
		Spread	linf [0…6]	6
		Factor A	enum [1/4, 3/8, 1/2, 2/3, 1, 4/3, 3/2, 2, 3]	7
		Gain A	linf [0…100]	8
		Factor B	enum [1/4, 3/8, 1/2, 2/3, 1, 4/3, 3/2, 2, 3]	9
		Gain B	linf [0…100]	10
		Factor C	enum [1/4, 3/8, 1/2, 2/3, 1, 4/3, 3/2, 2, 3]	11
		Gain C	linf [0…100]	12
		Cross Feed	enum [OFF, ON]	13
		Mono	enum [OFF, ON]	14
		Dry	enum [OFF, ON]	15

Vintage Reverb

Effect Name	Parameters	Parameter Name	Type & Range	Par #
VREV (vintage reverb)	fffiiffffff	Pre Delay	linf [0…120]	1
		Decay	linf [0.4…4.5]	2
		Modulate	linf [0…100]	3
		Vintage	enum [OFF, ON]	4
		Position	enum [FRONT, REAR]	5
		Level	linf [-12…+12]	6
		Lo Cut	logf [10…500]	7
		Hi Cut	logf [10k…20k]	8
		Lo Multiply	logf [0.5…2]	9
		Hi Multiply	logf [0.25…1]	10

Vintage Room

Effect Name	Parameters	Parameter Name	Type & Range	Par #
VRM (vintage room)	fffffffffffi	Reverb Delay	linf [0…200]	1
		Decay	logf [0.1…20]	2
		Size	linf [2…100]	3
		Density	linf [1…30]	4
		ER Level	linf [0…100]	5
		Level	linf [-12…+12]	6
		Lo Multiply	logf [0.1…10]	7
		Hi Multiply	logf [0.1…10]	8
		Lo Cut	logf [10…500]	9
		Hi Cut	logf [200…20k]	10
		ER Left	linf [0…200]	11
		ER Right	linf [0…200]	12
		Freeze	enum [OFF, ON]	13

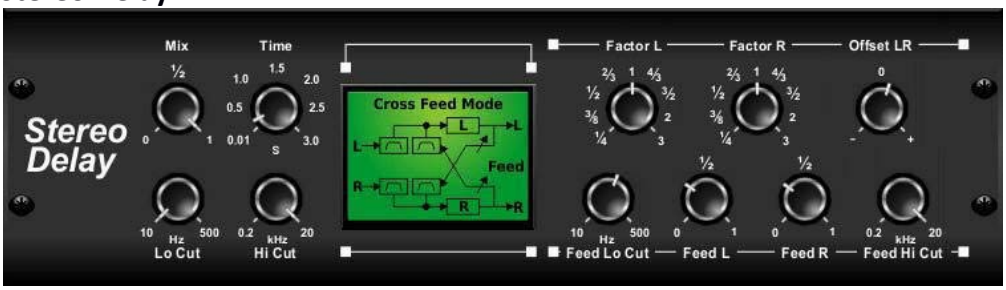
Gated Reverb

Effect Name	Parameters	Parameter Name	Type & Range	Par #
GATE (gated reverb)	fffffffff	Pre Delay	linf [0…200]	1
		Decay	linf [140…1000]	2
		Attack	linf [0…30]	3
		Density	linf [1…50]	4
		Spread	linf [0…100]	5
		Level	linf [-12…+12]	6
		Lo Cut	logf [10…500]	7
		Hi Shv Freq	logf [200…20k]	8
		Hi Shv Gain	linf [-30…0]	9
		Diffuse	linf [1…30]	10

Reverse Reverb

Effect Name	Parameters	Parameter Name	Type & Range	Par #
RVRS (reverse reverb)	fffffffff	Pre Delay	linf [0…200]	1
		Decay	logf [140…1000]	2
		Rise	linf [0…50]	3
		Diffuse	linf [1…30]	4
		Spread	linf [0…100]	5
		Level	linf [-12…+12]	6
		Lo Cut	logf [10…500]	7
		Hi Shv Freq	logf [200…20k]	8
		Hi Shv Gain	linf [-30…0]	9

Stereo Delay



Effect Name	Parameters	Parameter Name	Type & Range	Par #
DLY (stereo delay)	fffiiiffffff	Mix	linf [0…100]	1
		Time	linf [1…3000]	2
		Mode	enum [ST, X, M]	3
		Factor L	enum [1/4, 3/8, 1/2, 2/3, 1, 4/3, 3/2, 2, 3]	4
		Factor R	enum [1/4, 3/8, 1/2, 2/3, 1, 4/3, 3/2, 2, 3]	5
		Offset L/R	linf [-100…+100]	6
		Lo Cut	logf [10…500]	7
		Hi Cut	logf [200…20k]	8
		Feed Lo Cut	logf [10…500]	9
		Feed Left	linf [1…100]	10
		Feed Right	linf [1…100]	11
		Feed Hi Cut	logf [200…20k]	12

3-Tap Delay

Effect Name	Parameters	Parameter Name	Type & Range	Par #
3TAP (3-tap delay)	`fffffiffiffiii`	Time	linf [1…3000]	1
		Gain Base	linf [0…100]	2
		Pan Base	linf [-100…+100]	3
		Feedback	linf [0…100]	4
		Lo Cut	logf [10…500]	5
		Hi Cut	logf [200…20k]	6
		Factor A	enum [1/4, 3/8, 1/2, 2/3, 1, 4/3, 3/2, 2, 3]	7
		Gain A	linf [0…100]	8
		Pan A	linf [-100…+100]	9
		Factor B	enum [1/4, 3/8, 1/2, 2/3, 1, 4/3, 3/2, 2, 3]	10
		Gain B	linf [0…100]	11
		Pan B	linf [-100…+100]	12
		Cross Feed	enum [OFF, ON]	13
		Mono	enum [OFF, ON]	14
		Dry	enum [OFF, ON]	15

Stereo Chorus

Effect Name	Parameters	Parameter Name	Type & Range	Par #
CRS (stereo chorus)	fffffffffff	Speed	logf [0.05…5]	1
		Depth L	linf [0…100]	2
		Depth R	linf [0…100]	3
		Delay L	logf [0.5…50]	4
		Delay R	logf [0.5…50]	5
		Mix	linf [0…100]	6
		Lo Cut	logf [10…500]	7
		Hi Cut	logf [200…20k]	8
		Phase	linf [0…180]	9
		Wave	linf [0…100]	10
		Spread	linf [0…100]	11

Stereo Flanger

Effect Name	Parameters	Parameter Name	Type & Range	Par #
FLNG (stereo flanger)	ffffffffffff	Speed	logf [0.05…5]	1
		Depth L	linf [0…100]	2
		Depth R	linf [0…100]	3
		Delay L	logf [0.5…20]	4
		Delay R	logf [0.5…20]	5
		Mix	linf [0…100]	6
		Lo Cut	logf [10…500]	7
		Hi Cut	logf [200…20k]	8
		Phase	linf [0…180]	9
		Feed Lo Cut	logf [10…500]	10
		Feed Hi Cut	logf [200…20k]	11
		Feed	linf [-90…+90]	12

Stereo Phaser

Effect Name	Parameters	Parameter Name	Type & Range	Par #
PHAS (stereo Phaser)	fffffffffff	Speed	logf [0.05…5]	1
		Depth	linf [0…100]	2
		Resonance	linf [0…80]	3
		Base	linf [0…50]	4
		Stages	linf [2…12]	5
		Mix	linf [0…100]	6
		Wave	linf [-50…+50]	7
		Phase	linf [0…180]	8
		Env. Modulation	linf [-100…+100]	9
		Attack	logf [10…1000]	10
		Hold	logf [1…2000]	11
		Release	logf [10…1000]	12

Dimensional Chorus

Effect Name	Parameters	Parameter Name	Type & Range	Par #
DIMC (dimensional chorus)	iiiiiii	Active	enum [OFF, ON]	1
		Mode	enum [M, ST]	2
		Dry	enum [OFF, ON]	3
		Mode 1	enum [OFF, ON]	4
		Mode 2	enum [OFF, ON]	5
		Mode 3	enum [OFF, ON]	6
		Mode 4	enum [OFF, ON]	7

Mood Filter

Effect Name	Parameters	Parameter Name	Type & Range	Par #
FILT (mood filter)	ffffififfffii	Speed	logf [0.05…20]	1
		Depth	linf [0…100]	2
		Resonance	linf [0…100]	3
		Base	logf [20…15000]	4
		Mode	enum [LP, HP, BP, NO]	5
		Mix	linf [0…100]	6
		Wave	enum [TRI, SIN, SAW, SAW-, RMP, SQU. RND]	7
		Phase	linf [0…180]	8
		Env. Modulation	linf [-100…+100]	9
		Attack	logf [10…250]	10
		Release	logf [10…500]	11
		Drive	linf [0…100]	12
		4 Pole	enum [2POL, 4POL]	13
		Side Chain	enum [OFF, ON]	14

Rotary Speaker

Effect Name	Parameters	Parameter Name	Type & Range	Par #
ROTA (rotary speaker)	fffffii	Lo Speed	logf [0.1…4]	1
		Hi Speed	logf [2…10]	2
		Accelerate	linf [0…100]	3
		Distance	linf [0…100]	4
		Balance	linf [-100…+100]	5
		Mix	linf [0…100]	6
		Stop	enum [RUN, STOP]	7
		Slow	enum [SLOW, FAST]	8

Tremolo / Panner

Effect Name	Parameters	Parameter Name	Type & Range	Par #
PAN (tremolo / panner)	fffffffff	Speed	logf [0.05…5]	1
		Phase	linf [0…180]	2
		Wave	linf [-50…+50]	3
		Depth	linf [0…100]	4
		Env. Speed	linf [0…100]	5
		Env. Depth	linf [0…100]	6
		Attack	logf [10…1000]	7
		Hold	logf [1…2000]	8
		Release	logf [10…1000]	9

Delay / Chamber

Effect Name	Parameters	Parameter Name	Type & Range	Par #
D/RV (delay / chamber)	fiffffffffff	Time	linf [1…3000]	1
		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]	2
		Feed Hi Cut	logf [1000…20000]	3
		Feedback	linf [0…100]	4
		Cross Feed	linf [0…100]	5
		Balance	linf [-100…+100]	6
		Pre Delay	linf [0…200]	7
		Decay	logf [0.1…5]	8
		Size	linf [2…100]	9
		Damping	logf [1000…20000]	10
		Lo Cut	logf [10…500]	11
		Mix	linf [0…100]	12

Suboctaver

Effect Name	Parameters	Parameter Name	Type & Range	Par #
SUB (suboctaver)	iifffiifff	Active	enum [OFF, ON]	1
		Range	enum [LO, MID, HI]	2
		Dry	linf [0…100]	3
		Octave -1	linf [0…100]	4
		Octave -2	linf [0…100]	5
		Active	enum [OFF, ON]	6
		Range	enum [LO, MID, HI]	7
		Dry	linf [0…100]	8
		Octave -1	linf [0…100]	9
		Octave -2	linf [0…100]	10

Delay / Chorus

Effect Name	Parameters	Parameter Name	Type & Range	Par #
D/CR (delay / chorus)	fifffffffffff	Time	linf [1…3000]	1
		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]	2
		Feed Hi Cut	logf [1000…20000]	3
		Feedback	linf [0…100]	4
		Cross Feed	linf [0…100]	5
		Balance	linf [-100…+100]	6
		Speed	logf [0.05…4]	7
		Depth	linf [0…100]	8
		Delay	logf [0.5…50]	9
		Phase	linf [0…180]	10
		Wave	linf [0…100]	11
		Mix	linf [0…100]	12

Delay /Flanger

Effect Name	Parameters	Parameter Name	Type & Range	Par #
D/FL (delay / flanger)	fiffffffffff	Time	linf [1…3000]	1
		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]	2
		Feed Hi Cut	logf [1000…20000]	3
		Feedback	linf [0…100]	4
		Cross Feed	linf [0…100]	5
		Balance	linf [-100…+100]	6
		Speed	logf [0.05…4]	7
		Depth	linf [0…100]	8
		Delay	logf [0.5…20]	9
		Phase	linf [0…180]	10
		Feed	linf [-90…+90]	11
		Mix	linf [0…100]	12

Chorus / Chamber

Effect Name	Parameters	Parameter Name	Type & Range	Par #
CR/R (chorus / chamber)	fffffffffffff	Speed	logf [0.05…4]	1
		Depth	linf [0…100]	2
		Delay	logf [0.5…50]	3
		Phase	linf [0…180]	4
		Wave	linf [0…100]	5
		Balance	linf [-100…+100]	6
		Pre Delay	linf [0…200]	7
		Decay	logf [0.1…5]	8
		Size	linf [2…100]	9
		Damping	logf [1k…20k]	10
		Lo Cut	logf [10…500]	11
		Mix	linf [0…100]	12

Flanger / Chamber

Effect Name	Parameters	Parameter Name	Type & Range	Par #
FL/R (flanger / chamber)	fffffffffff	Speed	logf [0.05…4]	1
		Depth	linf [0…100]	2
		Delay	logf [0.5…20]	3
		Phase	linf [0…180]	4
		Feed	linf [-90…+90]	5
		Balance	linf [-100…+100]	6
		Pre Delay	linf [0…200]	7
		Decay	logf [0.1…5]	8
		Size	linf [2…100]	9
		Damping	logf [1k…20k]	10
		Lo Cut	logf [10…500]	11
		Mix	linf [0…100]	12

Modulation Delay

Effect Name	Parameters	Parameter Name	Type & Range	Par #
MODD (modulation delay)	fiffffiiffff	Time	linf [1…3000]	1
		Delay	enum [1, 1/2, 2/3, 3/2]	2
		Feed	linf [0…100]	3
		Lo Cut	logf [10…500]	4
		Hi Cut	logf [200…20k]	5
		Depth	linf [0…100]	6
		Rate	logf [0.05…10]	7
		Setup	enum [PAR, SER]	8
		Type	enum [AMB, CLUB, HALL]	9
		Decay	linf [1…10]	10
		Damping	logf [1k…20k]	11
		Balance	linf [-100…+100]	12
		Mix	linf [0…100]	13

Dual Graphic Equalizer / True Dual Graphic Equalizer

Effect Name	Parameters	Parameter Name	Type & Range	Par #
GEQ2 (dual graphic eq) TEQ2 (true dual graphic eq)	64 f	31 x Eq Level A	linf [-15…+15]	1…31
		Master Level A	linf [-15…+15]	32
		31 x Eq Level B	linf [-15…+15]	33…63
		Master Level B	linf [-15…+15]	64

Graphic Equalizer / True Graphic Equalizer

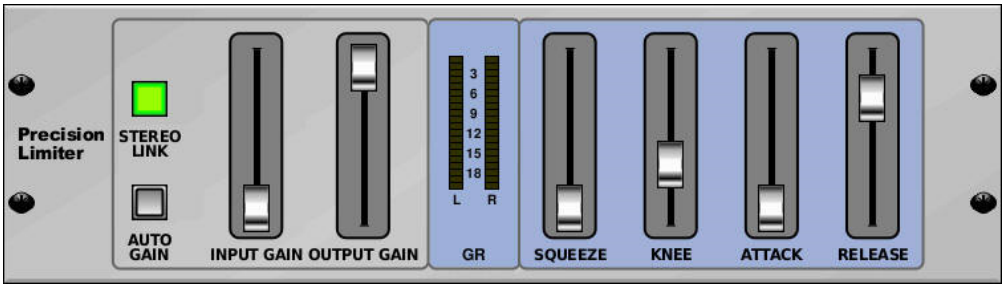
Effect Name	Parameters	Parameter Name	Type & Range	Par #
GEQ (stereo graphic eq) TEQ (true stereo graphic eq)	32 f	31 x Eq Level L/R	linf [-15…+15]	1…31
		Master Level L/R	linf [-15…+15]	32

Stereo / Dual De-Esser

Effect Name	Parameters	Parameter Name	Type & Range	Par #
DES (stereo deesser)	ffffii	Lo Band L	linf [0…50]	1
		Hi Band L	linf [0…50]	2
		Lo Band R	linf [0…50]	3
		Hi Band R	linf [0…50]	4
		Voice	enum [FEM / MALE]	5
		Mode	enum [ST / MS]	6

Effect Name	Parameters	Parameter Name	Type & Range	Par #
DES2 (dual deesser)	ffffii	Lo Band A	linf [0…50]	1
		Hi Band A	linf [0…50]	2
		Lo Band B	linf [0…50]	3
		Hi Band B	linf [0…50]	4
		Voice A	enum [FEM / MALE]	5
		Voice B	enum [FEM / MALE]	6

Precision Limiter



Effect Name	Parameters	Parameter Name	Type & Range	Par #
LIM (precision limiter)	ffffffii	Input Gain	linf [0…18]	1
		Out Gain	linf [-18…0]	2
		Squeeze	linf [0…100]	3
		Knee	linf [0…10]	4
		Attack	logf 0.05…1]	5
		Release	logf [20…2000]	6
		Stereo Link	enum [OFF, ON]	7
		Auto Gain	enum [OFF, ON]	8

Stereo / Dual Program EQ

Effect Name	Parameters	Parameter Name	Type & Range	Par #
P1A (stereo program eq)	`ifffiffffiii`	Active	enum [OFF, ON]	1
		Gain	linf [-12…+12]	2
		Lo Boost	linf [0…10]	3
		Lo Freq	enum [20, 30, 60, 100]	4
		Lo Att	linf [0…10l	5
		Hi Width	linf [0…10]	6
		Hi Boost	linf [0…10]	7
		Hi Freq	enum [3k, 4k, 5k, 8k, 10k, 12k, 16k]	8
		Alt Sel	linf [0…10]	9
		Hi Freq	enum [5k, 10k, 20k]	10
		Transformer	enum [OFF, ON]	11

Effect Name	Parameters	Parameter Name	Type & Range	Par #
P1A2 (dual program eq)	`ifffiffffiiiffffffifii`	Active A	enum [OFF, ON]	1
		Gain A	linf [-12…+12]	2
		Lo Boost A	linf [0…10]	3
		Lo Freq A	enum [20, 30, 60, 100]	4
		Lo Att A	linf [0…10l	5
		Hi Width A	linf [0…10]	6
		Hi Boost A	linf [0…10]	7
		Hi Freq A	enum [3k, 4k, 5k, 8k, 10k, 12k, 16k]	8
		Alt Sel A	linf [0…10]	9
		Hi Freq A	enum [5k, 10k, 20k]	10
		Transformer A	enum [OFF, ON]	11
		Active B	enum [OFF, ON]	12
		Gain B	linf [-12…+12]	13
		Lo Boost B	linf [0…10]	14
		Lo Freq B	enum [20, 30, 60, 100]	15
		Lo Att B	linf [0…10l	16
		Hi Width B	linf [0…10]	17
		Hi Boost B	linf [0…10]	18
		Hi Freq B	enum [3k, 4k, 5k, 8k, 10k, 12k, 16k]	19
		Alt Sel B	linf [0…10]	20
		Hi Freq B	enum [5k, 10k, 20k]	21
		Transformer B	enum [OFF, ON]	22

Stereo / Dual Midrange EQ

Effect Name	Parameters	Parameter Name	Type & Range	Par #
PQ5 (stereo midrange eq)	ifififi fi	Active	enum [OFF, ON]	1
		Gain	linf [-12…+12]	2
		Lo Freq	enum [200, 300, 500, 700, 1000]	3
		Lo Boost	linf [0…10]	4
		Mid Freq	enum [200, 300, 500, 700, 1k, 1k5, 2k, 3k, 4k, 5k, 7k]	5
		Mid Boost	linf [0…10]	6
		Hi Freq	enum [1k5, 2k, 3k, 4k, 5k]	7
		Hi Boost	linf [0…10]	8
		Transformer	enum [OFF, ON]	9

Effect Name	Parameters	Parameter Name	Type & Range	Par #
PQ5S (dual midrange eq)	ifififiiifififi fi	Active A	enum [OFF, ON]	1
		Gain A	linf [-12…+12]	2
		Lo Freq A	enum [200, 300, 500, 700, 1000]	3
		Lo Boost A	linf [0…10]	4
		Mid Freq A	enum [200, 300, 500, 700, 1k, 1k5, 2k, 3k, 4k, 5k, 7k]	5
		Mid Boost A	linf [0…10]	6
		Hi Freq A	enum [1k5, 2k, 3k, 4k, 5k]	7
		Hi Boost A	linf [0…10]	8
		Transformer A	enum [OFF, ON]	9
		Active B	enum [OFF, ON]	10
		Gain B	linf [-12…+12]	11
		Lo Freq B	enum [200, 300, 500, 700, 1000]	12
		Lo Boost B	linf [0…10]	13
		Mid Freq B	enum [200, 300, 500, 700, 1k, 1k5, 2k, 3k, 4k, 5k, 7k]	14
		Mid Boost B	linf [0…10]	15
		Hi Freq B	enum [1k5, 2k, 3k, 4k, 5k]	16
		Hi Boost B	linf [0…10]	17
		Transformer B	enum [OFF, ON]	18

