M2T Command Reference v10Aug2017

Table of Contents

M2T Frame Commands

default	Restore factory default settings
desc	Frame description string
fplcdbl	Front panel LCD backlight brightness
fplock	Front panel lock status
fpmon	Front panel headphone monitor configuration
hversion	Frame hardware version
id	Frame id string
portctl	Communication port control flags
serial	Frame serial number
version	Frame firmware version

Termination: all commands are terminated with an ASCII **carriage return** character (hex code 0x0D), represented by **<CR>** in the examples below. All responses are terminated with an ASCII **carriage return, line feed** pair (hex codes 0x0D, 0x0A), represented by **<CRLF>** in the examples below. An ellipsis (**...**) represents members of an array that have been omitted from an example for the sake of brevity.

Verbose response: commands prefixed with an exclamation point (bang) character result in a "verbose" response containing both the name of the property being addressed and its current value (if any). The verbose response returns the property/value pair in the "assignment" form, for example **`OK ingn(2)=40 <CRLF>`**. This supports certain 3rd party control programming styles where the response needs to be self-describing.

default (Restore factory default settings)

This command may be used to restore the factory default settings.

Example:

	REQUEST	RESPONSE
COMMAND	**`default<CR>`**	**`OK`**

desc (Frame description string)

This command may be used as a query to read the user defined frame description, or as an update to set it. The data is a string type, with a limit of 30 characters.

Note: String arguments in commands need to be passed in **quoted** form, contained in a pair of **double-quote** (") characters. A problem arises when using the `desc` command to read or write a string that already contains double-quote characters, for example: `The "Hula" Room`. The solution is to **escape** the double quotes within `The "Hula" Room` so that it can be passed as a string argument for the `desc` command. This is done by preceding the double-quote characters with a **backslash** character like this: `The \"Hula\" Room`. Now it can be passed as a string argument to the `desc` command: `desc="The \"Hula\" Room"`. Since the **backslash** serves as the escape character in quoted-string arguments, it too must be escaped if it is part of the string, so `"foo\bar"` would become `"foo\\bar"`. If necessary, any character, printable or non-printable, can be represented in the hexadecimal escaped form `\xHH` where `HH` is any 2-digit hexadecimal number. The special escaped character forms `\r` (carriage return), `\n` (new line) and `\t` (tab) are also recognized.

Examples:

	REQUEST	RESPONSE
QUERY	**`!desc?<CR>`**	**`OK desc="Vocalists"<CRLF>`**
UPDATE	**`!desc="Vocalists"<CR>`**	**`OK desc="Vocalists"<CRLF>`**

fplcdbl (Front panel LCD backlight brightness)

This command may be used as a query to read the front panel LCD backlight brightness level, or as an update to set it. The data type is integer, in the range 0 to 3, where 0 is the minimum brightness (25%) and 3 the maximum brightness (100%).

Examples:

	REQUEST	RESPONSE
QUERY	**`!fplcdbl?<CR>`**	**`OK fplcdbl=1<CRLF>`**
UPDATE	**`!fplcdbl=3<CR>`**	**`OK fplcdbl=3<CRLF>`**

fplock (Front panel lock status)

This command may be used as a query to read the front panel lock status, or as an update to set it. The data type is integer, either 1 or 0, where 1 is "locked" and 0 is "not locked".

Examples:

	REQUEST	RESPONSE
QUERY	**`!fplock?<CR>`**	**`OK fplock=0<CRLF>`**
UPDATE	**`!fplock=1<CR>`**	**`OK fplock=1<CRLF>`**

fpmon (Front panel headphone monitor configuration)

This command may be used as a query to read the headphone monitor configuration, or as an update to set it. The data type is integer and is a code that specifies the audio channel assignment to headphone Left/Right. It may be in the range 0 to 5, with the following meanings:

Code	**Headphone Monitor Configuration**
0	Channel A1 (Mono)
1	Channel A2 (Mono)
2	Channel B1 (Mono)
3	Channel B2 (Mono)
4	Channels A1+A2 (Stereo)
5	Channels B1+B2 (Stereo)

Examples:

	REQUEST	RESPONSE
QUERY	**`!fpmon?<CR>`**	**`OK fpmon=4<CRLF>`**
UPDATE	**`!fpmon=3<CR>`**	**`OK fpmon=3<CRLF>`**

hversion (Frame hardware version)

This command may be used as a query to read the frame's hardware version number. The data is a string type.

Example:

	REQUEST	RESPONSE
QUERY	**`!hversion?<CR>`**	**`OK hversion="M2T-1"<CRLF>`**

id (Frame ID string)

This command may be used as a query to read the frame id string. This is the "name" of the device used by the control protocol and is always "M2T". The data is a string type.

Example:

	REQUEST	RESPONSE
QUERY	`!id?<CR>`	`OK id="M2T"<CRLF>`

portctl (Communication port control flags)

This command may be used as a query to read the network port control flags, or as an update to set them. The port is specified by using the address syntax. Addresses must be in the range 0 to 3, representing the one of the following:

- **0** - USB port
- **1** - TCP port 1
- **2** - TCP port 2
- **3** - HTTP port

The data type is integer, in the range 0 to 3. The value is a code representing the control settings for the communication port:

Code	Port Setting
0	Port is disabled
1	Port is receive only
2	Port is send only
3	Port is send/receive

If the port address is wildcarded, then the data type is an array of integer of size 4. By default, all communication ports are send/receive (full duplex) but in some special 3rd party control scenarios a port may need to be set otherwise. *Note: to preserve the ability to communicate with the device, changes to the USB port setting have no effect; the default of send/receive is always in force.*

Examples:

	REQUEST	RESPONSE

QUERY	`!portctl(1)?<CR>`	`OK portctl(1)=3<CRLF>`
QUERY	`!portctl(*)?<CR>`	`OK portctl(*)={3,3,3,3}<CRLF>`
UPDATE	`!portctl(1)=2<CR>`	`OK portctl(1)=2<CRLF>`
UPDATE	`!portctl(*)={3,3,3,3}<CR>`	`OK portctl(*)={3,3,3,3}<CRLF>`

serial (Frame serial number)

This command may be used as a query to read the frame's serial number. The data is a string type.