Stereo / Dual Combinator

Effect Name	Parameters	Parameter Name	Type & Range	Par #
CMB (stereo combinator)	iifffififiiffffiffiffiffiffiiii	Active	enum [OFF, ON]	1
		Band Solo	enum [OFF, Bd1, Bd2, Bd3, Bd4, Bd5]	2
		Mix	linf [0…100]	3
		Attack	linf [0…19]	4
		Release	logf [20…3000]	5
		Autorelease	enum [OFF, ON]	6
		SBC speed	linf [0…10]	7
		SBC ON	enum [OFF, ON]	8
		Xover	linf [-50…+50]	9
		Xover Slope	enum [12, 48]	10
		Ratio	enum [1.1, 1.2, 1.3, 1.5, 1.7, 2, 2.5, 3, 3.5, 4, 5, 7, 10, LIM]	11
		Threshold	linf [-40…0]	12
		Gain	linf [-10…+10]	13
		Band 1 Threshold	linf [-10…+10]	14
		Band 1 Gain	linf [-10…+10]	15
		Band 1 Lock	enum [0, 1]	16
		Band 2 Threshold	linf [-10…+10]	17
		Band 2 Gain	linf [-10…+10]	18
		Band 2 Lock	enum [0, 1]	19
		Band 3 Threshold	linf [-10…+10]	20
		Band 3 Gain	linf [-10…+10]	21
		Band 3 Lock	enum [0, 1]	22
		Band 4 Threshold	linf [-10…+10]	23
		Band 4 Gain	linf [-10…+10]	24
		Band 4 Lock	enum [0, 1]	25
		Band 5 Threshold	linf [-10…+10]	26
		Band 5 Gain	linf [-10…+10]	27
		Band 5 Lock	enum [0, 1]	28
		Meter Mode	enum [GR, SBC, PEAK]	29

Effect Name	Parameters	Parameter Name	Type & Range	Par #
CMB2 (dual combinator)	iifffififiifffffiffiffiffiffiiiifffifiifiiffffiffiffiffiffiffii	Active A	enum [OFF, ON]	1
		Band Solo A	enum [OFF, Bd1, Bd2, Bd3, Bd4, Bd5]	2
		Mix A	linf [0…100]	3
		Attack A	linf [0…19]	4
		Release A	logf [20…3000]	5
		Autorelease A	enum [OFF, ON]	6
		SBC speed A	linf [0…10]	7
		SBC ON A	enum [OFF, ON]	8
		Xover A	linf [-50…+50]	9
		Xover Slope A	enum [12, 48]	10
		Ratio A	enum [1.1, 1.2, 1.3, 1.5, 1.7, 2, 2.5, 3, 3.5, 4, 5, 7, 10, LIM]	11
		Threshold A	linf [-40…0]	12
		Gain A	linf [-10…+10]	13
		Band 1 Threshold A	linf [-10…+10]	14
		Band 1 Gain A	linf [-10…+10]	15
		Band 1 Lock A	enum [0, 1]	16
		Band 2 Threshold A	linf [-10…+10]	17
		Band 2 Gain A	linf [-10…+10]	18
		Band 2 Lock A	enum [0, 1]	19
		Band 3 Threshold A	linf [-10…+10]	20
		Band 3 Gain A	linf [-10…+10]	21
		Band 3 Lock A	enum [0, 1]	22
		Band 4 Threshold A	linf [-10…+10]	23
		Band 4 Gain A	linf [-10…+10]	24
		Band 4 Lock A	enum [0, 1]	25
		Band 5 Threshold A	linf [-10…+10]	26
		Band 5 Gain A	linf [-10…+10]	27
		Band 5 Lock A	enum [0, 1]	28
		Meter Mode A	enum [GR, SBC, PEAK]	29
		Active B	enum [OFF, ON]	30
		Band Solo B	enum [OFF, Bd1, Bd2, Bd3, Bd4, Bd5]	31
		Mix B	linf [0…100]	32
		Attack B	linf [0…19]	33
		Release B	logf [20…3000]	34
		Autorelease B	enum [OFF, ON]	35
		SBC speed B	linf [0…10]	36
		SBC ON B	enum [OFF, ON]	37
		Xover B	linf [-50…+50]	38
		Xover Slope B	enum [12, 48]	39
		Ratio B	enum [1.1, 1.2, 1.3, 1.5, 1.7, 2, 2.5, 3, 3.5, 4, 5, 7, 10, LIM]	40
		Threshold B	linf [-40…0]	41
		Gain B	linf [-10…+10]	42
		Band 1 Threshold B	linf [-10…+10]	43
		Band 1 Gain B	linf [-10…+10]	44
		Band 1 Lock B	enum [0, 1]	45

		Band 2 Threshold B	linf [-10…+10]	46
		Band 2 Gain B	linf [-10…+10]	47
		Band 2 Lock B	enum [0, 1]	48
		Band 3 Threshold B	linf [-10…+10]	49
		Band 3 Gain B	linf [-10…+10]	50
		Band 3 Lock B	enum [0, 1]	51
		Band 4 Threshold B	linf [-10…+10]	52
		Band 4 Gain B	linf [-10…+10]	53
		Band 4 Lock B	enum [0, 1]	54
		Band 5 Threshold B	linf [-10…+10]	55
		Band 5 Gain B	linf [-10…+10]	56
		Band 5 Lock B	enum [0, 1]	57
		Meter Mode B	enum [GR, SBC, PEAK]	58

Stereo / Dual Fair Compressor

Effect Name	Parameters	Parameter Name	Type & Range	Par #
FAC (stereo fair compressor)	ifffffff	Active	enum [OFF, ON]	1
		Input Gain	linf [-20…0]	2
		Threshold	linf [0…10]	3
		Time	linf [1…6]	4
		Bias	linf [0…100]	5
		Gain	linf [-18…6]	6
		Balance	linf [-100…+100]	7

Effect Name	Parameters	Parameter Name	Type & Range	Par #
FAC2 (dual fair compressor) FAC1M (m/s fair compressor)	ifffffffffffff	Active	enum [OFF, ON]	1
		Input Gain	linf [-20…0]	2
		Threshold	linf [0…10]	3
		Time	linf [1…6]	4
		Bias	linf [0…100]	5
		Gain	linf [-18…6]	6
		Balance	linf [-100…+100]	7
		Input Gain	linf [-20…0]	9
		Threshold	linf [0…10]	10
		Time	linf [1…6]	11
		Bias	linf [0…100]	12
		Gain	linf [-18…6]	13
		Balance	linf [-100…+100]	14

Stereo / Dual Leisure Compressor

Effect Name	Parameters	Parameter Name	Type & Range	Par #
LEC (stereo leisure compressor)	iffif	Active	enum [OFF, ON]	1
		Gain	linf [0…100]	2
		Peak	linf [0…100]	3
		Mode	enum [COMP, LIM]	4
		Gain	linf [-18…6]	5

Effect Name	Parameters	Parameter Name	Type & Range	Par #
LEC2 (dual leisure compressor)	iffiffiffif	Active A	enum [OFF, ON]	1
		Gain A	linf [0…100]	2
		Peak A	linf [0…100]	3
		Mode A	enum [COMP, LIM]	4
		Gain A	linf [-18…6]	5
		Active B	enum [OFF, ON]	6
		Gain B	linf [0…100]	7
		Peak B	linf [0…100]	8
		Mode B	enum [COMP, LIM]	9
		Gain B	linf [-18…6]	10

Edison EX1

Effect Name	Parameters	Parameter Name	Type & Range	Par #
EDI (edison ex1)	iiiffffff	Active	enum [OFF, ON]	1
		Stereo Input	enum [ST / M/S]	2
		Stereo Output	enum [ST / M/S]	3
		ST Spread	linf [-50…+50]	4
		LMF Spread	linf [-50…+50]	5
		Balance	linf [-50…+50]	6
		Center Distance	linf [-50…+50]	7
		Out Gain	linf [-12…+12]	8

Stereo / Dual Ultimo Compressor

Effect Name	Parameters	Parameter Name	Type & Range	Par #
ULC (stereo ultimo compressor)	`iffffi`	Active	enum [OFF, ON]	1
		Input Gain	linf [-48…0]	2
		Out Gain	linf [-48…0]	3
		Attack	linf [1…7]	4
		Release	linf [1…7]	5
		Ratio	enum [4, 8, 12, 20, ALL]	6

Effect Name	Parameters	Parameter Name	Type & Range	Par #
ULC2 (dual ultimo compressor)	`iffffiiffffi`	Active A	enum [OFF, ON]	1
		Input Gain A	linf [-48…0]	2
		Out Gain A	linf [-48…0]	3
		Attack A	linf [1…7]	4
		Release A	linf [1…7]	5
		Ratio A	enum [4, 8, 12, 20, ALL]	6
		Active B	enum [OFF, ON]	7
		Input Gain B	linf [-48…0]	8
		Out Gain B	linf [-48…0]	9
		Attack B	linf [1…7]	10
		Release B	linf [1…7]	11
		Ratio B	enum [4, 8, 12, 20, ALL]	12

Sound Maxer

Effect Name	Parameters	Parameter Name	Type & Range	Par #
SON (sound maxer)	ifffifff	Active A	enum [OFF, ON]	1
		Lo Contour A	linf [0…10]	2
		Process A	linf [0…10]	3
		Out Gain A	linf [-12…+12]	4
		Active B	enum [OFF, ON]	5
		Lo Contour B	linf [0…10]	6
		Process B	linf [0…10]	7
		Out Gain B	linf [-12…+12]	8

Stereo / Dual Enhancer

Effect Name	Parameters	Parameter Name	Type & Range	Par #
ENH (stereo enhancer)	`fffffffi`	Out Gain	linf [-12…+12]	1
		Spread	linf [0…100]	2
		Bass Gain	linf [0…100]	3
		Bass Freq	linf [1…50]	4
		Mid Gain	linf [0…100]	5
		Mid Q	linf [1…50]	6
		Hi Gain	linf [0…100]	7
		Hi Freq	linf [1…50]	8
		Solo	enum [OFF, ON]	9PIT

Effect Name	Parameters	Parameter Name	Type & Range	Par #
ENH2 (dual enhancer)	`ffffffiffffffi`	Out Gain A	linf [-12…+12]	1
		Bass Gain A	linf [0…100]	2
		Bass Freq A	linf [1…50]	3
		Mid Gain A	linf [0…100]	4
		Mid Q A	linf [1…50]	5
		Hi Gain A	linf [0…100]	6
		Hi Freq A	linf [1…50]	7
		Solo A	enum [OFF, ON]	8
		Out Gain B	linf [-12…+12]	9
		Bass Gain B	linf [0…100]	10
		Bass Freq B	linf [1…50]	11
		Mid Gain B	linf [0…100]	12
		Mid Q B	linf [1…50]	13
		Hi Gain B	linf [0…100]	14
		Hi Freq B	linf [1…50]	15
		Solo B	enum [OFF, ON]	16

Stereo / Dual Exciter

Effect Name	Parameters	Parameter Name	Type & Range	Par #
EXC (stereo exciter)	`ffffffi`	Tune	logf 1k…10k]	1
		Peak	linf [0…100]	2
		Zero Fill	linf [0…100]	3
		Timbre	linf [-50…+50]	4
		Harmonics	linf [0…100]	5
		Mix	linf [0…100]	6
		Solo	enum [OFF, ON]	7

Effect Name	Parameters	Parameter Name	Type & Range	Par #
EXC2 (dual exciter)	`fffffffffffffi`	Tune A	logf 1k…10k]	1
		Peak A	linf [0…100]	2
		Zero Fill A	linf [0…100]	3
		Timbre A	linf [-50…+50]	4
		Harmonics A	linf [0…100]	5
		Mix A	linf [0…100]	6
		Solo A	enum [OFF, ON]	7
		Tune B	logf 1k…10k]	8
		Peak B	linf [0…100]	9
		Zero Fill B	linf [0…100]	10
		Timbre B	linf [-50…+50]	11
		Harmonics B	linf [0…100]	12
		Mix B	linf [0…100]	13
		Solo B	enum [OFF, ON]	14

Stereo Imager

Effect Name	Parameters	Parameter Name	Type & Range	Par #
`IMG` (stereo imager)	`fffffff`	Balance	linf [-100…+100]	1
		Mono Pan	linf [-100…+100]	2
		Stereo Pan	linf [-100…+100]	3
		Shv Gain	linf [0…12]	4
		Shv Freq	logf [100…1000]	5
		Shv Q	logf [1…10]	6
		Out Gain	linf [-12…+12]	7

Stereo / Dual Guitar Amp

Effect Name	Parameters	Parameter Name	Type & Range	Par #
AMP (stereo guitar amp)	`ffffffffi`	Preamp	linf [0…10]	1
		Buzz	linf [0…10]	2
		Punch	linf [0…10]	3
		Crunch	linf [0…10]	4
		Drive	linf [0…10]	5
		Level	linf [0…10]	6
		Low	linf [0…10]	7
		High	linf [0…10]	8
		Cabinet	enum [OFF, ON]	9

Effect Name	Parameters	Parameter Name	Type & Range	Par #
AMP2 (dual guitar amp)	`fffffffiffffffffi`	Preamp A	linf [0…10]	1
		Buzz A	linf [0…10]	2
		Punch A	linf [0…10]	3
		Crunch A	linf [0…10]	4
		Drive A	linf [0…10]	5
		Level A	linf [0…10]	6
		Low A	linf [0…10]	7
		High A	linf [0…10]	8
		Cabinet A	enum [OFF, ON]	9
		Preamp B	linf [0…10]	10
		Buzz B	linf [0…10]	11
		Punch B	linf [0…10]	12
		Crunch B	linf [0…10]	13
		Drive B	linf [0…10]	14
		Level B	linf [0…10]	15
		Low B	linf [0…10]	16
		High B	linf [0…10]	17
		Cabinet B	enum [OFF, ON]	18

Stereo / Dual Tube Stage

Effect Name	Parameters	Parameter Name	Type & Range	Par #
DRV (stereo tube stage)	fffffffff	Drive	linf [0…100]	1
		Even Har	linf [0…50]	2
		Odd Har	linf [0…50]	3
		Gain	linf [-12…+12]	4
		Lo Cut	logf [20…200]	5
		Hi Cut	logf [4k…20k]	6
		Lo Gain	linf [-12…+12]	7
		Lo Freq	logf [50…400]	8
		Hi Gain	linf [-12…+12]	9
		Hi Freq	logf [1k…10k]	10

Effect Name	Parameters	Parameter Name	Type & Range	Par #
DRV2 (dual tube stage)	fffffffffffffffffff	Drive A	linf [0…100]	1
		Even Har A	linf [0…50]	2
		Odd Har A	linf [0…50]	3
		Gain A	linf [-12…+12]	4
		Lo Cut A	logf [20…200]	5
		Hi Cut A	logf [4k…20k]	6
		Lo Gain A	linf [-12…+12]	7
		Lo Freq A	logf [50…400]	8
		Hi Gain A	linf [-12…+12]	9
		Hi Freq A	logf [1k…10k]	10
		Drive B	linf [0…100]	11
		Even Har B	linf [0…50]	12
		Odd Har B	linf [0…50]	13
		Gain B	linf [-12…+12]	14
		Lo Cut B	logf [20…200]	15
		Hi Cut B	logf [4k…20k]	16
		Lo Gain B	linf [-12…+12]	17
		Lo Freq B	logf [50…400]	18
		Hi Gain B	linf [-12…+12]	19
		Hi Freq B	logf [1k…10k]	20

Stereo / Dual Pitch Shifter

Effect Name	Parameters	Parameter Name	Type & Range	Par #
PIT (stereo pitch)	ffffff	Semitone	linf [-12…+12]	1
		Cent	linf [-50…+50]	2
		Delay	logf [1…500]	3
		Gain	Linf [0..100]	4
		Pan	Linf [-100..100]	5
		Mix	linf [0…100]	6

Effect Name	Parameters	Parameter Name	Type & Range	Par #
PIT2 (dual pitch)	fffffffffffff	Semitone 1	linf [-12…+12]	1
		Cent 1	linf [-50…+50]	2
		Delay 1	logf [1…500]	3
		Gain 1	Linf [0..100]	4
		Pan 1	Linf [-100..100]	5
		Mix 1	linf [0…100]	6
		Semitone 2	linf [-12…+12]	7
		Cent 2	linf [-50…+50]	8
		Delay 2	logf [1…500]	9
		Gain 2	Linf [0..100]	10
		Pan 2	Linf [-100..100]	11
		Mix 2	linf [0…100]	12
		Hi Cut	logf [2k…20k]	13

Wave Designer

Effect Name	Parameters	Parameter Name	Type & Range	Par #
WAV (wave designer)	ffffff	Attack A	linf [-100…+100]	1
		Sustain A	linf [-100…+100]	2
		Gain A	linf [-24…+24]	3
		Attack B	linf [-100…+100]	4
		Sustain B	linf [-100…+100]	5
		Gain B	linf [-24…+24]	6

User ASSIGN Section

User Definable Controls

This chapter describes the different settings and options linked to User Definable Controls, a.k.a. OSC command `/config/userctrl`, as presented in the **Configuration (/config) data** chapter.

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.

***Notes**:*
While the X32 Rack processing is fully compatible with the other variants of X32 or M32 series, it is the only product in the X32 family that does not feature any ASSIGN section controls. As a result, MIDI or other commands cannot be remotely triggered from the Rack. MIDI or other functions may be set up or edited, i.e. for creating scenes that shall be used on the larger variants where physical ASSIGN controls are available, but they cannot be used on the X32 Rack directly.
X32 Compact (Producer/Xrack) have different user assignable buttons numbering scheme. Button numbers go from 5 to 12 for X32 Standard, and 3 banks are available. X32 Compact Buttons 1 to 8 map to OSC numbers 5 to 12 of the X32 Standard.

Rotary Encoders

Scribbles

Buttons

Layer Selection

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

In the event a given button is assigned with a equal settings on more than one set (A, B, or C), a single action on a button will generate several OSC commands; for example if one assigns all 3 sets for button 5 witn “Mn05001” (MIDI toggle, sending note 001 on MIDI channel 5, see in the following pages), pressing button 5 in one of the 3 sets will result in sending a single MIDI note (note 1, channel 5), but 3 OSC commands issued from the X32 to listening clients.

Rotary Encoders (X32/M32 Standard)[55]

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" Pan	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" - 2 chars 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[56]	Format "Mxyyzzz" 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

[55] See restrictions at the beginning of this section as to how the physical layout of the X32 family members impacts useability of some parameters (can be set and edited, but not used).

[56] See paragraph on /-stat commands for the values returned by X32 when a button or encoder with MIDI assigned function is actioned

	"xxx"- three characters representing a remote assign: *000…007: remote 1 to remote 8* *008: Jog*
"D" Selected Channel	Format "Dx" x: Parameter "x": *@: Fader* *A: Gate threshold* *B: Gate range* *C: Gate attack* *D: Gate hold* *E: Gate release* *F: Dyn. threshold* *G: Dyn. ratio* *H: Dyn. knee* *I: Dyn. mgain* *J: Dyn. attack* *K: Dyn. hold* *L: Dyn. release*
"U" X-Live!	Format "Ux" x: Parameter "x": *0: X-Live! Locator(MarkerPosition)*[57] *1: X-Live! Marker List (Navigate)* *2: X-Live! Session List (Navigate)*

[57] See Button assignment for additional X-Live! Marker and Session functions (set, play, etc.)

Buttons (X32/M32 Standard)[58]

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: "Pxxyz", 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*

[58] See restrictions at the beginning of this section as to how the physical layout of the X32 family members impacts useability of some parameters (can be set and edited, but not used).

	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[59]	Format: "Mxyyzzz", 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[60]	Format: "Mxyyzzz", 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*

[59]

[60] See paragraph on /-stat commands for the values returned by X32 when a button or encoder with MIDI assigned function is actioned

	014…017: UP, DOWN, LEFT, RIGHT *018: STOP* *019: PLAY* *020: REC* *021: FF* *022: REW* *023: MRK/RTZ* *024: CYCLE* *025: SCRUB* *026: NDG/SHUT* *027: DROP/IN* *028: REP/OUT* *029: CLI/OFF* *030: READ* *031: WRITE* *032: TOUCH* *033: TRIM* *034: LATCH*

Button assignment	Associated function
"S" Show Control	Format: "S9xx", xx: Control Number "xx": *00: PREV* *01: NEXT* *02: UNDO* *03: GO*

Button assignment	Associated function
"S" Cue Recall	Format: "S4xx", xx: Cue Number "xx": *00…99: Cue number*

Button assignment	Associated function
"S" Scene Recall	Format: "S0xx", xx: Scene Number "xx": *00…99: Scene number*

Button assignment	Associated function
"S" Snippet Recall	Format: "S2xx", xx: Snippet Number "xx": *00…99: Snippet number*

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

Button assignment	Associated function
“U” SD Recorder	Format: “Uxx”, xx: control “xx”: *00: Stop* *01: Play* *02: Record* *03: Pause* *04: Play/Stop* *05: Play/Pause* *06: Add Marker* *07: Previous Marker* *08: Next Marker* *09: Play Marker* *10: Select Marker* *11: Select Session* *12: USB Play Back* *13: Chn. Routing*

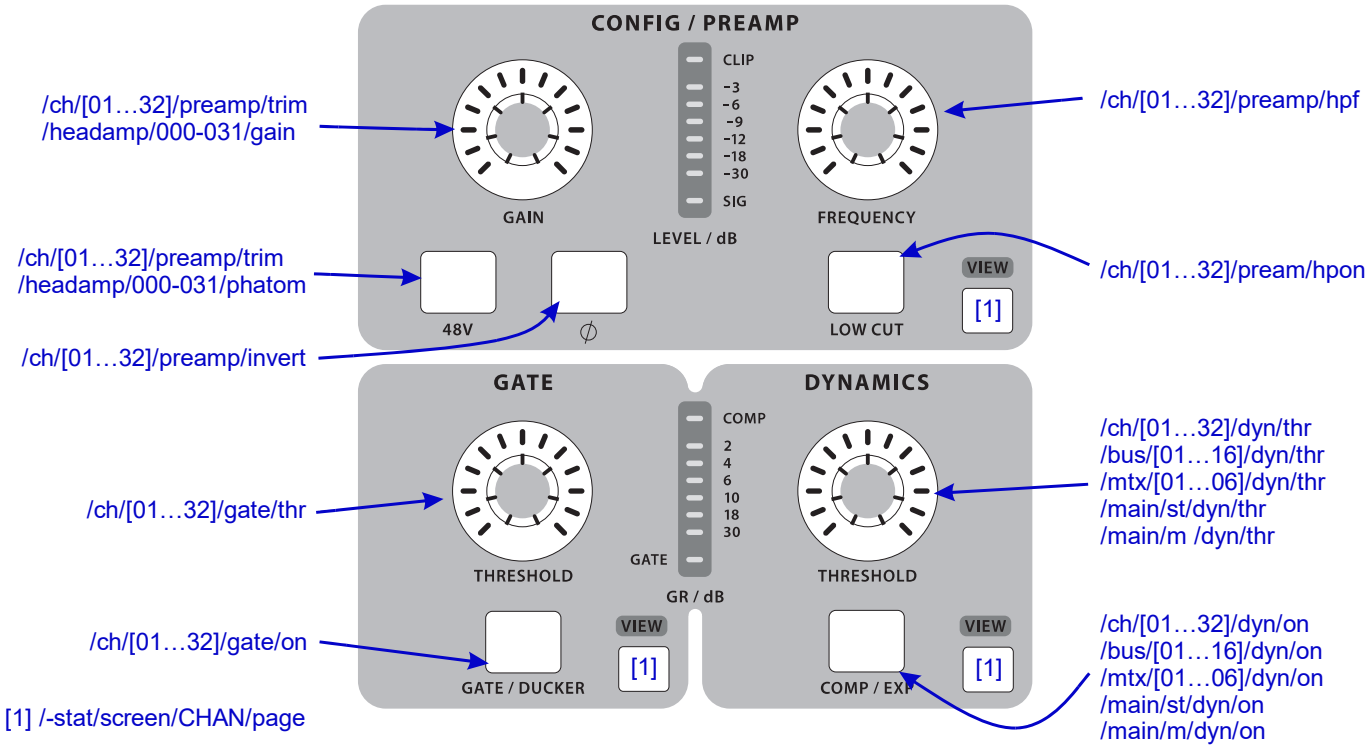
Button assignment	Associated function
“L” SD Session, SD Marker	Format: “Lxxx”, xxx: value “xxx”: *0…99 : X-Live! session* *100…199: X-Live! marker*

Button assignment	Associated function
“A” Automix Enable	Format: “Ax”, x: group “x”: *0: X 1: Y*

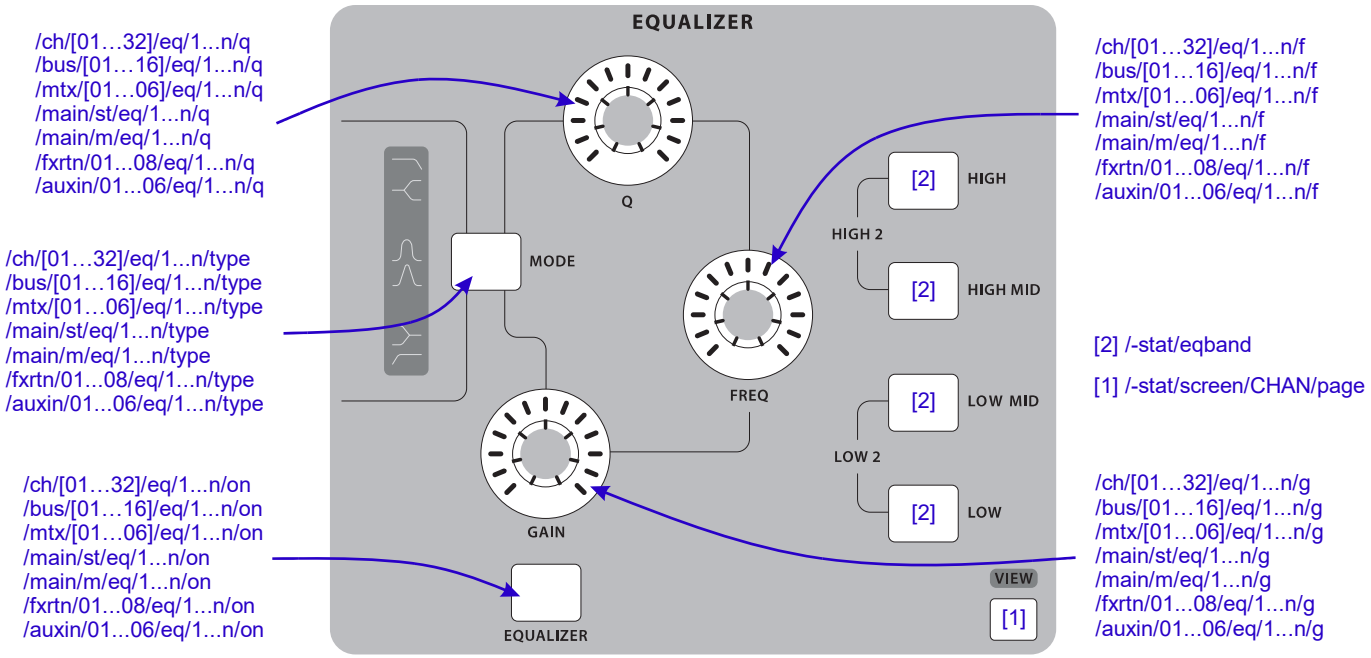
Appendices

Appendix – X32 Standard: Surface Controls

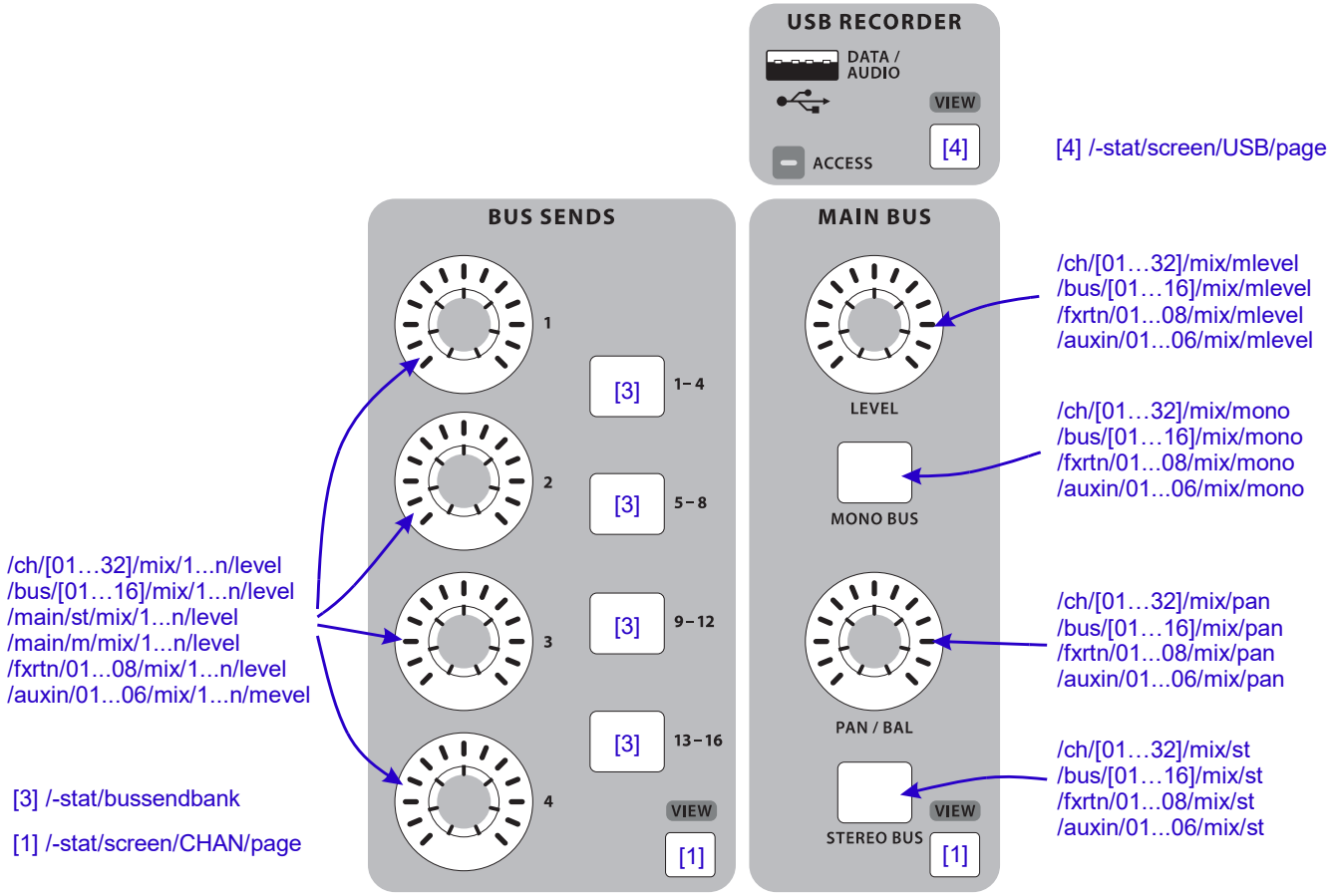
Config/Preamp/Gate/Dynamics



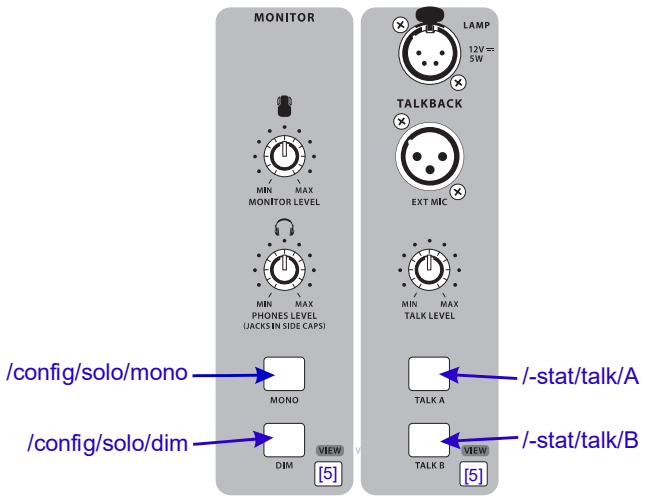
Equalizer



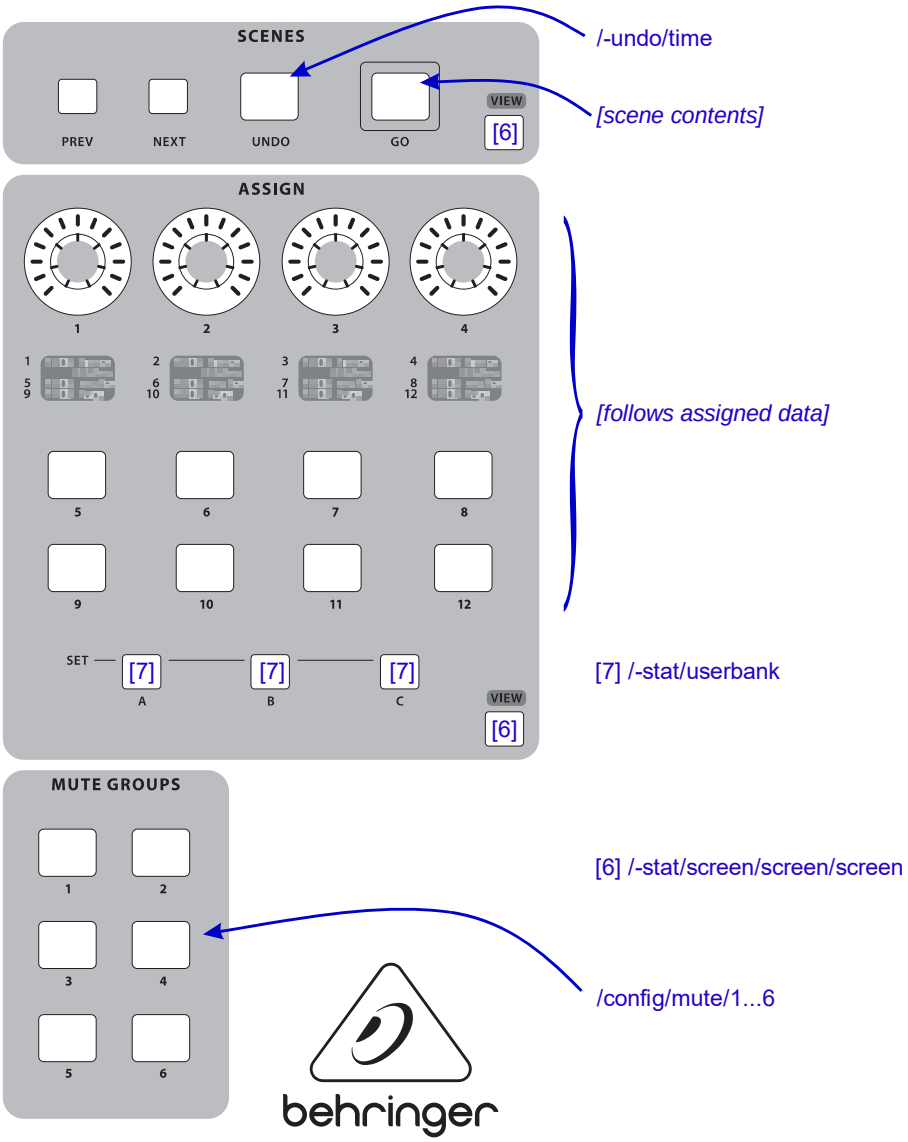
USB/Bus Sends/Main Bus



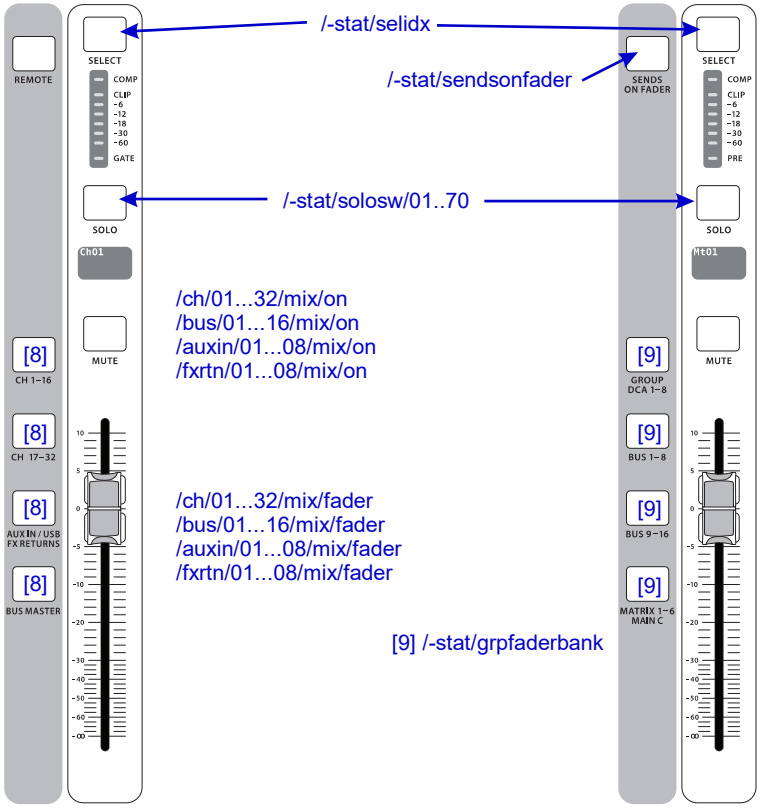
Monitor/Talkback



Scenes/Assign/Mute Groups



Fader Strips

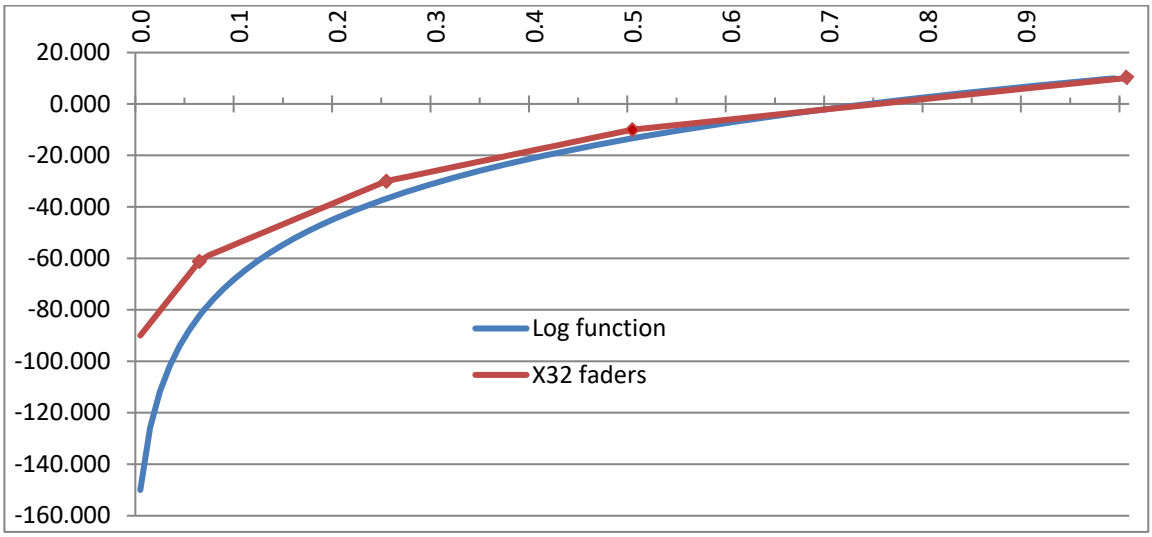


Appendix – Converting X32 fader data to decibels and vice-versa

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 [0.0, 1.0] to dB [-90 , +10] with value 0 matching –oo, 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:[-oo, +10]
      if (f >= 0.5)           d = f * 40. - 30.;      // max dB value: +10.
      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.;     // min dB value: -90 or -oo[61].