Example:

	REQUEST	RESPONSE
QUERY	`!serial?<CR>`	`OK serial="6300101"<CRLF>`

version (Frame firmware version)

This command may be used as a query to read the frame's firmware version number. The data is a string type.

Example:

	REQUEST	RESPONSE
QUERY	`!version?<CR>`	`OK version="1.0.1"<CRLF>`

M2T Network Setup Commands

defgate	Default gateway
dhcpen	DHCP enable
httpport	HTTP port number
ipaddr	IP address
macaddr	MAC address
netmask	Network mask
tcpport	TCP port number

Termination: all commands are terminated with an ASCII **carriage return** character (hex code 0x0D), represented by **<CR>** in the examples below. All responses are terminated with an ASCII **carriage return, line feed** pair (hex codes 0x0D, 0x0A), represented by **<CRLF>** in the examples below. An ellipsis (**...**) represents members of an array that have been omitted from an example for the sake of brevity.

Verbose response: commands prefixed with an exclamation point (bang) character result in a "verbose" response containing both the name of the property being addressed and its current value (if any). The verbose response returns the property/value pair in the "assignment" form, for example **`OK ingn(2)=40 <CRLF>`**. This supports certain 3rd party control programming styles where the response needs to be self-describing.

defgate (default gateway)

This command may be used as a query to read the Default Gateway address, or as an update to set it. The data type is string, containing the address in IP "dotted quad" format.

Example:

	REQUEST	RESPONSE
QUERY	**`defgate?<CR>`**	**`OK "172.16.4.1"<CRLF>`**
UPDATE	**`defgate="172.16.4.1"<CR>`**	**`OK<CRLF>`**

dhcpen (DHCP enable)

This command may be used as a query to read the DHCP enable status, or as an update to set it. The data type is integer, either "1" meaning that the DHCP client feature is enabled, or "0" meaning that it is not. If enabled, DHCP (Dynamic Host Configuration Protocol) is used at power up to obtain an IP address. **Note:** If this setting is changed, the new value takes effect the next time the device is powered up.

Example:

	REQUEST	RESPONSE
QUERY	**`dhcpen?<CR>`**	**`OK 0<CRLF>`**
UPDATE	**`dhcpen=1<CR>`**	**`OK<CRLF>`**

httpport (HTTP port number)

This command may be used as a query to read the HTTP port number assignment, or as an update to set it. The data type is integer, in the range 0 to 65535, representing the port number used for HTTP connections to the device. The default value is 80.

Example:

	REQUEST	RESPONSE
QUERY	**`httpport?<CR>`**	**`OK 80<CRLF>`**
UPDATE	**`httpport=80<CR>`**	**`OK<CRLF>`**

ipaddr (IP address)

This command may be used as a query to read the IP address of the device, or as an update to set it. The data type is string, containing the address in IP "dotted quad" format.

Example:

	REQUEST	RESPONSE
QUERY	**`ipaddr?<CR>`**	**`OK "172.16.4.151"<CRLF>`**
UPDATE	**`ipaddr="172.16.4.151"<CR>`**	**`OK<CRLF>`**

macaddr (MAC address)

This command may be used as a query to read the ethernet MAC address of the device. The data type is string, containing the address in IEEE MAC-48 format.

Example:

	REQUEST	RESPONSE
QUERY	**`macaddr?<CR>`**	**`OK "00-24-34-32-00-22"<CRLF>`**

netmask (network mask)

This command may be used as a query to read the Network Mask, or as an update to set it. The data type is string, containing the mask in IP "dotted quad" format.

Example:

	REQUEST	RESPONSE
QUERY	**`netmask?<CR>`**	**`OK "255.255.255.0"<CRLF>`**
UPDATE	**`netmask="255.255.255.0"<CR>`**	**`OK<CRLF>`**

tcpport (TCP port number)

This command may be used as a query to read the TCP port number assignment, or as an update to set it. The data type is integer, in the range 0 to 65535, representing the port number used for TCP connections to the device. The default value is 4080.

Example:

	REQUEST	RESPONSE
QUERY	**`tcpport?<CR>`**	**`OK 4080<CRLF>`**
UPDATE	**`tcpport=4080<CR>`**	**`OK<CRLF>`**

M2T Receiver Commands

rxafbal	Receiver audio balance
rxafbstfreq	Receiver audio HF boost frequency
rxafbstgain	Receiver audio HF boost gain
rxaflim	Receiver audio limiter threshold
rxafwidth	Receiver audio width
rxbatttype	Receiver battery type
rxfplock	Receiver front panel lock status
rxir	Receiver IR Sync
rxlcdbltime	Receiver LCD backlight timeout period
rxmetmode	Receiver audio meter mode
rxmixmode	Receiver audio mix mode
rxname	Receiver name
rxpregain	Receiver audio pre-gain
rxusetxname	Receiver "use transmitter name" flag

Termination: all commands are terminated with an ASCII **carriage return** character (hex code 0x0D), represented by `<CR>` in the examples below. All responses are terminated with an ASCII **carriage return, line feed** pair (hex codes 0x0D, 0x0A), represented by `<CRLF>` in the examples below. An ellipsis (**...**) represents members of an array that have been omitted from an example for the sake of brevity.