// 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 = int(f * 1023.5) / 1023.0;[62]
```

[61] A member wrote the X32/M32 have an internal –oo value of -144dB, but this doesn't appear in any data reported by X32.
[62] Thanks to F. Homman for correcting the rounding function initially proposed.

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 2127 lines of editable data representing the working state of the X32/M32 and controlling almost all X32/M32 parameters.

Starting with FW realease 4.02, X32/M32 support a special scene file that can be automatically loaded at boot time. The file must be in the root directory of the USB drive and be named CustomBootState.scn. This can be quite useful for ensuring a proper reset of the desk to a specific set of parameters at start time, rather than relying on a manual load of scene 0 or a console reset state from the config screen.

/config/chlink	*/ch/[01…32]/gate/filter*	*/mtx/[01…06]/dyn/filter*
/config/auxlink	*/ch/[01…32]/dyn*	*/mtx/[01…06]/insert*
/config/fxlink	*/ch/[01…32]/dyn/filter*	*/mtx/[01…06]/eq*
/config/buslink	*/ch/[01…32]/insert*	*/mtx/[01…06]/eq[1…6]*
/config/mtxlink	*/ch/[01…32]/eq*	*/mtx/[01…06]/mix*
/config/mute	*/ch/[01…32]/eq/[1…4]*	*/main/st/config*
/config/linkcfg	*/ch/[01…32]/mix*	*/main/st/dyn*
/config/mono	*/ch/[01…32]/mix/[01…16]*	*/main/st/dyn/filter*
/config/solo	*/ch/[01…32]/grp*	*/main/st/insert*
/config/talk	*/ch/[01…32]/automix*	*/main/st/eq*
/config/talk/A	*/auxin/[01…08]/config*	*/main/st/eq[1…6]*
/config/talk/B	*/auxin/[01…08]/preamp*	*/main/st/mix*
/config/osc	*/auxin/[01…08]/eq*	*/main/st/mix/[01…06]*
/config/userrout/out	*/auxin/[01…08]/eq/[1…4]*	*/main/m/config*
/config/userrout/in	*/auxin/[01…08]/mix*	*/main/m/dyn*
/config/routing	*/auxin/[01…08]/mix/[01…16]*	*/main/m/dyn/filter*
/config/routing/IN	*/auxin/[01…08]/grp*	*/main/m/insert*
/config/routing/AES50A	*/fxrtn/[01…08]/config*	*/main/m/eq*
/config/routing/AES50B	*/fxrtn/[01…08]/eq*	*/main/m/eq[1…6]*
/config/routing/CARD	*/fxrtn/[01…08]/eq[1…4]*	*/main/m/mix*
/config/routing/OUT	*/fxrtn/[01…08]/mix*	*/main/m/mix/[01…06]*
/config/routing/PLAY	*/fxrtn/[01…08]/mix/[01…16]*	*/dca/[1…8]*
/config/userctrl/{A,B,C}	*/fxrtn/[01…08]/grp*	*/dca/[1…8]/config*
/config/userctrl/{A,B,C}/enc	*/bus/[01…16]/config*	*/fx/[1…8]*
/config/userctrl/{A,B,C}/btn	*/bus/[01…16]/dyn*	*/fx/[1…8]/source*
/config/tape	*/bus/[01…16]/dyn/filter*	*/fx/[1…8]/par*
/config/amixenable	*/bus/[01…16]/insert*	*/outputs/main/[01…16]*
/config/dp48	*/bus/[01…16]/eq*	*/outputs/main/[01…16]/delay*
/config/dp48/assign	*/bus/[01…16]/eq[1…6]*	*/outputs/aux/[01…06]*
/config/dp48/link	*/bus/[01…16]/mix*	*/outputs/p16/[01…16]*
/config/dp48/grpname	*/bus/[01…16]/mix/[01…06]*	*/outputs/p16/[01…16]/iQ*
/ch/[01…32]/config	*/bus/[01…16]/grp*	*/outputs/aes/[01…02]*
/ch/[01…32]/delay	*/mtx/[01…06]/config*	*/outputs/rec/[01…02]*
/ch/[01…32]/preamp	*/mtx/[01…06]/preamp*	*/headamp/[000…127]*
/ch/[01…32]/gate	*/mtx/[01…06]/dyn*	

A scene file starts with a line such as:

```
#4.0# "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, or X32nodes) 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

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:

```
#4.0# "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]	*/config/routing/PLAY/AUX*	*/fxrtn/[01…08]/mix/fader*
/fx/[1…8]/source	*/config/routing/routswitch*	*/fxrtn/[01…08]/mix/pan*
/fx/[1…8]/par	*/outputs/main/[01…16]*	*/fxrtn/[01…08]/mix/on*
/config/solo	*/outputs/main/[01…16]/delay*	*/fxrtn/[01…08]/mix/[01…16]*
/config/dp48	*/outputs/aux/[01…06]*	*/fxrtn/[01…08]/mix/mono*
/config/dp48/assign	*/outputs/p16/[01…16]*	*/fxrtn/[01…08]/mix/mlevel*
/config/dp48/link	*/outputs/p16/[01…16]/iQ*	*/bus/[01…16]/config*
/config/dp48/grpname	*/outputs/aes/[01…02]*	*/bus/[01…16]/eq*
/config/talk	*/headamp/[000…127]*	*/bus/[01…16]/eq/[1…6]*
/config/talk/A	*/ch/[01…32]/preamp*	*/bus/[01…16]/dyn*
/config/talk/B	*/ch/[01…32]/delay*	*/mtx/[01…06]/config*
/config/routing/routswitch	*/ch/[01…32]/config*	*/mtx/[01…06]/eq*
/config/routing/IN/1-8	*/ch/[01…32]/eq*	*/mtx/[01…06]/eq/[1…6]*
/config/routing/IN/9-16	*/ch/[01…32]/eq/[1…4]*	*/mtx/[01…06]/dyn*
/config/routing/IN/17-24	*/ch/[01…32]/gate*	*/mtx/[01…06]/dyn/filter*
/config/routing/IN/25-32	*/ch/[01…32]/gate/filter*	*/mtx/[01…06]/insert*
/config/routing/IN/AUX	*/ch/[01…32]/dyn*	*/mtx/[01…06]/mix/fader*
/config/routing/AES50A/1-8	*/ch/[01…32]/dyn/filter*	*/mtx/[01…06]/mix/on*
/config/routing/AES50A/9-16	*/ch/[01…32]/insert*	*/main/st/config*
/config/routing/AES50A/17-24	*/ch/[01…32]/grp*	*/main/st/eq*
/config/routing/AES50A/25-32	*/ch/[01…32]/mix/fader*	*/main/st/eq/[1…6]*
/config/routing/AES50A/33-40	*/ch/[01…32]/mix/pan*	*/main/st/dyn*
/config/routing/AES50A/41-48	*/ch/[01…32]/mix/on*	*/main/st/dyn/filter*
/config/routing/AES50B/1-8	*/ch/[01…32]/mix/[01…16]*	*/main/st/insert*
/config/routing/AES50B/9-16	*/ch/[01…32]/mix/mono*	*/main/st/mix/fader*
/config/routing/AES50B/17-24	*/ch/[01…32]/mix/mlevel*	*/main/st/mix/pan*
/config/routing/AES50B/25-32	*/auxin/[01…06]/preamp*	*/main/st/mix/on*
/config/routing/AES50B/33-40	*/auxin/[01…06]/config*	*/main/st/mix/[01…06]*
/config/routing/AES50B/41-48	*/auxin/[01…06]/eq*	*/main/m/config*
/config/routing/CARD/1-8	*/auxin/[01…06]/eq/[1…4]*	*/main/m/eq*
/config/routing/CARD/9-16	*/auxin/[01…06]/grp*	*/main/m/eq/[1…6]*
/config/routing/CARD/17-24	*/auxin/[01…06]/mix/fader*	*/main/m/dyn*
/config/routing/CARD/25-32	*/auxin/[01…06]/mix/pan*	*/main/m/dyn/filter*
/config/routing/OUT/1-4	*/auxin/[01…06]/mix/on*	*/main/m/insert*
/config/routing/OUT/5-8	*/auxin/[01…06]/mix/[01…16]*	*/main/m/mix/fader*
/config/routing/OUT/9-12	*/auxin/[01…06]/mix/mono*	*/main/m/mix/on*
/config/routing/OUT/13-16	*/auxin/[01…06]/mix/mlevel*	*/main/m/mix/[01…06]*
/config/routing/PLAY/1-8	*/fxrtn/[01…08]/config*	*/dca/[1…8]/config*
/config/routing/PLAY/9-16	*/fxrtn/[01…08]/eq*	*/dca/[1…8]/fader*
/config/routing/PLAY/17-24	*/fxrtn/[01…08]/eq/[1…4]*	*/dca/[1…8]/on*
/config/routing/PLAY/25-32	*/fxrtn/[01…08]/grp*	

Appendix – Channel, Effects, Routing, and AES/DP48 preset files data elements

The table below lists the control elements that can be found in a Channel, Library, Routing, or AES/DP48 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:

```
#4.0# <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*	Effects*	Routing	AES/DP48
/config */delay* */preamp* */gate* */gate/filter* */dyn* */dyn/filter* */eq* */eq/[1…6]* */mix* */mix/[01…16]* */headamp/[000…127]*	*/type* */source* */par*	*/config/routing/routswitch* */config/routing/IN* */config/routing/AES50A* */config/routing/AES50B* */config/routing/CARD* */config/routing/OUT* */config/routing/PLAY* */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]*	*/config/dp48* */config/dp48/assign* */config/dp48/link* */config/dp48/grpname*

* 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 does not apply to a specific channel or effect number.

Appendix – X32/M32 Custom Boot and Lock Screens

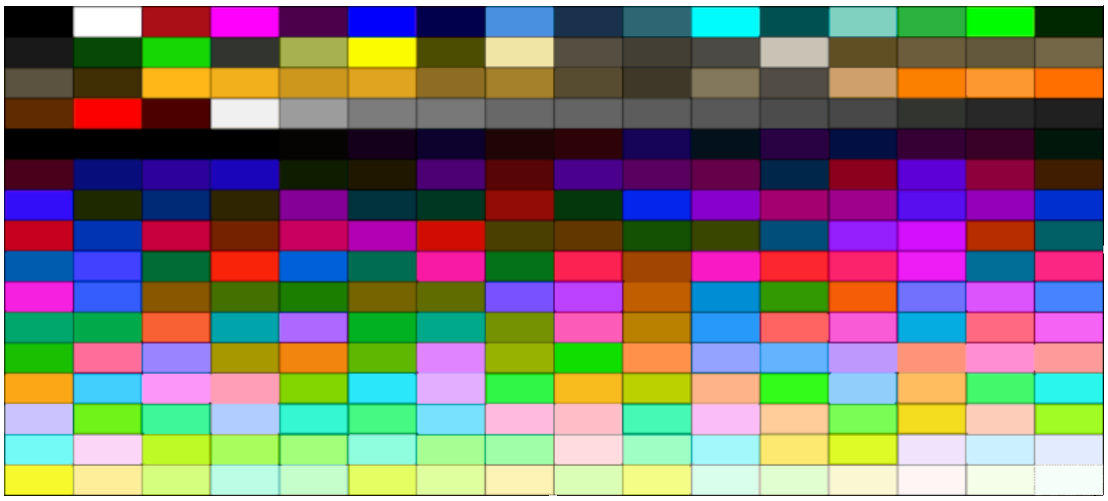
Firmware versions before 4.01 only propose a customizable boot screen which name must be ***logo.bmp***. Custom lock screen image is not supported.

Starting with Firmware version 4.01, the X32/M32 family of products accepts customizable boot and lock screens; The boot or lock screens are images and have to be on the root of the USB drive, where to software looks for them. If the images don't fill the listed criteria, the default boot/lock screens will be shown. Images smaller than the max screen size will be centered on the screen.

Nothing is copied in the internal memory; the USB drive must be plugged in for image to be loaded. The boot and lock images must be named ***CustomBoot.bmp*** and ***CustomLock.bmp***, respectively. They must be placed root of the USB drive.

The files must fit the following format specifications:

- Bitmap only (.bmp) in 8bit indexed mode
- Minimum size: 10 x 10 p;, Maximum size: 799 x 399 px
- Colour palette embedded; Do not change the color order and use dithering for smoother pictures
- Background color will be 1st color in the palette, default text color the 2nd color
- Boot text color will be 2nd color of the palette



You can download the bitmap file above showing the X32 color palette at:
https://drive.google.com/file/d/1BrKN4UxHpjJ_iXx39ulLIsooQN0znsv-/view?usp=sharing

You can download the X32 palette as an xml file at:
https://drive.google.com/file/d/15X4WADJHvql9dOCE09QeWI55ekHKLzCa/view?usp=sharing

Convert the 256 entries color palette to the format of your choice (i.e. needed by your drawing software) using tools such as SwatchBooker, ColorZilla, Paletton, COLOURlovers, Chroma, Colormind, Coolors, Adobe Color CC, to name a few.

It seems Corel PhotoPaint and PhotoShop [maybe others too] do not save the .bmp file in the expected Windows bmp header flavor (there are many choices). The Irfanview program does, so we recommend using this one.

Below are steps gathered from a Facebook link on how to create a custom boot or custom lock screen; Just follow these simple steps (remember you will need FW 4.0 at least) using Irfanview:

You'll need this software as it is free, and the easiest to use: www.irfanview.net

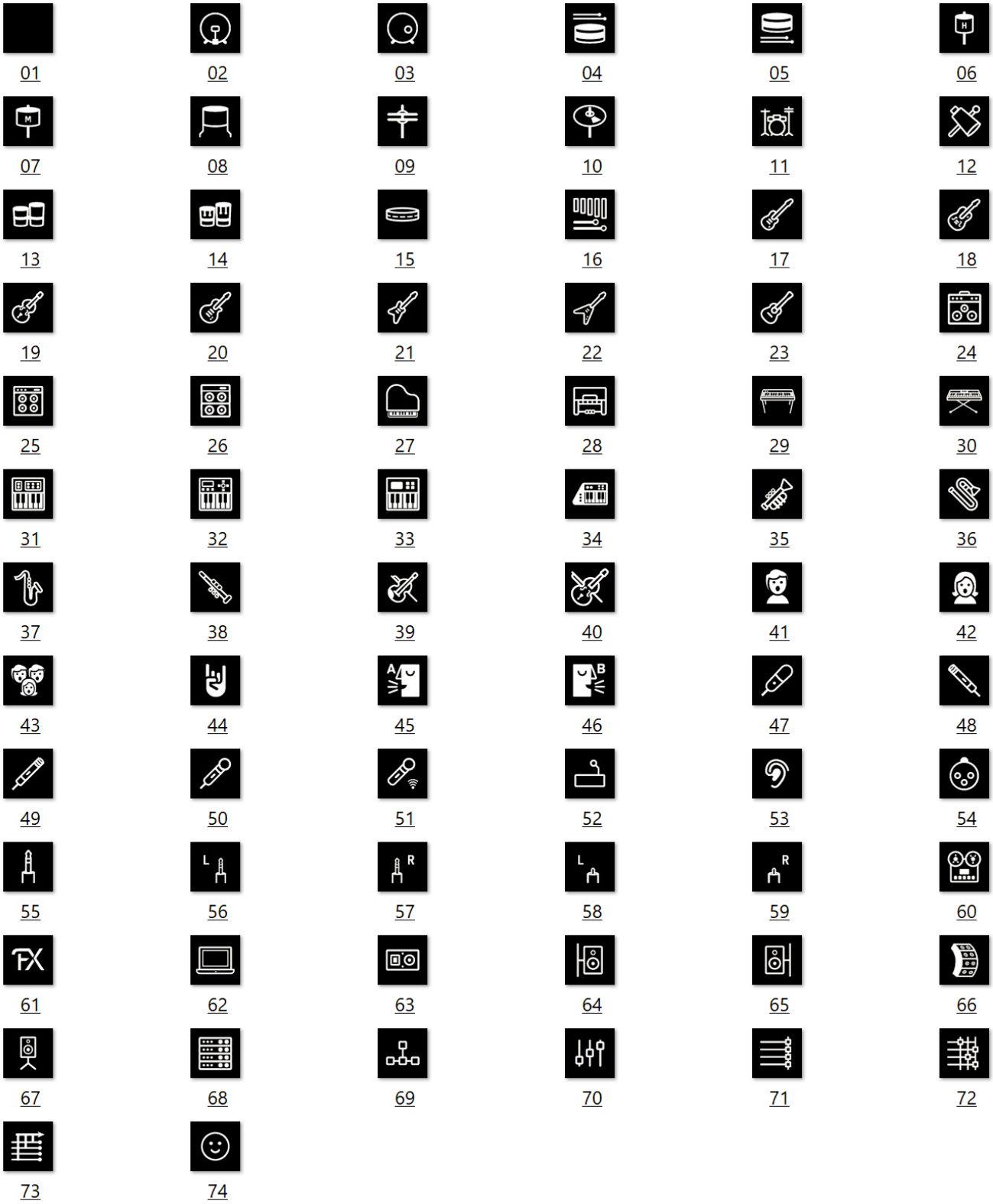
- To generate the necessary colour palette, open the image customlock.bmp using the link provided above in this document.
- When opened, export the color palette (Left top on screen) Image -> Palette -> Export Palette. That's the most important step!!! Otherwise the console will not use the desired colors while locking.
- Load up any image (or create it) you'd like to have on your lock screen (please note that this works only with the USB stick containing the image plugged in. You can remove the stick when locked, but you'll need the drive everytime you lock it)
- For the lock screen, you can import any *.bmp image (max resolution 799 x 399 pixels). If not already a 256 colors index file, you will need to convert it to 256 colors, such as by decreasing the colordepth to 256 colors (Image -> Decrease Color Depth)
- Next thing you'll need to do is convert the image to the colour palette you just generated: Image -> Palette -> Import palette.
- Make sure that the image's resolution doesn't exceed the maximum size of 799 x 399 pixels. You can resize it at "Image -> Resize Resample"
- Save the result as a BMP image, name it "CustomLock.bmp" and copy it to the root directory of your USB drive.

In order to create a custom boot image, just repeat these steps and rename the file to spell "CustomBoot.bmp".

Appendix – X32/M32 Icons

X32/M32 Icons are numbered *01…74* and shown below as names and icons below.

1. None	45. Talk A
2. Kick Back	46. Talk B
3. Kick Front	47. Large Diaphragm Mic
4. Snare Top	48. Condenser Mic Left
5. Snare Bottom	49. Condenser Mic Right
6. High Tom	50. Handheld Mic
7. Mid Tom	51. Wireless Mic
8. Floor Tom	52. Podium Mic
9. Hi-Hat	53. Headset Mic
10. Ride	54. XLR Jack
11. Drum Kit	55. TRS Plug
12. Cowbell	56. TRS Plug Left
13. Bongos	57. TRS Plug Right
14. Congas	58. RCA Plug Left
15. Tambourine	59. RCA Plug Right
16. Vibraphone	60. Reel to Reel
17. Electric Bass	61. FX
18. Acoustic Bass	62. Computer
19. Contrabass	63. Monitor Wedge
20. Les Paul Guitar	64. Left Speaker
21. Ibanez Guitar	65. Right Speaker
22. Washburn Guitar	66. Speaker Array
23. Acoustic Guitar	67. Speaker on a Pole
24. Bass Amp	68. Amp Rack
25. Guitar Amp	69. Controls
26. Amp Cabinet	70. Faders
27. Piano	71. MixBus
28. Organ	72. Matrix
29. Harpsichord	73. Routing
30. Keyboard	74. Smiley
31. Synthesizer 1	
32. Synthesizer 2	
33. Synthesizer 3	
34. Keytar	
35. Trumpet	
36. Trombone	
37. Saxophone	
38. Clarinet	
39. Violin	
40. Cello	
41. Male Vocal	
42. Female Vocal	
43. Choir	
44. Hand Sign	


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

Appendix – X32 MIDI Implementation[63]

MIDI RX > SCENES

Whenever program change messages in the range 1-100 are received on MIDI CH01, the corresponding cue/scene/snippet of the X32 internal show memory will be loaded.

General Notes and Requirements

Setup / remote

- MIDI In/Out check marks must be set according to the connection on which MIDI input will be accepted (via physical MIDI connectors on X32 or S16, or via USB expansion card)
- MIDI In/Out check mark must be set for "MIDI Receive Prog Change"

Setup / global

- When "Confirm Pop-Ups" or "Scene Load" is active, MIDI scene recalls will only become active after manual confirmation on the console
- If you prefer activating scenes via MIDI program changes immediately, un-check the "Confirm Pop-Ups" / "Scene Load" tick mark

Scenes View / home

- Valid cues/scenes/snippets must be stored to the internal X32 show file. It is not possible to recall empty scenes, snippets or cues via MIDI.
- The scope of changes applied by a MIDI scene recall depends on the Scene Safes, Parameter Safes and Channel Safes settings, as if the scene was recalled locally.

MIDI TX > CUES

Every Cue can be assigned one specific MIDI command/event. Each time the Cue is loaded the MIDI command will be sent out once.