Verbose response: commands prefixed with an exclamation point (bang) character result in a "verbose" response containing both the name of the property being addressed and its current value (if any). The verbose response returns the property/value pair in the "assignment" form, for example `OK rxphase(2)=0 <CRLF>`. This supports certain 3rd party control programming styles where the response needs to be self-describing.

rxafbal (Receiver audio balance)

This command may be used as a query to read the receiver audio balance, or as an update to set it. The receiver is specified by using the address syntax. Addresses must be in the range 1 to 2. The data type is integer, in the range is -100 to +100 and represents the balance where 0 means equal levels in left and right audio channels. If the receiver address is wildcarded, then the data type is an array of integer of size 2. In this case the value **99** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	`!rxafbal(1)?<CR>`	`OK rxafwidth(1)=100<CRLF>`
QUERY	`!rxafbal(*)?<CR>`	`OK  rxafwidth(*)={50,100}<CRLF>`
UPDATE	`!rxafbal(1)=100<CR>`	`OK rxafwidth(1)=100<CRLF>`
UPDATE	`!rxafbal(*)={50,100}<CR>`	`OK rxafwidth(*)={50,100}<CRLF>`

rxafbstfreq (Receiver audio HF boost frequency)

This command may be used as a query to read the audio "high frequency boost" corner frequency, or as an update to set it. The receiver is specified by using the address syntax. Addresses must be in the range 1 to 2. If the receiver address is wildcarded, then the data type is an array of integer of size 2. In this case the value **99** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command. The data type is integer, and is a code representing one of the following corner frequencies:

Code	**HF Boost Corner Frequency**
0	5kHz
1	7kHz

Examples:

	REQUEST	RESPONSE
QUERY	`!rxafbstfreq(1)?<CR>`	`OK rxafbstfreq(1)=1<CRLF>`
QUERY	`!rxafbstfreq(*)?<CR>`	`OK rxafbstfreq(*)={0,0}<CRLF>`
UPDATE	`!rxafbstfreq(2)=1<CR>`	`OK rxafbstfreq(2)=0<CRLF>`
UPDATE	`!rxafbstfreq(*)={1,1}<CR>`	`OK rxafbstfreq(*)={1,1}<CRLF>`

rxafbstgain (Receiver audio HF boost gain)

This command may be used as a query to read the audio "high frequency boost" gain, or as an update to set it. The receiver is specified by using the address syntax. Addresses must be in the range 1 to 2. If the receiver address is wildcarded, then the data type is an array of integer of size 2. In this case the value **99** may be used in an update to indicate that a particular receiver is to

remain unchanged by the command. The data type is integer, and is a code representing one of the following gain values:

Code	HF Boost Gain
0	0dB
1	3dB
2	6dB
3	9dB

Examples:

	REQUEST	RESPONSE
QUERY	`!rxafbstgain(1)?<CR>`	`OK rxafbstgain(1)=2<CRLF>`
QUERY	`!rxafbstgain(*)?<CR>`	`OK rxafbstgain(*)={0,0}<CRLF>`
UPDATE	`!rxafbstgain(2)=2<CR>`	`OK rxafbstgain(2)=2<CRLF>`
UPDATE	`!rxafbstgain(*)={1,1}<CR>`	`OK rxafbstgain(*)={1,1}<CRLF>`

rxaflim (Receiver audio limiter threshold)

This command may be used as a query to read the audio limiter threshold, or as an update to set it. The receiver is specified by using the address syntax. Addresses must be in the range 1 to 2. If the receiver address is wildcarded, then the data type is an array of integer of size 2. In this case the value **99** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command. The data type is integer, and is a code representing one of the following threshold values in dB relative to Full Scale:

Code	Limiter Threshold
0	0dBFS
1	-3dBFS
2	-6dBFS
3	-9dBFS
4	-12dBFS
5	-15dBFS
6	-18dBFS

Examples:

	REQUEST	RESPONSE
QUERY	`!rxaflim(1)?<CR>`	`OK rxaflim(1)=2<CRLF>`
QUERY	`!rxaflim(*)?<CR>`	`OK rxaflim(*)={3,4}<CRLF>`
UPDATE	`!rxaflim(2)=2<CR>`	`OK rxaflim(2)=2<CRLF>`
UPDATE	`!rxaflim(*)={2,3}<CR>`	`OK rxaflim(*)={2,3}<CRLF>`

rxafwidth (Receiver audio width)

This command may be used as a query to read the receiver audio width, or as an update to set it. The receiver is specified by using the address syntax. Addresses must be in the range 1 to 2. The data type is integer, in the range is 0 to 100 and represents the width (left/right channel separation) as a percentage. If the receiver address is wildcarded, then the data type is an array of integer of size 2. In this case the value **99** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	`!rxafwidth(1)?<CR>`	`OK rxafwidth(1)=100<CRLF>`
QUERY	`!rxafwidth(*)?<CR>`	`OK  rxafwidth(*)={50,100}<CRLF>`
UPDATE	`!rxafwidth(1)=100<CR>`	`OK rxafwidth(1)=100<CRLF>`
UPDATE	`!rxafwidth(*)={50,100}<CR>`	`OK rxafwidth(*)={50,100}<CRLF>`

rxbatttype (Receiver battery type)