Possible choices for MIDI commands/events to be found on Scenes View/MIDI page:

- Off > no message will be sent upon scene load
- Program Change > select the MIDI Channel and the Program Number (using knobs 4/5 at the main display)
- Control Change > select the MIDI Channel, Controller number and value (using knobs 4-6 at the main display)
- Note > select the MIDI Channel, Note number and velocity (using knobs 4-6 at the main display)

will send out a Note On command directly followed by the same Note Off command

[63] This section is taken from the Behringer MIDI implementation document dated Sep. 18, 2017

PERMANENT MIDI ASSIGNMENTS OVERVIEW (REMOTE OFF)								
MIDI RX	RX	TX	Midi Ch	Controller	Value	Description	RX comment	TX Comment
Scenes	x	x	1	Prog Change	0-99	Load Scene 0-99		
Snippet	x	x	2	Prog Change	0-99	Load Snippet 0-99		
Cue	x	x	3-6	Prog Change	0-127	Load Cue	Ch3 -> Cue 1-127, Ch4 -> Cue 128-255, etc.	
Faders	x	x	1	CC0-79	0-127	Fader Level	Value 95 = 0dB	
Mute	x	x	2	CC0-79	on (127) / off (0), toggle latching	Mute Ch01-32, Aux1-8, FX1L-4R, Bus 1-16, Mtx 1-6, Main LR, Main C, DCA 1-8		
MuteGroups	x	x	2	CC80-85	on (127) / off (0), toggle latching	Mute Group		
Pan	x	x	3	CC0-79	1-127	Panorama/Balance	Value 64 = Pan Mid	
SD-Session*	x	x	7	Prog Change	0-99	Load Session 1-100	Loads session # from SD card for playback	Session # loaded for playback
SD-Marker*	x	x	7	CC00	127	Add Marker	Stores a new marker at current location	Returns new Marker number after Adding
				CC1-99	127	GoToMarker '1-99	Sets the current song position to a specific Marker	Sends Marker number upon Go operation
				CC101	127	Previous Marker (127)	Sets the song position to the previous Marker	Returns the Marker number
				CC102	127	Next Marker (127)	Sets the song position to the next Marker	Returns the Marker number
				CC103	0/127	Input Routing REC(0)/PLAY(127)	Toggles the input routing between Rec and Play	Sends the input routing status Rec or Play
				CC104	127	STOP (127)	Stops Rec/Play and resets song location to zero	Sends a Stop Rec/Play cmd
				CC105	127/0	PLAY (127) / PAUSE (0)	Toggles Play/Pause operation	Sends the Play/Pause status
				CC106	127	Pause (127)	Pauses the current operation (i.e. Play or Rec)	Sends a Pause cmd
				CC107	127/0	REC (127) / STOP (0)	Ttoggles Recording/Stop operation	Sends the Record/Stop status
				CC110	0/1	SD-1 (0) / SD-2 (1)	Selects the active SD-Card slot	Sends active SD-Card after select operation

*: only available when X-Live! expansion card is installed in the console

X-TOUCH (XCTL) or MC REMOTE CONTROL OVERVIEW						Only available when other MIDI CC or Sysex comms are off, and setup/config Transport Control is on X-Live!
Sections:			Midi Ch	MIDI "note"	Value On/Off	Default MC / X-TOUCH names	XCTL function for X32 series
All other sections than the ones stated below remain as they are in current XCTL implementation						
Global View						
			1	E 3	127/0	Name/Value	n/a
			1	F 3	127/0	SMPTE/Beats	Toggles between X-LIVE (SMPTE) and X-CNTL (BEATS) main section assignments
			1	D 4	127/0	F9, MIDI Tracks	SoF Aux Bus 1
			1	D# 4	127/0	F10, Inputs	SoF Aux Bus 2
			1	E 4	127/0	F11, Audio Tracks	SoF Aux Bus 3
			1	F 4	127/0	F12, Audio Instruments	SoF Aux Bus 4
			1	F# 4	127/0	F13, Aux	SoF Aux Bus 5
			1	G 4	127/0	F14, Buses	SoF Aux Bus 6
			1	G# 4	127/0	F15, Outputs	SoF Aux Bus 7
			1	A 4	127/0	F16, User	SoF Aux Bus 8
FUNCTION							
			1	F# 3	127/0	F1	SoF Aux Bus 9
			1	G 3	127/0	F2	SoF Aux Bus 10
			1	G# 3	127/0	F3	SoF Aux Bus 11
			1	A 3	127/0	F4	SoF Aux Bus 12
			1	A# 3	127/0	F5	SoF Aux Bus 13
			1	B 3	127/0	F6	SoF Aux Bus 14
			1	C 4	127/0	F7	SoF Aux Bus 15
			1	C# 4	127/0	F8	SoF Aux Bus 16
MODIFY							
			1	A# 4	127/0	Shift	FX1 on v-pots
			1	B 4	127/0	Option	FX2 on v-pots
			1	C 5	127/0	Control	FX3 on v-pots
			1	C# 5	127/0	Alt	FX4 on v-pots
AUTOMATION							
			1	D 5	127/0	Read	Mute Grp 1
			1	D# 5	127/0	Write	Mute Grp 2
			1	E 5	127/0	Trim	Mute Grp 3
			1	F 5	127/0	Touch	Mute Grp 4
			1	F# 5	127/0	Latch	Mute Grp 5
			1	G 5	127/0	Group	Mute Grp 6
UTILITY							
			1	G# 5	127/0	Save	FX5 on v-pots
			1	A 5	127/0	Undo	FX6 on v-pots
			1	A# 5	127/0	Cancel	FX7 on v-pots
			1	B 5	127/0	Enter	FX8 on v-pots
TRANSPORT							
			1	C 6	127/0	Marker	Add Marker at current locator position
			1	C# 6	127/0	Nudge	Save locator postion to selected Marker
			1	D 6	127/0	Cycle	Input Routing Recording/Playback
			1	D# 6	127/0	Drop	
			1	E 6	127/0	Replace	USB Play Folder
			1	F 6	127/0	Click	AutoMix X

			1	F# 6	127/0	Solo	AutoMix Y
			1	G 6	127/0	REW	Previous Marker
			1	G# 6	127/0	FFW	Next Marker
			1	A 6	127/0	STOP	Stop
			1	A# 6	127/0	PLAY	Play/Pause
			1	B 6	127/0	REC	Record Session
			1	C 7	127/0	Up	Select next higher (recent) session number (indicated in Assignment LED display)
			1	C#7	127/0	Down	Select next lowe (older) session number (indicated in Assignment LED display)
			1	D7	127/0	SCRUB	Preview the selcted Marker/Locator while pressing, return and pause on release
			1	D#7	127/0	Zoom Mode	Load the selected Session that is indicated in the Assignment LED display: > off= loaded > flashing= selected not loaded yet
			1	E7	127/0	Left	
			1	F7	127/0	Right	
JOG WHEEL							
			1	CC60	CW(1), CCW(65)	Wheel	Adjust Locator > STOP = +/- 1 s steps per click > PLAY/PAUSE = +/-10 ms steps > indicate absolute song position in display

MIDI RX > ASSIGN

Whenever assignable controls are set up for transmitting MIDI commands, reception of that same command (status or continuous) will be reflected on the respective assignable control element (button light, encoder LED collar).

MIDI TX > ASSIGN

We restricted the user assignable MIDI commands to some generic elements, in order to keep things simple enough:

Encoders 1-4 > can be assigned to sending control changes, program changes or notes

- parameters are currently ‘Channel’ and ‘Value’
- for CC and Note commands ‘Value’ = controller number/note number, and the encoder rotation determines the controller value/note-on velocity
- for Program Changes only the channel is specified, and the encoder rotation determines the program number

Buttons 5-12 > can be operated in two modes, ‘MIDI Push’ (non-latching) for momentary commands, or ‘MIDI Toggle’ (latching) for static commands

MIDI Push:

- can be assigned to sending control changes, program changes or notes
- parameters are currently ‘Channel’ and ‘Value’
- for CC and Note commands ‘Value’ 0…127 = controller number/note number, and the button momentarily toggles the controller value/note-on velocity
- to 127 (depressed/released]
- for Program Changes ‘Value’ 0…127 = program/preset number, that will be sent upon pressing the button

MIDI Toggle:

- can be assigned to sending control changes or notes
- parameters are currently ‘Channel’ and ‘Value’
- for CC and Note commands ‘Value’ 0…127 = controller number/note number, and the button toggles the controller value/note-on velocity
- between value/velocity 127 and 0 with every operation

Please Note:

- The ASSIGN section also reflects/displays reception of the same MIDI commands that are selected for transmission
- The MIDI commands assigned to the ASSIGN controls can be transferred to and from stage via AES50 using the S16 stage box MIDI I/O

MIDI RX/TX > DAW REMOTE CONTROL SURFACE

Enables a specific form of bi-directional MIDI communication for remote controlling a computer DAW application using control elements of the X32 console. REMOTE can be used in 3 modes, Mackie Control, HUI and raw MIDI CC (raw) controllers (see Setup/remote)

MIDI CC (raw) selected and Remote is enabled+active, the group section controls will transmit/ receive the following messages:

MIDITX	Midi Ch	Controller	Comment	
Group 1-8 SELECT	1	Note 64-71	on (127) / off (0), push non-latching	Note that the exact button functions may vary from DAW to DAW
Group 1-8 SOLO	1	CC 32-39	on (127) / off (0), toggle latching	
Group 1-8 MUTE	1	CC 40-47	on (127) / off (0), toggle latching	
Sends On Fader	1	CC 48	on (127) / off (0), toggle latching	
Group DCA 1-8	1	Note 72	on (127) / off (0), push non-latching	
BUS 1-8	1	Note 73	on (127) / off (0), push non-latching	
BUS 9-16	1	Note 74	on (127) / off (0), push non-latching	
MTX 1-6	1	Note 75	on (127) / off (0), push non-latching	
GROUP Faders	1	CC0-7	0-127	

HUI selected and Remote is enabled+active, then the group fader section and buttons will emulate the HUI control surface protocol, i.e. for ProTools.

- SELECT/SOLO 1-8 buttons will select or solo the corresponding track in the DAW, in banks of 8 tracks
- Sends On Fader = enables touch-writing a fader automation on selected track, track automation mode in DAW must be ‘touch’, (latching)
- use the layer buttons to determine the function assigned to the MUTE 1-8 buttons, the LED displays indicate that function
  - Group DCA 1-8 = allows to move the bank selection of tracks in a DAW, (push non-latching)
  - BUS 1-8 = allows to set DAW tracks to ‘Record Ready’, (push non-latching)
  - BUS 9-16 = enables using the MUTE buttons for track mute in the DAW, (latching)
  - MTX 1-6 = enables using the MUTE buttons for transport controls in the DAW, (latching)

MACKIE CTRL selected and Remote is enabled+active, then the group fader section and buttons will emulate the Mackie Control Universal protocol

- SELECT/SOLO 1-8 buttons will select or solo the corresponding track in the DAW, in banks of 8 tracks
- Sends On Fader = enables touch-writing a fader automation on selected track, track automation mode in DAW must be ‘touch’ or ‘latch’, (latching)
- use the layer buttons to determine the function assigned to the MUTE 1-8 buttons, the LED displays indicate that function
  - Group DCA 1-8 = allows to move the bank selection of tracks in a DAW, (push non-latching)
  - BUS 1-8 = allows to set DAW tracks to ‘Record Ready’, (push non-latching)
  - BUS 9-16 = enables using the MUTE buttons for track mute in the DAW, (latching)
  - MTX 1-6 = enables using the MUTE buttons for transport controls in the DAW, (latching)

Appendix – OSC over MIDI Sysex commands

Additionaly to Behringer's document/note "X32 MIDI Implementation" 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 kbtyes in length. The space character *0x20* is used as separator between command and data, as shown below. Parameter data are converted from int or float to their string equivalent, respecting known X32 values where appropriate. Enums are sent as strings too.

Examples: (~ stands for the NULL character, \0; data within brackets are sent as 32 bits big endian values). OSC below only applies to commands sent over ethernet; Sysex commands are to be sent over MIDI only.

Setting channel 01 mute ON (muting the channel):
OSC: */ch/01/mix/on~~~,i~~[0]*
OSC: */ch/01/mix/on~~~,s~~OFF*
<OSCtext>: */ch/01/mix/on 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*
<OSCtext>: */ch/01/mix/on 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]*
<OSCtext>: */ch/01/eq/2/f 1020*
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]*
<OSCtext>: */ch/01/dyn/hold 100*
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~*
<OSCtext>: */config/userctrl/C/btn/5 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* and *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).

Appendix – Fader Floats Table – 1024 float values – [0.0, 1.0]

The data is presented as [fader index, float, float hex value, node value] quadruplets

1	0.0000	0x00000000	-oo	57	0.0547	0x3d60380e	-63.7	113	0.1095	0x3de0380e	-52.5
2	0.0010	0x3a802008	-89.5	58	0.0557	0x3d64390e	-63.3	114	0.1105	0x3de2388e	-52.3
3	0.0020	0x3b002008	-89.1	59	0.0567	0x3d683a0f	-62.8	115	0.1114	0x3de4390e	-52.2
4	0.0029	0x3b40300c	-88.6	60	0.0577	0x3d6c3b0f	-62.3	116	0.1124	0x3de6398e	-52
5	0.0039	0x3b802008	-88.1	61	0.0587	0x3d703c0f	-61.8	117	0.1134	0x3de83a0f	-51.9
6	0.0049	0x3ba0280a	-87.7	62	0.0596	0x3d743d0f	-61.4	118	0.1144	0x3dea3a8f	-51.7
7	0.0059	0x3bc0300c	-87.2	63	0.0606	0x3d783e10	-60.9	119	0.1153	0x3dec3b0f	-51.5
8	0.0068	0x3be0380e	-86.7	64	0.0616	0x3d7c3f10	-60.4	120	0.1163	0x3dee3b8f	-51.4
9	0.0078	0x3c002008	-86.2	65	0.0626	0x3d802008	-60	121	0.1173	0x3df03c0f	-51.2
10	0.0088	0x3c102409	-85.8	66	0.0635	0x3d822088	-59.8	122	0.1183	0x3df23c8f	-51.1
11	0.0098	0x3c20280a	-85.3	67	0.0645	0x3d842108	-59.7	123	0.1193	0x3df43d0f	-50.9
12	0.0108	0x3c302c0b	-84.8	68	0.0655	0x3d862188	-59.5	124	0.1202	0x3df63d8f	-50.8
13	0.0117	0x3c40300c	-84.4	69	0.0665	0x3d882209	-59.4	125	0.1212	0x3df83e10	-50.6
14	0.0127	0x3c50340d	-83.9	70	0.0674	0x3d8a2289	-59.2	126	0.1222	0x3dfa3e90	-50.4
15	0.0137	0x3c60380e	-83.4	71	0.0684	0x3d8c2309	-59.1	127	0.1232	0x3dfc3f10	-50.3
16	0.0147	0x3c703c0f	-83	72	0.0694	0x3d8e2389	-58.9	128	0.1241	0x3dfe3f90	-50.1
17	0.0156	0x3c802008	-82.5	73	0.0704	0x3d902409	-58.7	129	0.1251	0x3e002008	-50
18	0.0166	0x3c882209	-82	74	0.0714	0x3d922489	-58.6	130	0.1261	0x3e012048	-49.8
19	0.0176	0x3c902409	-81.6	75	0.0723	0x3d942509	-58.4	131	0.1271	0x3e022088	-49.7
20	0.0186	0x3c98260a	-81.1	76	0.0733	0x3d962589	-58.3	132	0.1281	0x3e0320c8	-49.5
21	0.0196	0x3ca0280a	-80.6	77	0.0743	0x3d98260a	-58.1	133	0.1290	0x3e042108	-49.4
22	0.0205	0x3ca82a0b	-80.1	78	0.0753	0x3d9a268a	-58	134	0.1300	0x3e052148	-49.2
23	0.0215	0x3cb02c0b	-79.7	79	0.0762	0x3d9c270a	-57.8	135	0.1310	0x3e062188	-49
24	0.0225	0x3cb82e0c	-79.2	80	0.0772	0x3d9e278a	-57.6	136	0.1320	0x3e0721c8	-48.9
25	0.0235	0x3cc0300c	-78.7	81	0.0782	0x3da0280a	-57.5	137	0.1329	0x3e082209	-48.7
26	0.0244	0x3cc8320d	-78.3	82	0.0792	0x3da2288a	-57.3	138	0.1339	0x3e092249	-48.6
27	0.0254	0x3cd0340d	-77.8	83	0.0802	0x3da4290a	-57.2	139	0.1349	0x3e0a2289	-48.4
28	0.0264	0x3cd8360e	-77.3	84	0.0811	0x3da6298a	-57	140	0.1359	0x3e0b22c9	-48.3
29	0.0274	0x3ce0380e	-76.9	85	0.0821	0x3da82a0b	-56.9	141	0.1369	0x3e0c2309	-48.1
30	0.0283	0x3ce83a0f	-76.4	86	0.0831	0x3daa2a8b	-56.7	142	0.1378	0x3e0d2349	-47.9
31	0.0293	0x3cf03c0f	-75.9	87	0.0841	0x3dac2b0b	-56.5	143	0.1388	0x3e0e2389	-47.8
32	0.0303	0x3cf83e10	-75.5	88	0.0850	0x3dae2b8b	-56.4	144	0.1398	0x3e0f23c9	-47.6
33	0.0313	0x3d002008	-75	89	0.0860	0x3db02c0b	-56.2	145	0.1408	0x3e102409	-47.5
34	0.0323	0x3d042108	-74.5	90	0.0870	0x3db22c8b	-56.1	146	0.1417	0x3e112449	-47.3
35	0.0332	0x3d082209	-74	91	0.0880	0x3db42d0b	-55.9	147	0.1427	0x3e122489	-47.2
36	0.0342	0x3d0c2309	-73.6	92	0.0890	0x3db62d8b	-55.8	148	0.1437	0x3e1324c9	-47
37	0.0352	0x3d102409	-73.1	93	0.0899	0x3db82e0c	-55.6	149	0.1447	0x3e142509	-46.9
38	0.0362	0x3d142509	-72.6	94	0.0909	0x3dba2e8c	-55.5	150	0.1457	0x3e152549	-46.7
39	0.0371	0x3d18260a	-72.2	95	0.0919	0x3dbc2f0c	-55.3	151	0.1466	0x3e162589	-46.5
40	0.0381	0x3d1c270a	-71.7	96	0.0929	0x3dbe2f8c	-55.1	152	0.1476	0x3e1725c9	-46.4
41	0.0391	0x3d20280a	-71.2	97	0.0938	0x3dc0300c	-55	153	0.1486	0x3e18260a	-46.2
42	0.0401	0x3d24290a	-70.8	98	0.0948	0x3dc2308c	-54.8	154	0.1496	0x3e19264a	-46.1
43	0.0411	0x3d282a0b	-70.3	99	0.0958	0x3dc4310c	-54.7	155	0.1505	0x3e1a268a	-45.9
44	0.0420	0x3d2c2b0b	-69.8	100	0.0968	0x3dc6318c	-54.5	156	0.1515	0x3e1b26ca	-45.8
45	0.0430	0x3d302c0b	-69.4	101	0.0978	0x3dc8320d	-54.4	157	0.1525	0x3e1c270a	-45.6
46	0.0440	0x3d342d0b	-68.9	102	0.0987	0x3dca328d	-54.2	158	0.1535	0x3e1d274a	-45.4
47	0.0450	0x3d382e0c	-68.4	103	0.0997	0x3dcc330d	-54	159	0.1544	0x3e1e278a	-45.3
48	0.0459	0x3d3c2f0c	-67.9	104	0.1007	0x3dce338d	-53.9	160	0.1554	0x3e1f27ca	-45.1
49	0.0469	0x3d40300c	-67.5	105	0.1017	0x3dd0340d	-53.7	161	0.1564	0x3e20280a	-45
50	0.0479	0x3d44310c	-67	106	0.1026	0x3dd2348d	-53.6	162	0.1574	0x3e21284a	-44.8
51	0.0489	0x3d48320d	-66.5	107	0.1036	0x3dd4350d	-53.4	163	0.1584	0x3e22288a	-44.7
52	0.0499	0x3d4c330d	-66.1	108	0.1046	0x3dd6358d	-53.3	164	0.1593	0x3e2328ca	-44.5
53	0.0508	0x3d50340d	-65.6	109	0.1056	0x3dd8360e	-53.1	165	0.1603	0x3e24290a	-44.3
54	0.0518	0x3d54350d	-65.1	110	0.1065	0x3dda368e	-53	166	0.1613	0x3e25294a	-44.2
55	0.0528	0x3d58360e	-64.7	111	0.1075	0x3ddc370e	-52.8	167	0.1623	0x3e26298a	-44
56	0.0538	0x3d5c370e	-64.2	112	0.1085	0x3dde378e	-52.6	168	0.1632	0x3e2729ca	-43.9

169	0.1642	0x3e282a0b	-43.7	230	0.2239	0x3e65394e	-34.2	291	0.2835	0x3e912449	-27.3
170	0.1652	0x3e292a4b	-43.6	231	0.2248	0x3e66398e	-34	292	0.2845	0x3e91a469	-27.2
171	0.1662	0x3e2a2a8b	-43.4	232	0.2258	0x3e6739ce	-33.9	293	0.2854	0x3e922489	-27.2
172	0.1672	0x3e2b2acb	-43.3	233	0.2268	0x3e683a0f	-33.7	294	0.2864	0x3e92a4a9	-27.1
173	0.1681	0x3e2c2b0b	-43.1	234	0.2278	0x3e693a4f	-33.6	295	0.2874	0x3e9324c9	-27
174	0.1691	0x3e2d2b4b	-42.9	235	0.2287	0x3e6a3a8f	-33.4	296	0.2884	0x3e93a4e9	-26.9
175	0.1701	0x3e2e2b8b	-42.8	236	0.2297	0x3e6b3acf	-33.2	297	0.2893	0x3e942509	-26.9
176	0.1711	0x3e2f2bcb	-42.6	237	0.2307	0x3e6c3b0f	-33.1	298	0.2903	0x3e94a529	-26.8
177	0.1720	0x3e302c0b	-42.5	238	0.2317	0x3e6d3b4f	-32.9	299	0.2913	0x3e952549	-26.7
178	0.1730	0x3e312c4b	-42.3	239	0.2326	0x3e6e3b8f	-32.8	300	0.2923	0x3e95a569	-26.6
179	0.1740	0x3e322c8b	-42.2	240	0.2336	0x3e6f3bcf	-32.6	301	0.2933	0x3e962589	-26.5
180	0.1750	0x3e332ccb	-42	241	0.2346	0x3e703c0f	-32.5	302	0.2942	0x3e96a5a9	-26.5
181	0.1760	0x3e342d0b	-41.8	242	0.2356	0x3e713c4f	-32.3	303	0.2952	0x3e9725c9	-26.4
182	0.1769	0x3e352d4b	-41.7	243	0.2366	0x3e723c8f	-32.2	304	0.2962	0x3e97a5e9	-26.3
183	0.1779	0x3e362d8b	-41.5	244	0.2375	0x3e733ccf	-32	305	0.2972	0x3e98260a	-26.2
184	0.1789	0x3e372dcb	-41.4	245	0.2385	0x3e743d0f	-31.8	306	0.2981	0x3e98a62a	-26.1
185	0.1799	0x3e382e0c	-41.2	246	0.2395	0x3e753d4f	-31.7	307	0.2991	0x3e99264a	-26.1
186	0.1808	0x3e392e4c	-41.1	247	0.2405	0x3e763d8f	-31.5	308	0.3001	0x3e99a66a	-26
187	0.1818	0x3e3a2e8c	-40.9	248	0.2414	0x3e773dcf	-31.4	309	0.3011	0x3e9a268a	-25.9
188	0.1828	0x3e3b2ecc	-40.8	249	0.2424	0x3e783e10	-31.2	310	0.3021	0x3e9aa6aa	-25.8
189	0.1838	0x3e3c2f0c	-40.6	250	0.2434	0x3e793e50	-31.1	311	0.3030	0x3e9b26ca	-25.8
190	0.1848	0x3e3d2f4c	-40.4	251	0.2444	0x3e7a3e90	-30.9	312	0.3040	0x3e9ba6ea	-25.7
191	0.1857	0x3e3e2f8c	-40.3	252	0.2454	0x3e7b3ed0	-30.7	313	0.3050	0x3e9c270a	-25.6
192	0.1867	0x3e3f2fcc	-40.1	253	0.2463	0x3e7c3f10	-30.6	314	0.3060	0x3e9ca72a	-25.5
193	0.1877	0x3e40300c	-40	254	0.2473	0x3e7d3f50	-30.4	315	0.3069	0x3e9d274a	-25.4
194	0.1887	0x3e41304c	-39.8	255	0.2483	0x3e7e3f90	-30.3	316	0.3079	0x3e9da76a	-25.4
195	0.1896	0x3e42308c	-39.7	256	0.2493	0x3e7f3fd0	-30.1	317	0.3089	0x3e9e278a	-25.3
196	0.1906	0x3e4330cc	-39.5	257	0.2502	0x3e802008	-30	318	0.3099	0x3e9ea7aa	-25.2
197	0.1916	0x3e44310c	-39.3	258	0.2512	0x3e80a028	-29.9	319	0.3109	0x3e9f27ca	-25.1
198	0.1926	0x3e45314c	-39.2	259	0.2522	0x3e812048	-29.8	320	0.3118	0x3e9fa7ea	-25.1
199	0.1935	0x3e46318c	-39	260	0.2532	0x3e81a068	-29.7	321	0.3128	0x3ea0280a	-25
200	0.1945	0x3e4731cc	-38.9	261	0.2542	0x3e822088	-29.7	322	0.3138	0x3ea0a82a	-24.9
201	0.1955	0x3e48320d	-38.7	262	0.2551	0x3e82a0a8	-29.6	323	0.3148	0x3ea1284a	-24.8
202	0.1965	0x3e49324d	-38.6	263	0.2561	0x3e8320c8	-29.5	324	0.3157	0x3ea1a86a	-24.7
203	0.1975	0x3e4a328d	-38.4	264	0.2571	0x3e83a0e8	-29.4	325	0.3167	0x3ea2288a	-24.7
204	0.1984	0x3e4b32cd	-38.3	265	0.2581	0x3e842108	-29.4	326	0.3177	0x3ea2a8aa	-24.6
205	0.1994	0x3e4c330d	-38.1	266	0.2590	0x3e84a128	-29.3	327	0.3187	0x3ea328ca	-24.5
206	0.2004	0x3e4d334d	-37.9	267	0.2600	0x3e852148	-29.2	328	0.3196	0x3ea3a8ea	-24.4
207	0.2014	0x3e4e338d	-37.8	268	0.2610	0x3e85a168	-29.1	329	0.3206	0x3ea4290a	-24.3
208	0.2023	0x3e4f33cd	-37.6	269	0.2620	0x3e862188	-29	330	0.3216	0x3ea4a92a	-24.3
209	0.2033	0x3e50340d	-37.5	270	0.2630	0x3e86a1a8	-29	331	0.3226	0x3ea5294a	-24.2
210	0.2043	0x3e51344d	-37.3	271	0.2639	0x3e8721c8	-28.9	332	0.3236	0x3ea5a96a	-24.1
211	0.2053	0x3e52348d	-37.2	272	0.2649	0x3e87a1e8	-28.8	333	0.3245	0x3ea6298a	-24
212	0.2063	0x3e5334cd	-37	273	0.2659	0x3e882209	-28.7	334	0.3255	0x3ea6a9aa	-24
213	0.2072	0x3e54350d	-36.8	274	0.2669	0x3e88a229	-28.7	335	0.3265	0x3ea729ca	-23.9
214	0.2082	0x3e55354d	-36.7	275	0.2678	0x3e892249	-28.6	336	0.3275	0x3ea7a9ea	-23.8
215	0.2092	0x3e56358d	-36.5	276	0.2688	0x3e89a269	-28.5	337	0.3284	0x3ea82a0b	-23.7
216	0.2102	0x3e5735cd	-36.4	277	0.2698	0x3e8a2289	-28.4	338	0.3294	0x3ea8aa2b	-23.6
217	0.2111	0x3e58360e	-36.2	278	0.2708	0x3e8aa2a9	-28.3	339	0.3304	0x3ea92a4b	-23.6
218	0.2121	0x3e59364e	-36.1	279	0.2717	0x3e8b22c9	-28.3	340	0.3314	0x3ea9aa6b	-23.5
219	0.2131	0x3e5a368e	-35.9	280	0.2727	0x3e8ba2e9	-28.2	341	0.3324	0x3eaa2a8b	-23.4
220	0.2141	0x3e5b36ce	-35.7	281	0.2737	0x3e8c2309	-28.1	342	0.3333	0x3eaaaaab	-23.3
221	0.2151	0x3e5c370e	-35.6	282	0.2747	0x3e8ca329	-28	343	0.3343	0x3eab2acb	-23.2
222	0.2160	0x3e5d374e	-35.4	283	0.2757	0x3e8d2349	-27.9	344	0.3353	0x3eabaaeb	-23.2
223	0.2170	0x3e5e378e	-35.3	284	0.2766	0x3e8da369	-27.9	345	0.3363	0x3eac2b0b	-23.1
224	0.2180	0x3e5f37ce	-35.1	285	0.2776	0x3e8e2389	-27.8	346	0.3372	0x3eacab2b	-23
225	0.2190	0x3e60380e	-35	286	0.2786	0x3e8ea3a9	-27.7	347	0.3382	0x3ead2b4b	-22.9
226	0.2199	0x3e61384e	-34.8	287	0.2796	0x3e8f23c9	-27.6	348	0.3392	0x3eadab6b	-22.9
227	0.2209	0x3e62388e	-34.7	288	0.2805	0x3e8fa3e9	-27.6	349	0.3402	0x3eae2b8b	-22.8
228	0.2219	0x3e6338ce	-34.5	289	0.2815	0x3e902409	-27.5	350	0.3412	0x3eaeabab	-22.7
229	0.2229	0x3e64390e	-34.3	290	0.2825	0x3e90a429	-27.4	351	0.3421	0x3eaf2bcb	-22.6

352	0.3431	0x3eafabeb	-22.6	413	0.4027	0x3ece338d	-17.8	474	0.4624	0x3eecbb2f	-13
353	0.3441	0x3eb02c0b	-22.5	414	0.4037	0x3eceb3ad	-17.7	475	0.4633	0x3eed3b4f	-12.9
354	0.3451	0x3eb0ac2b	-22.4	415	0.4047	0x3ecf33cd	-17.6	476	0.4643	0x3eedbb6f	-12.9
355	0.3460	0x3eb12c4b	-22.3	416	0.4057	0x3ecfb3ed	-17.5	477	0.4653	0x3eee3b8f	-12.8
356	0.3470	0x3eb1ac6b	-22.2	417	0.4066	0x3ed0340d	-17.5	478	0.4663	0x3eeebbaf	-12.7
357	0.3480	0x3eb22c8b	-22.2	418	0.4076	0x3ed0b42d	-17.4	479	0.4673	0x3eef3bcf	-12.6
358	0.3490	0x3eb2acab	-22.1	419	0.4086	0x3ed1344d	-17.3	480	0.4682	0x3eefbbef	-12.5
359	0.3500	0x3eb32ccb	-22	420	0.4096	0x3ed1b46d	-17.2	481	0.4692	0x3ef03c0f	-12.5
360	0.3509	0x3eb3aceb	-21.9	421	0.4106	0x3ed2348d	-17.2	482	0.4702	0x3ef0bc2f	-12.4
361	0.3519	0x3eb42d0b	-21.8	422	0.4115	0x3ed2b4ad	-17.1	483	0.4712	0x3ef13c4f	-12.3
362	0.3529	0x3eb4ad2b	-21.8	423	0.4125	0x3ed334cd	-17	484	0.4721	0x3ef1bc6f	-12.2
363	0.3539	0x3eb52d4b	-21.7	424	0.4135	0x3ed3b4ed	-16.9	485	0.4731	0x3ef23c8f	-12.2
364	0.3548	0x3eb5ad6b	-21.6	425	0.4145	0x3ed4350d	-16.8	486	0.4741	0x3ef2bcaf	-12.1
365	0.3558	0x3eb62d8b	-21.5	426	0.4154	0x3ed4b52d	-16.8	487	0.4751	0x3ef33ccf	-12
366	0.3568	0x3eb6adab	-21.5	427	0.4164	0x3ed5354d	-16.7	488	0.4761	0x3ef3bcef	-11.9
367	0.3578	0x3eb72dcb	-21.4	428	0.4174	0x3ed5b56d	-16.6	489	0.4770	0x3ef43d0f	-11.8
368	0.3587	0x3eb7adeb	-21.3	429	0.4184	0x3ed6358d	-16.5	490	0.4780	0x3ef4bd2f	-11.8
369	0.3597	0x3eb82e0c	-21.2	430	0.4194	0x3ed6b5ad	-16.5	491	0.4790	0x3ef53d4f	-11.7
370	0.3607	0x3eb8ae2c	-21.1	431	0.4203	0x3ed735cd	-16.4	492	0.4800	0x3ef5bd6f	-11.6
371	0.3617	0x3eb92e4c	-21.1	432	0.4213	0x3ed7b5ed	-16.3	493	0.4809	0x3ef63d8f	-11.5
372	0.3627	0x3eb9ae6c	-21	433	0.4223	0x3ed8360e	-16.2	494	0.4819	0x3ef6bdaf	-11.4
373	0.3636	0x3eba2e8c	-20.9	434	0.4233	0x3ed8b62e	-16.1	495	0.4829	0x3ef73dcf	-11.4
374	0.3646	0x3ebaaeac	-20.8	435	0.4242	0x3ed9364e	-16.1	496	0.4839	0x3ef7bdef	-11.3
375	0.3656	0x3ebb2ecc	-20.8	436	0.4252	0x3ed9b66e	-16	497	0.4848	0x3ef83e10	-11.2
376	0.3666	0x3ebbaeec	-20.7	437	0.4262	0x3eda368e	-15.9	498	0.4858	0x3ef8be30	-11.1
377	0.3675	0x3ebc2f0c	-20.6	438	0.4272	0x3edab6ae	-15.8	499	0.4868	0x3ef93e50	-11.1
378	0.3685	0x3ebcaf2c	-20.5	439	0.4282	0x3edb36ce	-15.7	500	0.4878	0x3ef9be70	-11
379	0.3695	0x3ebd2f4c	-20.4	440	0.4291	0x3edbb6ee	-15.7	501	0.4888	0x3efa3e90	-10.9
380	0.3705	0x3ebdaf6c	-20.4	441	0.4301	0x3edc370e	-15.6	502	0.4897	0x3efabeb0	-10.8
381	0.3715	0x3ebe2f8c	-20.3	442	0.4311	0x3edcb72e	-15.5	503	0.4907	0x3efb3ed0	-10.7
382	0.3724	0x3ebeafac	-20.2	443	0.4321	0x3edd374e	-15.4	504	0.4917	0x3efbbef0	-10.7
383	0.3734	0x3ebf2fcc	-20.1	444	0.4330	0x3eddb76e	-15.4	505	0.4927	0x3efc3f10	-10.6
384	0.3744	0x3ebfafec	-20	445	0.4340	0x3ede378e	-15.3	506	0.4936	0x3efcbf30	-10.5
385	0.3754	0x3ec0300c	-20	446	0.4350	0x3edeb7ae	-15.2	507	0.4946	0x3efd3f50	-10.4
386	0.3763	0x3ec0b02c	-19.9	447	0.4360	0x3edf37ce	-15.1	508	0.4956	0x3efdbf70	-10.4
387	0.3773	0x3ec1304c	-19.8	448	0.4370	0x3edfb7ee	-15	509	0.4966	0x3efe3f90	-10.3
388	0.3783	0x3ec1b06c	-19.7	449	0.4379	0x3ee0380e	-15	510	0.4976	0x3efebfb0	-10.2
389	0.3793	0x3ec2308c	-19.7	450	0.4389	0x3ee0b82e	-14.9	511	0.4985	0x3eff3fd0	-10.1
390	0.3803	0x3ec2b0ac	-19.6	451	0.4399	0x3ee1384e	-14.8	512	0.4995	0x3effbff0	-10
391	0.3812	0x3ec330cc	-19.5	452	0.4409	0x3ee1b86e	-14.7	513	0.5005	0x3f002008	-10
392	0.3822	0x3ec3b0ec	-19.4	453	0.4418	0x3ee2388e	-14.7	514	0.5015	0x3f006018	-9.9
393	0.3832	0x3ec4310c	-19.3	454	0.4428	0x3ee2b8ae	-14.6	515	0.5024	0x3f00a028	-9.9
394	0.3842	0x3ec4b12c	-19.3	455	0.4438	0x3ee338ce	-14.5	516	0.5034	0x3f00e038	-9.9
395	0.3851	0x3ec5314c	-19.2	456	0.4448	0x3ee3b8ee	-14.4	517	0.5044	0x3f012048	-9.8
396	0.3861	0x3ec5b16c	-19.1	457	0.4457	0x3ee4390e	-14.3	518	0.5054	0x3f016058	-9.8
397	0.3871	0x3ec6318c	-19	458	0.4467	0x3ee4b92e	-14.3	519	0.5064	0x3f01a068	-9.7
398	0.3881	0x3ec6b1ac	-19	459	0.4477	0x3ee5394e	-14.2	520	0.5073	0x3f01e078	-9.7
399	0.3891	0x3ec731cc	-18.9	460	0.4487	0x3ee5b96e	-14.1	521	0.5083	0x3f022088	-9.7
400	0.3900	0x3ec7b1ec	-18.8	461	0.4497	0x3ee6398e	-14	522	0.5093	0x3f026098	-9.6
401	0.3910	0x3ec8320d	-18.7	462	0.4506	0x3ee6b9ae	-13.9	523	0.5103	0x3f02a0a8	-9.6
402	0.3920	0x3ec8b22d	-18.6	463	0.4516	0x3ee739ce	-13.9	524	0.5112	0x3f02e0b8	-9.6
403	0.3930	0x3ec9324d	-18.6	464	0.4526	0x3ee7b9ee	-13.8	525	0.5122	0x3f0320c8	-9.5
404	0.3939	0x3ec9b26d	-18.5	465	0.4536	0x3ee83a0f	-13.7	526	0.5132	0x3f0360d8	-9.5
405	0.3949	0x3eca328d	-18.4	466	0.4545	0x3ee8ba2f	-13.6	527	0.5142	0x3f03a0e8	-9.4
406	0.3959	0x3ecab2ad	-18.3	467	0.4555	0x3ee93a4f	-13.6	528	0.5152	0x3f03e0f8	-9.4
407	0.3969	0x3ecb32cd	-18.3	468	0.4565	0x3ee9ba6f	-13.5	529	0.5161	0x3f042108	-9.4
408	0.3978	0x3ecbb2ed	-18.2	469	0.4575	0x3eea3a8f	-13.4	530	0.5171	0x3f046118	-9.3
409	0.3988	0x3ecc330d	-18.1	470	0.4585	0x3eeabaaf	-13.3	531	0.5181	0x3f04a128	-9.3
410	0.3998	0x3eccb32d	-18	471	0.4594	0x3eeb3acf	-13.2	532	0.5191	0x3f04e138	-9.2
411	0.4008	0x3ecd334d	-17.9	472	0.4604	0x3eebbaef	-13.2	533	0.5200	0x3f052148	-9.2
412	0.4018	0x3ecdb36d	-17.9	473	0.4614	0x3eec3b0f	-13.1	534	0.5210	0x3f056158	-9.2