This command may be used as a query to read the battery type, or as an update to set it. The receiver is specified by using the address syntax. Addresses must be in the range 1 to 2. If the receiver address is wildcarded, then the data type is an array of integer of size 2. In this case the value **99** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command. The data type is integer, and is a code representing one of the following battery types:

Code	**Battery Type**
0	Alkaline
1	Lithium

Examples:

	REQUEST	RESPONSE
QUERY	`!rxbatttype(1)?<CR>`	`OK rxbatttype(1)=1<CRLF>`
QUERY	`!rxbatttype(*)?<CR>`	`OK rxbatttype(*)={0,0}<CRLF>`
UPDATE	`!rxbatttype(2)=1<CR>`	`OK rxbatttype(2)=0<CRLF>`
UPDATE	`!rxbatttype(*)={1,1}<CR>`	`OK rxbatttype(*)={1,1}<CRLF>`

rxfplock (Receiver front panel lock status)

This command may be used as a query to read the front panel lock status, or as an update to set it. The receiver is specified by using the address syntax. Addresses must be in the range 1 to 2. The data type is integer, either 1 or 0, where 1 is "locked" and 0 is "not locked". If the receiver address is wildcarded, then the data type is an array of integer of size 2. In this case the value **99** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	`!rxfplock(1)?<CR>`	`OK rxfplock(1)=0<CRLF>`
QUERY	`!rxfplock(*)?<CR>`	`OK  rxfplock(*)={0,0}<CRLF>`
UPDATE	`!rxfplock(1)=1<CR>`	`OK rxfplock(1)=1<CRLF>`
UPDATE	`!rxfplock(*)={0,1}<CR>`	`OK rxfplock(*)={0,1}<CRLF>`

rxir (Receiver IR sync)

This command may be used as an update to initiate an IR sync between the transmitter and the receiver. The receiver is specified by using the address syntax. Addresses must be in the range 1 to 2. The data type is integer, and is a code representing one of the following transaction types:

Code	**IR Sync Transaction**
0	Send frequency to receiver
1	Send all settings to receiver
2	Get frequency from

	receiver
3	Get all settings from receiver

Examples:

	REQUEST	RESPONSE
UPDATE	`!rxir(2)=1<CR>`	`OK rxir(2)=1<CRLF>`

rxlcdbltime (Receiver LCD backlight timeout period)

This command may be used as a query to read the LCD backlight timeout period, or as an update to set it. The receiver is specified by using the address syntax. Addresses must be in the range 1 to 2. If the receiver address is wildcarded, then the data type is an array of integer of size 2. In this case the value **99** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command. The data type is integer, and is a code representing one of the following:

Code	**Backlight Timeout Period**
0	Always on
1	30 seconds
2	5 minutes

Examples:

	REQUEST	RESPONSE
QUERY	`!rxlcdbltime(1)?<CR>`	`OK rxlcdbltime(1)=2<CRLF>`
QUERY	`!rxlcdbltime(*)?<CR>`	`OK rxlcdbltime(*)={0,0}<CRLF>`
UPDATE	`!rxlcdbltime(2)=2<CR>`	`OK rxlcdbltime(2)=2<CRLF>`
UPDATE	`!rxlcdbltime(*)={1,1}<CR>`	`OK rxlcdbltime(*)={1,1}<CRLF>`

rxmetmode (Receiver audio meter mode)

This command may be used as a query to read the audio meter mode, or as an update to set it. The receiver is specified by using the address syntax. Addresses must be in the range 1 to 2. If the receiver address is wildcarded, then the data type is an array of integer of size 2. In this case the value **99** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command. The data type is integer, and is a code representing one of the following modes:

Code	Meter Mode
0	pre-mixer
1	post-mixer

Examples:

	REQUEST	RESPONSE
QUERY	`!rxmetmode(1)?<CR>`	`OK rxmetmode(1)=1<CRLF>`
QUERY	`!rxmetmode(*)?<CR>`	`OK rxmetmode(*)={0,0}<CRLF>`
UPDATE	`!rxmetmode(2)=1<CR>`	`OK rxmetmode(2)=0<CRLF>`
UPDATE	`!rxmetmode(*)={1,1}<CR>`	`OK rxmetmode(*)={1,1}<CRLF>`

rxmixmode (Receiver audio mix mode)

This command may be used as a query to read the audio mix mode, or as an update to set it. The receiver is specified by using the address syntax. Addresses must be in the range 1 to 2. If the receiver address is wildcarded, then the data type is an array of integer of size 2. In this case the value **99** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command. The data type is integer, and is a code representing one of the following mix modes::

Code	Mix Mode
0	Stereo
1	Mono (Ch1)
2	Mono (Ch2)
3	Mono (Mono Ch1+Ch2)
4	Custom
5	Reverse Stereo

Examples:

	REQUEST	RESPONSE
QUERY	`!rxmixmode(1)?<CR>`	`OK rxmixmode(1)=2<CRLF>`

QUERY	**`!rxmixmode(*)?<CR>`**	**`OK rxmixmode(*)={3,4}<CRLF>`**
UPDATE	**`!rxmixmode(2)=2<CR>`**	**`OK rxmixmode(2)=2<CRLF>`**
UPDATE	**`!rxmixmode(*)={2,3}<CR>`**	**`OK rxmixmode(*)={2,3}<CRLF>`**

rxname (Receiver name)

This command may be used as a query to read the receiver name, or as an update to set it. The channel is specified by using the address syntax. Addresses must be in the range 1 to 2. The data type is string, with a limit of 15 characters.

Note: String arguments in commands need to be passed in **quoted** form, contained in a pair of **double-quote** (") characters. A problem arises when using the `inlb` command to read or write a string that already contains double-quote characters, for example: `The "Hula" Room`. The solution is to **escape** the double quotes within `The "Hula" Room` so that it can be passed as a string argument for the `inlb` command. This is done by preceding the double-quote characters with a **backslash** character like this: `The \"Hula\" Room`. Now it can be passed as a string argument to the `inlb` command: `inlb(1)="The \"Hula\" Room"`. Since the **backslash** serves as the escape character in quoted-string arguments, it too must be escaped if it is part of the string, so `"foo\bar"` would become `"foo\\bar".` If necessary, any character, printable or non-printable, can be represented in the hexadecimal escaped form `\xHH` where `HH` is any 2-digit hexadecimal number. The special escaped character forms `\r` (carriage return), `\n` (new line) and `\t` (tab) are also recognized..

Examples:

	REQUEST	RESPONSE
QUERY	**`!rxname(1)?<CR>`**	**`OK rxname(1)="Poppy"<CRLF>`**
UPDATE	**`!rxname(2)="David"<CR>`**	**`OK rxname(2)="David"<CRLF>`**

rxpregain (Receiver audio pre-gain)

This command may be used as a query to read the audio pre-gain, or as an update to set it. The receiver is specified by using the address syntax. Addresses must be in the range 1 to 2. If the receiver address is wildcarded, then the data type is an array of integer of size 2. In this case the value **99** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command. The data type is integer, and is a code representing one of the following gain values:

Code	Pre-gain
0	0dB
1	3dB
2	6dB
3	9dB
4	12dB

Examples:

	REQUEST	RESPONSE
QUERY	`!rxpregain(1)?<CR>`	`OK rxpregain(1)=2<CRLF>`
QUERY	`!rxpregain(*)?<CR>`	`OK rxpregain(*)={0,0}<CRLF>`
UPDATE	`!rxpregain(2)=2<CR>`	`OK rxpregain(2)=2<CRLF>`
UPDATE	`!rxpregain(*)={1,1}<CR>`	`OK rxpregain(*)={1,1}<CRLF>`

rxusetxname (Receiver "use transmitter name" flag)

This command may be used as a query to read the "use transmitter name as receiver name" flag, or as an update to set it. The receiver is specified by using the address syntax. Addresses must be in the range 1 to 2. The data type is integer, either 1 or 0, where 1 is "use tx name as rx name" and 0 is "do not use tx name as rx name". If the receiver address is wildcarded, then the data type is an array of integer of size 2. In this case the value **99** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	`!rxusetxname(1)?<CR>`	`OK rxusetxname(1)=0<CRLF>`
QUERY	`!rxusetxname(*)?<CR>`	`OK  rxusetxname(*)={0,0}<CRLF>`
UPDATE	`!rxusetxname(1)=1<CR>`	`OK rxusetxname(1)=1<CRLF>`
UPDATE	`!rxusetxname(*)={0,1}<CR>`	`OK rxusetxname(*)={0,1}<CRLF>`

M2T Transmitter Commands

txafclip	Transmitter audio input clip status
txafinlevel	Transmitter audio input nominal level
txafmeter	Transmitter audio input meter
txafmute	Transmitter audio input mute
txafmutetog	Transmitter audio input mute toggle
txafname	Transmitter audio input name
txafphase	Transmitter audio input phase
txaftrim	Transmitter audio input trim
txaftype	Transmitter audio input type
txrfenable	Transmitter RF carrier enable
txrffreq	Transmitter RF carrier frequency
txrfname	Transmitter RF carrier name
txrfpwr	Transmitter RF carrier power

Termination: all commands are terminated with an ASCII **carriage return** character (hex code 0x0D), represented by **`<CR>`** in the examples below. All responses are terminated with an ASCII **carriage return, line feed** pair (hex codes 0x0D, 0x0A), represented by **`<CRLF>`** in the examples below. An ellipsis (**...**) represents members of an array that have been omitted from an example for the sake of brevity.

Verbose response: commands prefixed with an exclamation point (bang) character result in a "verbose" response containing both the name of the property being addressed and its current value (if any). The verbose response returns the property/value pair in the "assignment" form, for example **`OK rxphase(2)=0 <CRLF>`**. This supports certain 3rd party control programming styles where the response needs to be self-describing.

txafclip (Transmitter audio input clip status)

This command may be used as a query to read the transmitter audio input clip status. The audio channel is specified by using the 2 dimensional address syntax. Addresses for the first dimension (transmitter) must be in the range 1 to 2. Addresses for the second dimension (audio channel) must be in the range 1 to 2. The data type is integer, either "1" meaning that the audio channel is clipping, or "0" meaning that it is not. If the transmitter address or audio channel address is wildcarded, then the data type is an array of integer of size 2. If both are wildcarded then the data type is an array of integer of size 4.