535	0.5220	0x3f05a168	-9.1	596	0.5816	0x3f14e539	-6.7	657	0.6413	0x3f24290a	-4.3
536	0.5230	0x3f05e178	-9.1	597	0.5826	0x3f152549	-6.7	658	0.6422	0x3f24691a	-4.3
537	0.5239	0x3f062188	-9	598	0.5836	0x3f156559	-6.7	659	0.6432	0x3f24a92a	-4.3
538	0.5249	0x3f066198	-9	599	0.5846	0x3f15a569	-6.6	660	0.6442	0x3f24e93a	-4.2
539	0.5259	0x3f06a1a8	-9	600	0.5855	0x3f15e579	-6.6	661	0.6452	0x3f25294a	-4.2
540	0.5269	0x3f06e1b8	-8.9	601	0.5865	0x3f162589	-6.5	662	0.6461	0x3f25695a	-4.2
541	0.5279	0x3f0721c8	-8.9	602	0.5875	0x3f166599	-6.5	663	0.6471	0x3f25a96a	-4.1
542	0.5288	0x3f0761d8	-8.8	603	0.5885	0x3f16a5a9	-6.5	664	0.6481	0x3f25e97a	-4.1
543	0.5298	0x3f07a1e8	-8.8	604	0.5894	0x3f16e5b9	-6.4	665	0.6491	0x3f26298a	-4
544	0.5308	0x3f07e1f8	-8.8	605	0.5904	0x3f1725c9	-6.4	666	0.6500	0x3f26699a	-4
545	0.5318	0x3f082209	-8.7	606	0.5914	0x3f1765d9	-6.3	667	0.6510	0x3f26a9aa	-4
546	0.5327	0x3f086219	-8.7	607	0.5924	0x3f17a5e9	-6.3	668	0.6520	0x3f26e9ba	-3.9
547	0.5337	0x3f08a229	-8.7	608	0.5934	0x3f17e5f9	-6.3	669	0.6530	0x3f2729ca	-3.9
548	0.5347	0x3f08e239	-8.7	609	0.5943	0x3f18260a	-6.2	670	0.6540	0x3f2769da	-3.8
549	0.5357	0x3f092249	-8.6	610	0.5953	0x3f18661a	-6.2	671	0.6549	0x3f27a9ea	-3.8
550	0.5367	0x3f096259	-8.5	611	0.5963	0x3f18a62a	-6.1	672	0.6559	0x3f27e9fa	-3.8
551	0.5376	0x3f09a269	-8.5	612	0.5973	0x3f18e63a	-6.1	673	0.6569	0x3f282a0b	-3.7
552	0.5386	0x3f09e279	-8.5	613	0.5982	0x3f19264a	-6.1	674	0.6579	0x3f286a1b	-3.7
553	0.5396	0x3f0a2289	-8.4	614	0.5992	0x3f19665a	-6	675	0.6588	0x3f28aa2b	-3.6
554	0.5406	0x3f0a6299	-8.4	615	0.6002	0x3f19a66a	-6	676	0.6598	0x3f28ea3b	-3.6
555	0.5415	0x3f0aa2a9	-8.3	616	0.6012	0x3f19e67a	-6	677	0.6608	0x3f292a4b	-3.6
556	0.5425	0x3f0ae2b9	-8.3	617	0.6022	0x3f1a268a	-5.9	678	0.6618	0x3f296a5b	-3.5
557	0.5435	0x3f0b22c9	-8.3	618	0.6031	0x3f1a669a	-5.9	679	0.6628	0x3f29aa6b	-3.5
558	0.5445	0x3f0b62d9	-8.2	619	0.6041	0x3f1aa6aa	-5.8	680	0.6637	0x3f29ea7b	-3.5
559	0.5455	0x3f0ba2e9	-8.2	620	0.6051	0x3f1ae6ba	-5.8	681	0.6647	0x3f2a2a8b	-3.4
560	0.5464	0x3f0be2f9	-8.1	621	0.6061	0x3f1b26ca	-5.8	682	0.6657	0x3f2a6a9b	-3.4
561	0.5474	0x3f0c2309	-8.1	622	0.6070	0x3f1b66da	-5.7	683	0.6667	0x3f2aaaab	-3.3
562	0.5484	0x3f0c6319	-8.1	623	0.6080	0x3f1ba6ea	-5.7	684	0.6676	0x3f2aeabb	-3.3
563	0.5494	0x3f0ca329	-8	624	0.6090	0x3f1be6fa	-5.6	685	0.6686	0x3f2b2acb	-3.3
564	0.5503	0x3f0ce339	-8	625	0.6100	0x3f1c270a	-5.6	686	0.6696	0x3f2b6adb	-3.2
565	0.5513	0x3f0d2349	-7.9	626	0.6109	0x3f1c671a	-5.6	687	0.6706	0x3f2baaeb	-3.2
566	0.5523	0x3f0d6359	-7.9	627	0.6119	0x3f1ca72a	-5.5	688	0.6716	0x3f2beafb	-3.1
567	0.5533	0x3f0da369	-7.9	628	0.6129	0x3f1ce73a	-5.5	689	0.6725	0x3f2c2b0b	-3.1
568	0.5543	0x3f0de379	-7.8	629	0.6139	0x3f1d274a	-5.4	690	0.6735	0x3f2c6b1b	-3.1
569	0.5552	0x3f0e2389	-7.8	630	0.6149	0x3f1d675a	-5.4	691	0.6745	0x3f2cab2b	-3
570	0.5562	0x3f0e6399	-7.8	631	0.6158	0x3f1da76a	-5.4	692	0.6755	0x3f2ceb3b	-3
571	0.5572	0x3f0ea3a9	-7.7	632	0.6168	0x3f1de77a	-5.3	693	0.6764	0x3f2d2b4b	-2.9
572	0.5582	0x3f0ee3b9	-7.7	633	0.6178	0x3f1e278a	-5.3	694	0.6774	0x3f2d6b5b	-2.9
573	0.5591	0x3f0f23c9	-7.6	634	0.6188	0x3f1e679a	-5.2	695	0.6784	0x3f2dab6b	-2.9
574	0.5601	0x3f0f63d9	-7.6	635	0.6197	0x3f1ea7aa	-5.2	696	0.6794	0x3f2deb7b	-2.8
575	0.5611	0x3f0fa3e9	-7.6	636	0.6207	0x3f1ee7ba	-5.2	697	0.6804	0x3f2e2b8b	-2.8
576	0.5621	0x3f0fe3f9	-7.5	637	0.6217	0x3f1f27ca	-5.1	698	0.6813	0x3f2e6b9b	-2.7
577	0.5630	0x3f102409	-7.5	638	0.6227	0x3f1f67da	-5.1	699	0.6823	0x3f2eabab	-2.7
578	0.5640	0x3f106419	-7.4	639	0.6237	0x3f1fa7ea	-5.1	700	0.6833	0x3f2eebbb	-2.7
579	0.5650	0x3f10a429	-7.4	640	0.6246	0x3f1fe7fa	-5	701	0.6843	0x3f2f2bcb	-2.6
580	0.5660	0x3f10e439	-7.4	641	0.6256	0x3f20280a	-5	702	0.6852	0x3f2f6bdb	-2.6
581	0.5670	0x3f112449	-7.3	642	0.6266	0x3f20681a	-4.9	703	0.6862	0x3f2fabeb	-2.6
582	0.5679	0x3f116459	-7.3	643	0.6276	0x3f20a82a	-4.9	704	0.6872	0x3f2febfb	-2.5
583	0.5689	0x3f11a469	-7.2	644	0.6285	0x3f20e83a	-4.9	705	0.6882	0x3f302c0b	-2.5
584	0.5699	0x3f11e479	-7.2	645	0.6295	0x3f21284a	-4.8	706	0.6891	0x3f306c1b	-2.4
585	0.5709	0x3f122489	-7.2	646	0.6305	0x3f21685a	-4.8	707	0.6901	0x3f30ac2b	-2.4
586	0.5718	0x3f126499	-7.1	647	0.6315	0x3f21a86a	-4.7	708	0.6911	0x3f30ec3b	-2.4
587	0.5728	0x3f12a4a9	-7.1	648	0.6325	0x3f21e87a	-4.7	709	0.6921	0x3f312c4b	-2.3
588	0.5738	0x3f12e4b9	-7	649	0.6334	0x3f22288a	-4.7	710	0.6931	0x3f316c5b	-2.3
589	0.5748	0x3f1324c9	-7	650	0.6344	0x3f22689a	-4.6	711	0.6940	0x3f31ac6b	-2.2
590	0.5758	0x3f1364d9	-7	651	0.6354	0x3f22a8aa	-4.6	712	0.6950	0x3f31ec7b	-2.2
591	0.5767	0x3f13a4e9	-6.9	652	0.6364	0x3f22e8ba	-4.5	713	0.6960	0x3f322c8b	-2.2
592	0.5777	0x3f13e4f9	-6.9	653	0.6373	0x3f2328ca	-4.5	714	0.6970	0x3f326c9b	-2.1
593	0.5787	0x3f142509	-6.9	654	0.6383	0x3f2368da	-4.5	715	0.6979	0x3f32acab	-2.1
594	0.5797	0x3f146519	-6.8	655	0.6393	0x3f23a8ea	-4.4	716	0.6989	0x3f32ecbb	-2
595	0.5806	0x3f14a529	-6.8	656	0.6403	0x3f23e8fa	-4.4	717	0.6999	0x3f332ccb	-2

718	0.7009	0x3f336cdb	-2	779	0.7605	0x3f42b0ac	0.4	840	0.8201	0x3f51f47d	2.8
719	0.7019	0x3f33aceb	-1.9	780	0.7615	0x3f42f0bc	0.5	841	0.8211	0x3f52348d	2.8
720	0.7028	0x3f33ecfb	-1.9	781	0.7625	0x3f4330cc	0.5	842	0.8221	0x3f52749d	2.9
721	0.7038	0x3f342d0b	-1.8	782	0.7634	0x3f4370dc	0.5	843	0.8231	0x3f52b4ad	2.9
722	0.7048	0x3f346d1b	-1.8	783	0.7644	0x3f43b0ec	0.6	844	0.8240	0x3f52f4bd	3
723	0.7058	0x3f34ad2b	-1.8	784	0.7654	0x3f43f0fc	0.6	845	0.8250	0x3f5334cd	3
724	0.7067	0x3f34ed3b	-1.7	785	0.7664	0x3f44310c	0.7	846	0.8260	0x3f5374dd	3
725	0.7077	0x3f352d4b	-1.7	786	0.7674	0x3f44711c	0.7	847	0.8270	0x3f53b4ed	3.1
726	0.7087	0x3f356d5b	-1.7	787	0.7683	0x3f44b12c	0.7	848	0.8280	0x3f53f4fd	3.1
727	0.7097	0x3f35ad6b	-1.6	788	0.7693	0x3f44f13c	0.8	849	0.8289	0x3f54350d	3.2
728	0.7107	0x3f35ed7b	-1.6	789	0.7703	0x3f45314c	0.8	850	0.8299	0x3f54751d	3.2
729	0.7116	0x3f362d8b	-1.5	790	0.7713	0x3f45715c	0.9	851	0.8309	0x3f54b52d	3.2
730	0.7126	0x3f366d9b	-1.5	791	0.7722	0x3f45b16c	0.9	852	0.8319	0x3f54f53d	3.3
731	0.7136	0x3f36adab	-1.5	792	0.7732	0x3f45f17c	0.9	853	0.8328	0x3f55354d	3.3
732	0.7146	0x3f36edbb	-1.4	793	0.7742	0x3f46318c	1	854	0.8338	0x3f55755d	3.4
733	0.7155	0x3f372dcb	-1.4	794	0.7752	0x3f46719c	1	855	0.8348	0x3f55b56d	3.4
734	0.7165	0x3f376ddb	-1.3	795	0.7761	0x3f46b1ac	1	856	0.8358	0x3f55f57d	3.4
735	0.7175	0x3f37adeb	-1.3	796	0.7771	0x3f46f1bc	1.1	857	0.8368	0x3f56358d	3.5
736	0.7185	0x3f37edfb	-1.3	797	0.7781	0x3f4731cc	1.1	858	0.8377	0x3f56759d	3.5
737	0.7195	0x3f382e0c	-1.2	798	0.7791	0x3f4771dc	1.2	859	0.8387	0x3f56b5ad	3.5
738	0.7204	0x3f386e1c	-1.2	799	0.7801	0x3f47b1ec	1.2	860	0.8397	0x3f56f5bd	3.6
739	0.7214	0x3f38ae2c	-1.1	800	0.7810	0x3f47f1fc	1.2	861	0.8407	0x3f5735cd	3.6
740	0.7224	0x3f38ee3c	-1.1	801	0.7820	0x3f48320d	1.3	862	0.8416	0x3f5775dd	3.7
741	0.7234	0x3f392e4c	-1.1	802	0.7830	0x3f48721d	1.3	863	0.8426	0x3f57b5ed	3.7
742	0.7243	0x3f396e5c	-1	803	0.7840	0x3f48b22d	1.4	864	0.8436	0x3f57f5fd	3.7
743	0.7253	0x3f39ae6c	-1	804	0.7849	0x3f48f23d	1.4	865	0.8446	0x3f58360e	3.8
744	0.7263	0x3f39ee7c	-0.9	805	0.7859	0x3f49324d	1.4	866	0.8456	0x3f58761e	3.8
745	0.7273	0x3f3a2e8c	-0.9	806	0.7869	0x3f49725d	1.5	867	0.8465	0x3f58b62e	3.9
746	0.7283	0x3f3a6e9c	-0.9	807	0.7879	0x3f49b26d	1.5	868	0.8475	0x3f58f63e	3.9
747	0.7292	0x3f3aaeac	-0.8	808	0.7889	0x3f49f27d	1.6	869	0.8485	0x3f59364e	3.9
748	0.7302	0x3f3aeebc	-0.8	809	0.7898	0x3f4a328d	1.6	870	0.8495	0x3f59765e	4
749	0.7312	0x3f3b2ecc	-0.8	810	0.7908	0x3f4a729d	1.6	871	0.8504	0x3f59b66e	4
750	0.7322	0x3f3b6edc	-0.7	811	0.7918	0x3f4ab2ad	1.7	872	0.8514	0x3f59f67e	4.1
751	0.7331	0x3f3baeec	-0.7	812	0.7928	0x3f4af2bd	1.7	873	0.8524	0x3f5a368e	4.1
752	0.7341	0x3f3beefc	-0.6	813	0.7937	0x3f4b32cd	1.7	874	0.8534	0x3f5a769e	4.1
753	0.7351	0x3f3c2f0c	-0.6	814	0.7947	0x3f4b72dd	1.8	875	0.8543	0x3f5ab6ae	4.2
754	0.7361	0x3f3c6f1c	-0.6	815	0.7957	0x3f4bb2ed	1.8	876	0.8553	0x3f5af6be	4.2
755	0.7370	0x3f3caf2c	-0.5	816	0.7967	0x3f4bf2fd	1.9	877	0.8563	0x3f5b36ce	4.3
756	0.7380	0x3f3cef3c	-0.5	817	0.7977	0x3f4c330d	1.9	878	0.8573	0x3f5b76de	4.3
757	0.7390	0x3f3d2f4c	-0.4	818	0.7986	0x3f4c731d	1.9	879	0.8583	0x3f5bb6ee	4.3
758	0.7400	0x3f3d6f5c	-0.4	819	0.7996	0x3f4cb32d	2	880	0.8592	0x3f5bf6fe	4.4
759	0.7410	0x3f3daf6c	-0.4	820	0.8006	0x3f4cf33d	2	881	0.8602	0x3f5c370e	4.4
760	0.7419	0x3f3def7c	-0.3	821	0.8016	0x3f4d334d	2.1	882	0.8612	0x3f5c771e	4.4
761	0.7429	0x3f3e2f8c	-0.3	822	0.8025	0x3f4d735d	2.1	883	0.8622	0x3f5cb72e	4.5
762	0.7439	0x3f3e6f9c	-0.2	823	0.8035	0x3f4db36d	2.1	884	0.8631	0x3f5cf73e	4.5
763	0.7449	0x3f3eafac	-0.2	824	0.8045	0x3f4df37d	2.2	885	0.8641	0x3f5d374e	4.6
764	0.7458	0x3f3eefbc	-0.2	825	0.8055	0x3f4e338d	2.2	886	0.8651	0x3f5d775e	4.6
765	0.7468	0x3f3f2fcc	-0.1	826	0.8065	0x3f4e739d	2.3	887	0.8661	0x3f5db76e	4.6
766	0.7478	0x3f3f6fdc	0	827	0.8074	0x3f4eb3ad	2.3	888	0.8671	0x3f5df77e	4.7
767	0.7488	0x3f3fafec	0	828	0.8084	0x3f4ef3bd	2.3	889	0.8680	0x3f5e378e	4.7
768	0.7498	0x3f3feffc	0	829	0.8094	0x3f4f33cd	2.4	890	0.8690	0x3f5e779e	4.8
769	0.7507	0x3f40300c	0	830	0.8104	0x3f4f73dd	2.4	891	0.8700	0x3f5eb7ae	4.8
770	0.7517	0x3f40701c	0	831	0.8113	0x3f4fb3ed	2.5	892	0.8710	0x3f5ef7be	4.8
771	0.7527	0x3f40b02c	0.1	832	0.8123	0x3f4ff3fd	2.5	893	0.8719	0x3f5f37ce	4.9
772	0.7537	0x3f40f03c	0.1	833	0.8133	0x3f50340d	2.5	894	0.8729	0x3f5f77de	4.9
773	0.7546	0x3f41304c	0.2	834	0.8143	0x3f50741d	2.6	895	0.8739	0x3f5fb7ee	5
774	0.7556	0x3f41705c	0.2	835	0.8152	0x3f50b42d	2.6	896	0.8749	0x3f5ff7fe	5
775	0.7566	0x3f41b06c	0.3	836	0.8162	0x3f50f43d	2.6	897	0.8759	0x3f60380e	5
776	0.7576	0x3f41f07c	0.3	837	0.8172	0x3f51344d	2.7	898	0.8768	0x3f60781e	5.1
777	0.7586	0x3f42308c	0.3	838	0.8182	0x3f51745d	2.7	899	0.8778	0x3f60b82e	5.1
778	0.7595	0x3f42709c	0.4	839	0.8192	0x3f51b46d	2.8	900	0.8788	0x3f60f83e	5.2

901	0.8798	0x3f61384e	5.2	943	0.9208	0x3f6bbaef	6.8	985	0.9619	0x3f763d8f	8.5
902	0.8807	0x3f61785e	5.2	944	0.9218	0x3f6bfaff	6.9	986	0.9629	0x3f767d9f	8.5
903	0.8817	0x3f61b86e	5.3	945	0.9228	0x3f6c3b0f	6.9	987	0.9638	0x3f76bdaf	8.6
904	0.8827	0x3f61f87e	5.3	946	0.9238	0x3f6c7b1f	7	988	0.9648	0x3f76fdbf	8.6
905	0.8837	0x3f62388e	5.3	947	0.9247	0x3f6cbb2f	7	989	0.9658	0x3f773dcf	8.6
906	0.8847	0x3f62789e	5.4	948	0.9257	0x3f6cfb3f	7	990	0.9668	0x3f777ddf	8.7
907	0.8856	0x3f62b8ae	5.4	949	0.9267	0x3f6d3b4f	7.1	991	0.9677	0x3f77bdef	8.7
908	0.8866	0x3f62f8be	5.5	950	0.9277	0x3f6d7b5f	7.1	992	0.9687	0x3f77fdff	8.7
909	0.8876	0x3f6338ce	5.5	951	0.9286	0x3f6dbb6f	7.1	993	0.9697	0x3f783e10	8.8
910	0.8886	0x3f6378de	5.5	952	0.9296	0x3f6dfb7f	7.2	994	0.9707	0x3f787e20	8.8
911	0.8895	0x3f63b8ee	5.6	953	0.9306	0x3f6e3b8f	7.2	995	0.9717	0x3f78be30	8.9
912	0.8905	0x3f63f8fe	5.6	954	0.9316	0x3f6e7b9f	7.3	996	0.9726	0x3f78fe40	8.9
913	0.8915	0x3f64390e	5.7	955	0.9326	0x3f6ebbaf	7.3	997	0.9736	0x3f793e50	8.9
914	0.8925	0x3f64791e	5.7	956	0.9335	0x3f6efbbf	7.3	998	0.9746	0x3f797e60	9
915	0.8935	0x3f64b92e	5.7	957	0.9345	0x3f6f3bcf	7.4	999	0.9756	0x3f79be70	9
916	0.8944	0x3f64f93e	5.8	958	0.9355	0x3f6f7bdf	7.4	1000	0.9765	0x3f79fe80	9.1
917	0.8954	0x3f65394e	5.8	959	0.9365	0x3f6fbbef	7.5	1001	0.9775	0x3f7a3e90	9.1
918	0.8964	0x3f65795e	5.9	960	0.9374	0x3f6ffbff	7.5	1002	0.9785	0x3f7a7ea0	9.1
919	0.8974	0x3f65b96e	5.9	961	0.9384	0x3f703c0f	7.5	1003	0.9795	0x3f7abeb0	9.2
920	0.8983	0x3f65f97e	5.9	962	0.9394	0x3f707c1f	7.6	1004	0.9804	0x3f7afec0	9.2
921	0.8993	0x3f66398e	6	963	0.9404	0x3f70bc2f	7.6	1005	0.9814	0x3f7b3ed0	9.3
922	0.9003	0x3f66799e	6	964	0.9413	0x3f70fc3f	7.7	1006	0.9824	0x3f7b7ee0	9.3
923	0.9013	0x3f66b9ae	6.1	965	0.9423	0x3f713c4f	7.7	1007	0.9834	0x3f7bbef0	9.3
924	0.9022	0x3f66f9be	6.1	966	0.9433	0x3f717c5f	7.7	1008	0.9844	0x3f7bff00	9.4
925	0.9032	0x3f6739ce	6.1	967	0.9443	0x3f71bc6f	7.8	1009	0.9853	0x3f7c3f10	9.4
926	0.9042	0x3f6779de	6.2	968	0.9453	0x3f71fc7f	7.8	1010	0.9863	0x3f7c7f20	9.5
927	0.9052	0x3f67b9ee	6.2	969	0.9462	0x3f723c8f	7.8	1011	0.9873	0x3f7cbf30	9.5
928	0.9062	0x3f67f9fe	6.2	970	0.9472	0x3f727c9f	7.9	1012	0.9883	0x3f7cff40	9.5
929	0.9071	0x3f683a0f	6.3	971	0.9482	0x3f72bcaf	7.9	1013	0.9892	0x3f7d3f50	9.6
930	0.9081	0x3f687a1f	6.3	972	0.9492	0x3f72fcbf	8	1014	0.9902	0x3f7d7f60	9.6
931	0.9091	0x3f68ba2f	6.4	973	0.9501	0x3f733ccf	8	1015	0.9912	0x3f7dbf70	9.6
932	0.9101	0x3f68fa3f	6.4	974	0.9511	0x3f737cdf	8	1016	0.9922	0x3f7dff80	9.7
933	0.9110	0x3f693a4f	6.4	975	0.9521	0x3f73bcef	8.1	1017	0.9932	0x3f7e3f90	9.7
934	0.9120	0x3f697a5f	6.5	976	0.9531	0x3f73fcff	8.1	1018	0.9941	0x3f7e7fa0	9.8
935	0.9130	0x3f69ba6f	6.5	977	0.9541	0x3f743d0f	8.2	1019	0.9951	0x3f7ebfb0	9.8
936	0.9140	0x3f69fa7f	6.6	978	0.9550	0x3f747d1f	8.2	1020	0.9961	0x3f7effc0	9.8
937	0.9150	0x3f6a3a8f	6.6	979	0.9560	0x3f74bd2f	8.2	1021	0.9971	0x3f7f3fd0	9.9
938	0.9159	0x3f6a7a9f	6.6	980	0.9570	0x3f74fd3f	8.3	1022	0.9980	0x3f7f7fe0	9.9
939	0.9169	0x3f6abaaf	6.7	981	0.9580	0x3f753d4f	8.3	1023	0.9990	0x3f7fbff0	10
940	0.9179	0x3f6afabf	6.7	982	0.9589	0x3f757d5f	8.4	1024	1.0000	0x3f800000	10
941	0.9189	0x3f6b3acf	6.8	983	0.9599	0x3f75bd6f	8.4				
942	0.9198	0x3f6b7adf	6.8	984	0.9609	0x3f75fd7f	8.4				