Examples:

	REQUEST	RESPONSE
QUERY	`!txafclip(1,1)?<CR>`	`OK txafclip(1,1)=0<CRLF>`
QUERY	`!txafclip(*,1)?<CR>`	`OK txafclip(*,1)={0,1}<CRLF>`
QUERY	`!txafclip(2,*)?<CR>`	`OK txafclip(2,*)={0,0}<CRLF>`
QUERY	`!txafclip(*,*)?<CR>`	`OK txafclip(*,*)={0,1,0,0}<CRLF>`

txafinlevel (Transmitter audio input nominal level)

This command may be used as a query to read the transmitter audio input nominal level or as an update to set it. The audio channel is specified by using the 2 dimensional address syntax. Addresses for the first dimension (transmitter) must be in the range 1 to 2. Addresses for the second dimension (audio channel) must be in the range 1 to 2. The data type is integer and is a code that specifies the nominal input level (the 0dB point on the audio input scale). It may be either 0 or 1, with the following meanings:

Code	**Nominal Input Level**
0	-10 dBV
1	+4 dBu

If the transmitter address or audio channel address is wildcarded, then the data type is an array of integer of size 2. If both are wildcarded then the data type is an array of integer of size 4. In these cases the value 99 may be used in an update to indicate that a particular input level is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	`!txafinlevel(1,1)?<CR>`	`OK txafinlevel(1,1)=0<CRLF>`
QUERY	`!txafinlevel(*,1)?<CR>`	`OK txafinlevel(*,1)={1,1}<CRLF>`
QUERY	`!txafinlevel(2,*)?<CR>`	`OK txafinlevel(2,*)={1,1}<CRLF>`
QUERY	`!txafinlevel(*,*)?<CR>`	`OK txafinlevel(*,*)={1,1,0,0}<CRLF>`
UPDATE	`!txafinlevel(1,1)=0<CR>`	`OK txafinlevel(1,1)=0<CRLF>`
UPDATE	`!txafinlevel(*,0)={1,1}<CR>`	`OK txafinlevel(*,0)={1,1}<CRLF>`
UPDATE	`!txafinlevel(2,*)={1,0}<CR>`	`OK txafinlevel(1,1)=0<CRLF>`
UPDATE	`!txafinlevel(*,*)={1,1,0,0}<CR>`	`OK txafinlevel(*,*)={1,1,0,0}<CRLF>`

txafmeter (Transmitter audio input meter)

This command may be used as a query to read the transmitter audio input meter. The audio channel is specified by using the 2 dimensional address syntax. Addresses for the first dimension (transmitter) must be in the range 1 to 2. Addresses for the second dimension (audio channel) must be in the range 1 to 2. The data type is integer, in the range -40 to +20 and represents the input level in dB nominal. If the transmitter address or audio channel address is wildcarded, then the data type is an array of integer of size 2. If both are wildcarded then the data type is an array of integer of size 4.

Examples:

	REQUEST	RESPONSE
QUERY	`!txafmeter(1,1)?<CR>`	`OK txafmeter(1,1)=-7<CRLF>`
QUERY	`!txafmeter(*,1)?<CR>`	`OK txafmeter(*,1)={-21,3}<CRLF>`
QUERY	`!txafmeter(2,*)?<CR>`	`OK txafmeter(2,*)={-12,-6}<CRLF>`
QUERY	`!txafmeter(*,*)?<CR>`	`OK txafmeter(*,*)={-15,1,-9,-33}<CRLF>`

txafmute (Transmitter audio input mute)

This command may be used as a query to read the transmitter audio input mute status or as an update to set it. The audio channel is specified by using the 2 dimensional address syntax. Addresses for the first dimension (transmitter) must be in the range 1 to 2. Addresses for the second dimension (audio channel) must be in the range 1 to 2. The data type is integer, either "1" meaning that the audio channel is muted, or "0" meaning that it is not. If the transmitter address or audio channel address is wildcarded, then the data type is an array of integer of size 2. If both are wildcarded then the data type is an array of integer of size 4. In these cases the value 99 may be used in an update to indicate that a particular input mute is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	`!txafmute(1,1)?<CR>`	`OK txafmute(1,1)=0<CRLF>`
QUERY	`!txafmute(*,1)?<CR>`	`OK txafmute(*,1)={1,1}<CRLF>`
QUERY	`!txafmute(2,*)?<CR>`	`OK txafmute(2,*)={1,1}<CRLF>`
QUERY	`!txafmute(*,*)?<CR>`	`OK txafmute(*,*)={1,1,0,0}<CRLF>`
UPDATE	`!txafmute(1,1)=0<CR>`	`OK txafmute(1,1)=0<CRLF>`

UPDATE	**!txafmute(*,0)={1,1}<CR>**	**OK txafmute(*,0)={1,1}<CRLF>**
UPDATE	**!txafmute(2,*)={1,0}<CR>**	**OK txafmute(1,1)=0<CRLF>**
UPDATE	**!txafmute(*,*)={1,1,0,0}<CR>**	**OK txafmute(*,*)={1,1,0,0}<CRLF>**

txafmutetog (Transmitter audio input mute toggle)

This command may be used as a query to toggle the transmitter audio input mute status. The audio channel is specified by using the 2 dimensional address syntax. Addresses for the first dimension (transmitter) must be in the range 1 to 2. Addresses for the second dimension (audio channel) must be in the range 1 to 2. The response to a verbose command is the new mute status for the audio inputs that are addressed.

Examples:

	REQUEST	RESPONSE
COMMAND	**!txafmutetog(1,1)?<CR>**	**OK txafmute(1,1)=0<CRLF>**
COMMAND	**!txafmutetog(*,1)?<CR>**	**OK txafmute(*,1)={1,1}<CRLF>**
COMMAND	**!txafmutetog(2,*)?<CR>**	**OK txafmute(2,*)={0,0}<CRLF>**
COMMAND	**!txafmutetog(*,*)?<CR>**	**OK txafmute(*,*)={1,1,0,0}<CRLF>**

txafname (Transmitter audio input name)

This command may be used as a query to read the audio input name, or as an update to set it. The audio channel is specified by using the 2 dimensional address syntax. Addresses for the first dimension (transmitter) must be in the range 1 to 2. Addresses for the second dimension (audio channel) must be in the range 1 to 2. The data type is string, with a limit of 15 characters.

Note: String arguments in commands need to be passed in **quoted** form, contained in a pair of **double-quote** (") characters. A problem arises when using the `inlb` command to read or write a string that already contains double-quote characters, for example: `The "Hula" Room`. The solution is to **escape** the double quotes within `The "Hula" Room` so that it can be passed as a string argument for the `inlb` command. This is done by preceding the double-quote characters with a **backslash** character like this: `The \"Hula\" Room`. Now it can be passed as a string argument to the `inlb` command: `inlb(1)="The \"Hula\" Room"`. Since the **backslash** serves as the escape character in quoted-string arguments, it too must be escaped if it is part of the string, so `"foo\bar"` would become `"foo\\bar".` If necessary, any character, printable or non-printable, can be represented in the hexadecimal escaped form `\xHH` where `HH` is any 2-digit hexadecimal number. The special escaped character forms `\r` (carriage return), `\n` (new line) and `\t` (tab) are also recognized..

Examples:

	REQUEST	RESPONSE
QUERY	**!txafname(1,1)?<CR>**	**OK txrfname(1,1)="Poppy Left"<CRLF>**
UPDATE	**!txafname(1,2)="Poppy Right"<CR>**	**OK rxname(1,2)="Poppy Left"<CRLF>**

txafphase (Transmitter audio input phase)

This command may be used as a query to read the transmitter audio input phase status or as an update to set it. The audio channel is specified by using the 2 dimensional address syntax. Addresses for the first dimension (transmitter) must be in the range 1 to 2. Addresses for the second dimension (audio channel) must be in the range 1 to 2. The data type is integer, either "1" meaning that the audio phase is inverted (shifted by 180 degrees), or "0" meaning that it is not. If the transmitter address or audio channel address is wildcarded, then the data type is an array of integer of size 2. If both are wildcarded then the data type is an array of integer of size 4. In these cases the value 99 may be used in an update to indicate that a particular input phase is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	**!txafphase(1,1)?<CR>**	**OK txafphase(1,1)=0<CRLF>**
QUERY	**!txafphase(*,1)?<CR>**	**OK txafphase(*,1)={1,1}<CRLF>**
QUERY	**!txafphase(2,*)?<CR>**	**OK txafphase(2,*)={1,1}<CRLF>**
QUERY	**!txafphase(*,*)?<CR>**	**OK txafphase(*,*)={1,1,0,0}<CRLF>**
UPDATE	**!txafphase(1,1)=0<CR>**	**OK txafphase(1,1)=0<CRLF>**
UPDATE	**!txafphase(*,0)={1,1}<CR>**	**OK txafphase(*,0)={1,1}<CRLF>**
UPDATE	**!txafphase(2,*)={1,0}<CR>**	**OK txafphase(1,1)=0<CRLF>**
UPDATE	**!txafphase(*,*)={1,1,0,0}<CR>**	**OK txafphase(*,*)={1,1,0,0}<CRLF>**

txaftrim (Transmitter audio input trim)

This command may be used as a query to read the transmitter audio input trim or as an update to set it. The audio channel is specified by using the 2 dimensional address syntax. Addresses for the first dimension (transmitter) must be in the range 1 to 2. Addresses for the second dimension (audio channel) must be in the range 1 to 2. The data type is integer, in the range -5 to +5, representing the gain in dB. If the transmitter address or audio channel address is wildcarded, then the data type is an array of integer of size 2. If both are wildcarded then the data type is an array of integer of size 4. In these cases the value 99 may be used in an update to indicate that a particular audio trim is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	**`!txaftrim(1,1)?<CR>`**	**`OK txaftrim(1,1)=0<CRLF>`**
QUERY	**`!txaftrim(*,1)?<CR>`**	**`OK txaftrim(*,1)={0,-3}<CRLF>`**
QUERY	**`!txaftrim(2,*)?<CR>`**	**`OK txaftrim(2,*)={0,0}<CRLF>`**
QUERY	**`!txaftrim(*,*)?<CR>`**	**`OK txaftrim(*,*)={-3,-3,0,0}<CRLF>`**
UPDATE	**`!txaftrim(1,1)=0<CR>`**	**`OK txaftrim(1,1)=0<CRLF>`**
UPDATE	**`!txaftrim(*,0)={3,0}<CR>`**	**`OK txaftrim(*,0)={3,0}<CRLF>`**
UPDATE	**`!txaftrim(2,*)={0,0}<CR>`**	**`OK txaftrim(0,0)=0<CRLF>`**
UPDATE	**`!txaftrim(*,*)={0,0,0,0}<CR>`**	**`OK txaftrim(*,*)={0,0,0,0}<CRLF>`**

txaftype (Transmitter audio input type)

This command may be used as a query to read the transmitter audio input type or as an update to set it. The audio channel is specified by using the 2 dimensional address syntax. Addresses for the first dimension (transmitter) must be in the range 1 to 2. Addresses for the second dimension (audio channel) must be in the range 1 to 2. The data type is integer and is a code that specifies the input type. It may be either 0 or 1, with the following meanings:

Code	**Audio Input Type**
0	analog audio (from rear panel XLR)
1	digital audio (from Dante interface)