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 – [-oo, +10, 161]

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

0.0000 -oo	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 – Channel Dyn mgain – 49 lin scale values – [0, 24, 0.5]

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

0.00	00000000	8.00	41000000	16.00	41800000
0.50	3F000000	8.50	41080000	16.50	41840000
1.00	3F800000	9.00	41100000	17.00	41880000
1.50	3FC00000	9.50	41180000	17.50	418C0000
2.00	40000000	10.00	41200000	18.00	41900000
2.50	40200000	10.50	41280000	18.50	41940000
3.00	40400000	11.00	41300000	19.00	41980000
3.50	40600000	11.50	41380000	19.50	419C0000
4.00	40800000	12.00	41400000	20.00	41A00000
4.50	40900000	12.50	41480000	20.50	41A40000
5.00	40A00000	13.00	41500000	21.00	41A80000
5.50	40B00000	13.50	41580000	21.50	41AC0000
6.00	40C00000	14.00	41600000	22.00	41B00000
6.50	40D00000	14.50	41680000	22.50	41B40000
7.00	40E00000	15.00	41700000	23.00	41B80000
7.50	40F00000	15.50	41780000	23.50	41BC0000
				24.00	41C00000

Appendix – Automix weight – 49 lin scale values – [-12, 12, 0.5]

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

-12.00	C1400000	-4.00	C0800000	4.00	40800000
-11.50	C1380000	-3.50	C0600000	4.50	40900000
-11.00	C1300000	-3.00	C0400000	5.00	40A00000
-10.50	C1280000	-2.50	C0200000	5.50	40B00000
-10.00	C1200000	-2.00	C0000000	6.00	40C00000
-9.50	C1180000	-1.50	BFC00000	6.50	40D00000
-9.00	C1100000	-1.00	BF800000	7.00	40E00000
-8.50	C1080000	-0.50	BF000000	7.50	40F00000
-8.00	C1000000	0.00	00000000	8.00	41000000
-7.50	C0F00000	0.50	3F000000	8.50	41080000
-7.00	C0E00000	1.00	3F800000	9.00	41100000
-6.50	C0D00000	1.50	3FC00000	9.50	41180000
-6.00	C0C00000	2.00	40000000	10.00	41200000
-5.50	C0B00000	2.50	40200000	10.50	41280000
-5.00	C0A00000	3.00	40400000	11.00	41300000
-4.50	C0900000	3.50	40600000	11.50	41380000
				12.00	41400000

Appendix – Preamp trim – 145 lin scale values – [-18, 18, 0.25]

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

-18.00	C1900000	-8.75	C10C0000	0.50	3F000000	9.75	411C0000
-17.75	C18E0000	-8.50	C1080000	0.75	3F400000	10.00	41200000
-17.50	C18C0000	-8.25	C1040000	1.00	3F800000	10.25	41240000
-17.25	C18A0000	-8.00	C1000000	1.25	3FA00000	10.50	41280000
-17.00	C1880000	-7.75	C0F80000	1.50	3FC00000	10.75	412C0000
-16.75	C1860000	-7.50	C0F00000	1.75	3FE00000	11.00	41300000
-16.50	C1840000	-7.25	C0E80000	2.00	40000000	11.25	41340000
-16.25	C1820000	-7.00	C0E00000	2.25	40100000	11.50	41380000
-16.00	C1800000	-6.75	C0D80000	2.50	40200000	11.75	413C0000
-15.75	C17C0000	-6.50	C0D00000	2.75	40300000	12.00	41400000
-15.50	C1780000	-6.25	C0C80000	3.00	40400000	12.25	41440000
-15.25	C1740000	-6.00	C0C00000	3.25	40500000	12.50	41480000
-15.00	C1700000	-5.75	C0B80000	3.50	40600000	12.75	414C0000
-14.75	C16C0000	-5.50	C0B00000	3.75	40700000	13.00	41500000
-14.50	C1680000	-5.25	C0A80000	4.00	40800000	13.25	41540000
-14.25	C1640000	-5.00	C0A00000	4.25	40880000	13.50	41580000
-14.00	C1600000	-4.75	C0980000	4.50	40900000	13.75	415C0000
-13.75	C15C0000	-4.50	C0900000	4.75	40980000	14.00	41600000
-13.50	C1580000	-4.25	C0880000	5.00	40A00000	14.25	41640000
-13.25	C1540000	-4.00	C0800000	5.25	40A80000	14.50	41680000
-13.00	C1500000	-3.75	C0700000	5.50	40B00000	14.75	416C0000
-12.75	C14C0000	-3.50	C0600000	5.75	40B80000	15.00	41700000
-12.50	C1480000	-3.25	C0500000	6.00	40C00000	15.25	41740000
-12.25	C1440000	-3.00	C0400000	6.25	40C80000	15.50	41780000
-12.00	C1400000	-2.75	C0300000	6.50	40D00000	15.75	417C0000
-11.75	C13C0000	-2.50	C0200000	6.75	40D80000	16.00	41800000
-11.50	C1380000	-2.25	C0100000	7.00	40E00000	16.25	41820000
-11.25	C1340000	-2.00	C0000000	7.25	40E80000	16.50	41840000
-11.00	C1300000	-1.75	BFE00000	7.50	40F00000	16.75	41860000
-10.75	C12C0000	-1.50	BFC00000	7.75	40F80000	17.00	41880000
-10.50	C1280000	-1.25	BFA00000	8.00	41000000	17.25	418A0000
-10.25	C1240000	-1.00	BF800000	8.25	41040000	17.50	418C0000
-10.00	C1200000	-0.75	BF400000	8.50	41080000	17.75	418E0000
-9.75	C11C0000	-0.50	BF000000	8.75	410C0000	18.00	41900000
-9.50	C1180000	-0.25	BE800000	9.00	41100000		
-9.25	C1140000	0.00	00000000	9.25	41140000		
-9.00	C1100000	0.25	3E800000	9.50	41180000		

Appendix – Headamp gain – 145 lin scale values – [-12, 60, 0.5]

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

-12.00	C1400000	6.50	40D00000	25.00	41C80000	43.50	422E0000
-11.50	C1380000	7.00	40E00000	25.50	41CC0000	44.00	42300000
-11.00	C1300000	7.50	40F00000	26.00	41D00000	44.50	42320000
-10.50	C1280000	8.00	41000000	26.50	41D40000	45.00	42340000
-10.00	C1200000	8.50	41080000	27.00	41D80000	45.50	42360000
-9.50	C1180000	9.00	41100000	27.50	41DC0000	46.00	42380000
-9.00	C1100000	9.50	41180000	28.00	41E00000	46.50	423A0000
-8.50	C1080000	10.00	41200000	28.50	41E40000	47.00	423C0000
-8.00	C1000000	10.50	41280000	29.00	41E80000	47.50	423E0000
-7.50	C0F00000	11.00	41300000	29.50	41EC0000	48.00	42400000
-7.00	C0E00000	11.50	41380000	30.00	41F00000	48.50	42420000
-6.50	C0D00000	12.00	41400000	30.50	41F40000	49.00	42440000
-6.00	C0C00000	12.50	41480000	31.00	41F80000	49.50	42460000
-5.50	C0B00000	13.00	41500000	31.50	41FC0000	50.00	42480000
-5.00	C0A00000	13.50	41580000	32.00	42000000	50.50	424A0000
-4.50	C0900000	14.00	41600000	32.50	42020000	51.00	424C0000
-4.00	C0800000	14.50	41680000	33.00	42040000	51.50	424E0000
-3.50	C0600000	15.00	41700000	33.50	42060000	52.00	42500000
-3.00	C0400000	15.50	41780000	34.00	42080000	52.50	42520000
-2.50	C0200000	16.00	41800000	34.50	420A0000	53.00	42540000
-2.00	C0000000	16.50	41840000	35.00	420C0000	53.50	42560000
-1.50	BFC00000	17.00	41880000	35.50	420E0000	54.00	42580000
-1.00	BF800000	17.50	418C0000	36.00	42100000	54.50	425A0000
-0.50	BF000000	18.00	41900000	36.50	42120000	55.00	425C0000
0.00	00000000	18.50	41940000	37.00	42140000	55.50	425E0000
0.50	3F000000	19.00	41980000	37.50	42160000	56.00	42600000
1.00	3F800000	19.50	419C0000	38.00	42180000	56.50	42620000
1.50	3FC00000	20.00	41A00000	38.50	421A0000	57.00	42640000
2.00	40000000	20.50	41A40000	39.00	421C0000	57.50	42660000
2.50	40200000	21.00	41A80000	39.50	421E0000	58.00	42680000
3.00	40400000	21.50	41AC0000	40.00	42200000	58.50	426A0000
3.50	40600000	22.00	41B00000	40.50	42220000	59.00	426C0000
4.00	40800000	22.50	41B40000	41.00	42240000	59.50	426E0000
4.50	40900000	23.00	41B80000	41.50	42260000	60.00	42700000
5.00	40A00000	23.50	41BC0000	42.00	42280000		
5.50	40B00000	24.00	41C00000	42.50	422A0000		
6.00	40C00000	24.50	41C40000	43.00	422C0000		

Appendix – Channel Gate range – 58 lin scale values – [3, 60, 1.0]

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

3.00	40400000	22.00	41B00000	42.00	42280000
4.00	40800000	23.00	41B80000	43.00	422C0000
5.00	40A00000	24.00	41C00000	44.00	42300000
6.00	40C00000	25.00	41C80000	45.00	42340000
7.00	40E00000	26.00	41D00000	46.00	42380000
8.00	41000000	27.00	41D80000	47.00	423C0000
9.00	41100000	28.00	41E00000	48.00	42400000
10.00	41200000	29.00	41E80000	49.00	42440000
11.00	41300000	30.00	41F00000	50.00	42480000
12.00	41400000	31.00	41F80000	51.00	424C0000
13.00	41500000	32.00	42000000	52.00	42500000
14.00	41600000	33.00	42040000	53.00	42540000
15.00	41700000	34.00	42080000	54.00	42580000
16.00	41800000	35.00	420C0000	55.00	425C0000
17.00	41880000	36.00	42100000	56.00	42600000
18.00	41900000	37.00	42140000	57.00	42640000
19.00	41980000	38.00	42180000	58.00	42680000
20.00	41A00000	39.00	421C0000	59.00	426C0000
21.00	41A80000	40.00	42200000	60.00	42700000
		41.00	42240000		

Appendix – Channel EQ gain –121 lin scale values – [-15, 15, 0.250]

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

-15.00	C1700000	-6.75	C0D80000	1.50	3FC00000	9.75	411C0000
-14.75	C16C0000	-6.50	C0D00000	1.75	3FE00000	10.00	41200000
-14.50	C1680000	-6.25	C0C80000	2.00	40000000	10.25	41240000
-14.25	C1640000	-6.00	C0C00000	2.25	40100000	10.50	41280000
-14.00	C1600000	-5.75	C0B80000	2.50	40200000	10.75	412C0000
-13.75	C15C0000	-5.50	C0B00000	2.75	40300000	11.00	41300000
-13.50	C1580000	-5.25	C0A80000	3.00	40400000	11.25	41340000
-13.25	C1540000	-5.00	C0A00000	3.25	40500000	11.50	41380000
-13.00	C1500000	-4.75	C0980000	3.50	40600000	11.75	413C0000
-12.75	C14C0000	-4.50	C0900000	3.75	40700000	12.00	41400000
-12.50	C1480000	-4.25	C0880000	4.00	40800000	12.25	41440000
-12.25	C1440000	-4.00	C0800000	4.25	40880000	12.50	41480000
-12.00	C1400000	-3.75	C0700000	4.50	40900000	12.75	414C0000
-11.75	C13C0000	-3.50	C0600000	4.75	40980000	13.00	41500000
-11.50	C1380000	-3.25	C0500000	5.00	40A00000	13.25	41540000
-11.25	C1340000	-3.00	C0400000	5.25	40A80000	13.50	41580000
-11.00	C1300000	-2.75	C0300000	5.50	40B00000	13.75	415C0000
-10.75	C12C0000	-2.50	C0200000	5.75	40B80000	14.00	41600000
-10.50	C1280000	-2.25	C0100000	6.00	40C00000	14.25	41640000
-10.25	C1240000	-2.00	C0000000	6.25	40C80000	14.50	41680000
-10.00	C1200000	-1.75	BFE00000	6.50	40D00000	14.75	416C0000
-9.75	C11C0000	-1.50	BFC00000	6.75	40D80000	15.00	41700000
-9.50	C1180000	-1.25	BFA00000	7.00	40E00000		
-9.25	C1140000	-1.00	BF800000	7.25	40E80000		
-9.00	C1100000	-0.75	BF400000	7.50	40F00000		
-8.75	C10C0000	-0.50	BF000000	7.75	40F80000		
-8.50	C1080000	-0.25	BE800000	8.00	41000000		
-8.25	C1040000	0.00	00000000	8.25	41040000		
-8.00	C1000000	0.25	3E800000	8.50	41080000		
-7.75	C0F80000	0.50	3F000000	8.75	410C0000		
-7.50	C0F00000	0.75	3F400000	9.00	41100000		
-7.25	C0E80000	1.00	3F800000	9.25	41140000		
-7.00	C0E00000	1.25	3FA00000	9.50	41180000		

Appendix – Solo Dim Att – 41 lin scale values – [-40, 0, 1.0]

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

-40.00	C2200000	-30.00	C1F00000	-20.00	C1A00000	-10.00	C1200000
-39.00	C21C0000	-29.00	C1E80000	-19.00	C1980000	-9.00	C1100000
-38.00	C2180000	-28.00	C1E00000	-18.00	C1900000	-8.00	C1000000
-37.00	C2140000	-27.00	C1D80000	-17.00	C1880000	-7.00	C0E00000
-36.00	C2100000	-26.00	C1D00000	-16.00	C1800000	-6.00	C0C00000
-35.00	C20C0000	-25.00	C1C80000	-15.00	C1700000	-5.00	C0A00000
-34.00	C2080000	-24.00	C1C00000	-14.00	C1600000	-4.00	C0800000
-33.00	C2040000	-23.00	C1B80000	-13.00	C1500000	-3.00	C0400000
-32.00	C2000000	-22.00	C1B00000	-12.00	C1400000	-2.00	C0000000
-31.00	C1F80000	-21.00	C1A80000	-11.00	C1300000	-1.00	BF800000
						0.00	00000000

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

FX5…FX8:

```
 0 "GEQ2"  %011000 Dual Graphic EQ
 1 "GEQ"   %010111 Stereo Graphic EQ
 2 "TEQ2"  %011010 Dual TrueEQ
 3 "TEQ"   %011001 Strereo TrueEQ
 4 "DES2"  %101011 Dual DeEsser
 5 "DES"   %101010 Stereo DeEsser
 6 "P1A"   %101100 Stereo Xtec EQ1
 7 "P1A2"  %101101 Dual Xtec EQ1
 8 "PQ5"   %101110 Stereo Xtec EQ5
 9 "PQ5S"  %101111 Dual Xtec EQ5
10 "WAVD"  %011101 Wave Designer
11 "LIM"   %011110 Precision Limiter
12 "FAC"   %110000 Fair Comp
13 "FAC1M" %110001 M/S Fair Comp
14 "FAC2"  %110010 Dual Fair Comp
15 "LEC"   %110011 Leisure Comp
16 "LEC2"  %110100 Dual Leisure Comp
17 "ULC"   %110101 Ultimo Comp
18 "ULC2"  %110110 Dual Ultimo Comp
19 "ENH2"  %100000 Dual Enhancer
20 "ENH"   %011111 Stereo Enhancer
21 "EXC2"  %100010 Dual Exciter
22 "EXC"   %100001 Stereo Exciter
23 "IMG"   %100111 Stereo Imager
24 "EDI"   %111000 Edison EX1
25 "SON"   %110111 Sound Maxer
26 "AMP2"  %100100 Dual Guitar Amp
27 "AMP"   %100011 Stereo Guitar Amp
28 "DRV2"  %100110 Dual Tube Stage
29 "DRV"   %100101 Stereo Tube Stage
30 "PHAS"  %011011 Stereo Phaser
31 "FILT"  %101001 Mood Filter
32 "PAN"   %101000 Tremolo/Panner
33 "SUB"   %111001 Suboctaver
```

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, an 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/Windows)

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 to run on a Raspberry Pi or Mac.
Lazarus interfaces easily with C functions which can be compiled separately, and used at link time.

```
/*
 * X32UDP.c
 *     (POSIX compliant version, Linux... (WIndows too))
 *  CreatParam safe page Scene safe parameters Consoleed on: June 2, 2015
 *      Author: Patrick-Gilles Maillot
 *
 *      Copyright 2015, Patrick-Gilles Maillot
 *      This software is distributed under he 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.
 *
 *      Note: This package was updated to use select() timeouts rather than
 *      calls to poll() to handle non-blocking IO.
 */

#include <stdlib.h>
#include <string.h>
#include <sys/time.h>

#define BSIZE 512           // MAX receive buffer size
//
#ifdef __WIN32__
#include <winsock2.h>
WSADATA                     wsa;
struct sockaddr_in  Xip;
int                 Xip_len = sizeof(Xip);     // length of addresses
#else
#include <sys/socket.h>
#include <arpa/inet.h>
struct sockaddr_in  Xip;
socklen_t           Xip_len = sizeof(Xip);     // length of addresses
#endif
//
struct sockaddr*    Xip_addr = (struct sockaddr *)&Xip;
int                 Xfd;                       // X32 socket
fd_set              ufd;                       // associted file descriptor
//
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:
// -4 : (WIndows only) WSA init error
```

```
// -3 : Error on Sending data
// -2 : Socket creation error
// -1 : Error on waiting for data
//  0 : No error, no connection (timeout)
//  1 : Connected (connection validated with X32)
//
char   r_buf[64];          // receive buffer for /info command test
char   Info[8] = "/info";  // testing connection with /info request
struct timeval             timeout;

#ifdef __WIN32__
//Initialize winsock
       if (WSAStartup (MAKEWORD(2, 2), &wsa) != 0) {
              return -4; // Error on Windows sockets init
       }
#endif
//
// 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
       timeout.tv_sec = 0;                            // Set timeout for non
       timeout.tv_usec = 500000;                      // blocking recvfrom(): 500ms
// Validate connection by sending a /info command
       if (sendto (Xfd, Info, 8, 0, Xip_addr, Xip_len) < 0) {      // X32 sent something?
              return (-3);
       }
// register for receiving X32 console updates
       FD_ZERO (&ufd);
       FD_SET (Xfd, &ufd);
       if ((p_status = select(Xfd+1, &ufd, NULL, NULL, &timeout)) > 0) {
              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 reading (not connected)
       }
// Not connected on timeout after 500ms
       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 send data
       return (sendto (Xfd, buffer, length, 0, Xip_addr, Xip_len));
}
```

```
int X32Recv(char *buffer, int time_out) {
//
// Receives data from X32 server
//
// Input: String - Pointer to buffer to save data to,
//                 should be capable of receiving 512 bytes
// Input: integer - time_out (in us) for receiving data:
//                  depending on systems capabilities, positive and non-zero
//                  values may also result in no data (a typical value is 10ms)
//
// Returns int:
// -1 : Error on reading data
//  0 : No error, timeout reached or no data
// >0 : data length in the buffer
//
       struct timeval            timeout;

       timeout.tv_sec = 0;
       timeout.tv_usec = time_out; //Set timeout for non blocking recvfrom()
       FD_ZERO (&ufd);
       FD_SET (Xfd, &ufd);                                    // X32 sent something?
       if ((p_status = select(Xfd+1, &ufd, NULL, NULL, &timeout)) > 0) {
              return recvfrom(Xfd, buffer, BSIZE, 0, 0, 0); // return length
       } else if (p_status < 0) {
              return -1;                                     //An error occurred
       }
       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    ten_mills = 10000;
       int    status;

       status = X32Connect("192.168.1.62", "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, ten_mills);
                     printf ("Recv status: %d\n", r_len);
                     s_len = 0;
                     while (r_len--) {
                            if (r_buf[s_len] < ' ') putchar('~');
                            else putchar(r_buf[s_len]);
                            s_len++;
                     }
                     putchar('\n');
              }
       }
       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[BSIZE];              // 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 he 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 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
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[BSIZE];               // 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) {
      for i := 0; i < n; i++ {
            if p[i] == 0 {
                  p[i] = '~'
            }
      }
      fmt.Printf("%3d bytes - %s\n", n, p[0: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 – Misceallaneous

Floating point data representation

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

```
 / c h / 0 1 / m i x / f a d e r ~ ~ ~ ~ , f ~ ~ ? f f h
2f63682f30312f6d69782f666164657200000002c6600003f666668
```

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	3f4ccccе
0.9	3f666668
1.0	3f800000

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