If the transmitter address or audio channel address is wildcarded, then the data type is an array of integer of size 2. If both are wildcarded then the data type is an array of integer of size 4. In these cases the value 99 may be used in an update to indicate that a particular input type is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	**`!txaftype(1,1)?<CR>`**	**`OK txaftype(1,1)=0<CRLF>`**
QUERY	**`!txaftype(*,1)?<CR>`**	**`OK txaftype(*,1)={1,1}<CRLF>`**

QUERY	**!txaftype(2,*)?<CR>**	**OK txaftype(2,*)={1,1}<CRLF>**
QUERY	**!txaftype(*,*)?<CR>**	**OK txaftype(*,*)={1,1,0,0}<CRLF>**
UPDATE	**!txaftype(1,1)=0<CR>**	**OK txaftype(1,1)=0<CRLF>**
UPDATE	**!txaftype(*,0)={1,1}<CR>**	**OK txaftype(*,0)={1,1}<CRLF>**
UPDATE	**!txaftype(2,*)={1,0}<CR>**	**OK txaftype(1,1)=0<CRLF>**
UPDATE	**!txaftype(*,*)={1,1,0,0}<CR>**	**OK txaftype(*,*)={1,1,0,0}<CRLF>**

txrfenable (Transmitter RF carrier enable)

This command may be used as a query to read the transmitter RF carrier enable status or as an update to set it. The o set it. The transmitter is specified by using the address syntax. Addresses must be in the range 1 to 2. The data type is integer, either "1" meaning that the RF carrier is enabled, or "0" meaning that it is not. If the transmitter address is wildcarded, then the data type is an array of integer of size 2. In these cases the value 99 may be used in an update to indicate that a particular carrier enable is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	**!txrfenable(1)?<CR>**	**OK txrfenable(1)=1<CRLF>**
QUERY	**!txrfenable(*)?<CR>**	**OK txrfenable(*)={1,1}<CRLF>**
UPDATE	**!txrfenable(2)=0<CR>**	**OK txrfenable(2)=0<CRLF>**
UPDATE	**!txrfenable(*)={1,1}<CR>**	**OK txrfenable(*)={1,1}<CRLF>**

txrffreq (Transmitter RF carrier frequency)

This command may be used as a query to read the transmitter carrier frequency, or as an update to set it. The transmitter is specified by using the address syntax. Addresses must be in the range 1 to 2. The data type is integer, representing the frequency in kHz. If the transmitter address is wildcarded, then the data type is an array of integer of size 2. In this case the value **999999** may be used in an update to indicate that a particular transmitter frequency is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	**!txrffreq(1)?<CR>**	**OK txrffreq(1)=471200<CRLF>**

QUERY	**!txrffreq(*)?<CR>**	**OK txrffreq(*)={471200,520900}<CRLF>**
UPDATE	**!txrffreq(1)=471200<CR>**	**OK txrffreq(1)=471.200<CRLF>**
UPDATE	**!txrffreq(*)={471200,520900}<CR>**	**OK txrffreq(*)={471200,520900}<CRLF>**

txrfname (Transmitter RF carrier name)

This command may be used as a query to read the RF carrier name, or as an update to set it. The transmitter is specified by using the address syntax. Addresses must be in the range 1 to 2. The data type is string, with a limit of 15 characters.

Note: String arguments in commands need to be passed in **quoted** form, contained in a pair of **double-quote** (") characters. A problem arises when using the `inlb` command to read or write a string that already contains double-quote characters, for example: `The "Hula" Room`. The solution is to **escape** the double quotes within `The "Hula" Room` so that it can be passed as a string argument for the `inlb` command. This is done by preceding the double-quote characters with a **backslash** character like this: `The \"Hula\" Room`. Now it can be passed as a string argument to the `inlb` command: `inlb(1)="The \"Hula\" Room"`. Since the **backslash** serves as the escape character in quoted-string arguments, it too must be escaped if it is part of the string, so `"foo\bar"` would become `"foo\\bar".` If necessary, any character, printable or non-printable, can be represented in the hexadecimal escaped form `\xHH` where `HH` is any 2-digit hexadecimal number. The special escaped character forms `\r` (carriage return), `\n` (new line) and `\t` (tab) are also recognized..

Examples:

	REQUEST	RESPONSE
QUERY	**!txrfname(1)?<CR>**	**OK txrfname(1)="Vocalists"<CRLF>**
UPDATE	**!txrfname(2)="David"<CR>**	**OK txrfname(2)="David"<CRLF>**

txrfpwr (Transmitter RF carrier power)

This command may be used as a query to read the transmitter RF carrier power. The transmitter is specified by using the address syntax. Addresses must be in the range 1 to 2. The data type is integer and is a code that specifies the carrier power in mW. It may be in the range 0 to 2, with the following meanings:

Code	RF Carrier

	Power
0	10 mW
1	25 mW
2	50 mW

If the channel address is wildcarded, then the data type is an array of integer of size 2. In these cases the value 99 may be used in an update to indicate that a particular carrier power is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	**`!txrfpwr(1)?<CR>`**	**`OK txrfpwr(1)=2<CRLF>`**
QUERY	**`!txrfpwr(*)?<CR>`**	**`OK txrfpwr(*)={1,1}<CRLF>`**
UPDATE	**`!txrfpwr(2)=1<CR>`**	**`OK txrfpwr(1)=1<CRLF>`**
UPDATE	**`!txrfpwr(*)={2,2}<CR>`**	**`OK txrfpwr(*)={2,2}<CRLF>`**