ANIUSB-Matrix

Command Strings

ANIUSB-Matrix command strings for third-party control systems.
Version: 4.3 (2021-C)

Table of Contents

ANIUSB-Matrix Command Strings

ANIUSB Command Strings

The device is connected via Ethernet to a control system, such as AMX, Crestron or Extron.

Connection: Ethernet (TCP/IP; select “Client” in the AMX/Crestron program)
Port: 2202

Conventions

The device has 4 types of strings:

GET

Finds the status of a parameter. After the AMX/Crestron sends a GET command, the ANIUSB responds with a REPORT string

SET

Changes the status of a parameter. After the AMX/Crestron sends a SET command, the ANIUSB will respond with a REPORT string to indicate the new value of the parameter.

REP

When the ANIUSB receives a GET or SET command, it will reply with a REPORT command to indicate the status of the parameter. REPORT is also sent by the ANIUSB when a parameter is changed on the ANIUSB or through the GUI.

SAMPLE

Used for metering audio levels.

All messages sent and received are ASCII. Note that the level indicators and gain indicators are also in ASCII

Most parameters will send a REPORT command when they change. Thus, it is not necessary to constantly query parameters. The ANIUSB will send a REPORT command when any of these parameters change.

The character

“x”

in all of the following strings represents the channel of the ANIUSB and can be ASCII numbers 0 through 4 as in the following table

00	All channels
01-04	Dante Inputs
05	Analog Input
06	USB Input
07-08	Dante Outputs

09	Analog Output
10	USB Output

Command Strings (Common)

Get All	
Command String: `< GET xx ALL >`	*Where xx is ASCII channel number: 00 through 10. Use this command on first power on to update the status of all parameters.*
ANIUSB Response: `< REP ... >`	*The ANIUSB responds with individual Report strings for all parameters.*
Get Model Number	
Command String: `< GET MODEL >`	
ANIUSB Response: `< REP MODEL {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy is 32 characters of the model number. The ANIUSB always responds with a 32 character model number.*
Get Serial Number	
Command String: `< GET SERIAL_NUM >`	
ANIUSB Response: `< REP SERIAL_NUM {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy is 32 characters of the serial number. The ANIUSB always responds with a 32 character serial number.*
Get Channel Name	
Command String: `< GET xx CHAN_NAME >`	*Where xx is ASCII channel number: 00 through 10.*
ANIUSB Response: `< REP xx CHAN_NAME {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy is 31 characters of the channel name. The ANIUSB always responds with a 31 character name.*
Get Device ID	

Command String: `< GET DEVICE_ID >`	*The Device ID command does not contain the x channel character, as it is for the entire ANIUSB.*
ANIUSB Response: `< REP DEVICE_ID {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy is 31 characters of the device ID. The ANIUSB always responds with a 31 character device ID.*
Get Firmware Version	
Command String: `< GET FW_VER >`	
ANIUSB Response: `< REP FW_VER {yyyyyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyyyyy is 18 characters. The ANIUSB always responds with 18 characters.*
Get Preset	
Command String: `< GET PRESET >`	
ANIUSB Response: `< REP PRESET nn >`	*Where nn is the preset number 01-10. 0 = no preset active.*
Set Preset	
Command String: `< SET PRESET nn >`	*Where nn is the preset number 1-10. (Leading zero is optional when using the SET command).*

ANIUSB Response:
`< REP PRESET nn >`
`< REP ERR >`
	Where nn is the preset number 01-10.
	When user attempts to load an empty preset.

Get Preset Name	

Command String:
`< GET PRESET1 >`
`< GET PRESET2 >`
`< GET PRESET3 >`
etc
	Send one of these commands to the ANIUSB

ANIUSB Response:
`< REP PRESET1 {yyyyyyyyyyyyyyyyyyyyyyyyy} >`
`< REP PRESET2 {yyyyyyyyyyyyyyyyyyyyyyyyy} >`
`< REP PRESET3 {yyyyyyyyyyyyyyyyyyyyyyyyy} >`
etc
	Whereyyyyyyyyyyyyyyyyyyyyyyyyy is 25 characters of the device ID. The ANIUSB always responds with a 25 character device ID

Get Preset Audio Route	
Command String: `< GET PRESET_AUDIO_ROUTE >`	
ANIUSB Response: `< REP PRESET_AUDIO_ROUTE nn >`	*These presets store audio routing information. Where nn is the active routing preset number 01-10. 0 = no preset active.*
Set Preset Audio Route	
Command String: `< SET PRESET_AUDIO_ROUTE nn >`	*These presets store audio routing information. Where nn is the routing preset number 01-10.*
ANIUSB Response: `< REP PRESET_AUDIO_ROUTE nn >`	*Where nn is the active routing preset number 01-10. 0 = no preset active.*
Get Audio Gain	
Command String: `< GET xx AUDIO_GAIN_HI_RES >`	*Where xx is ASCII channel number: 00 through 10.*
ANIUSB Response: `< REP xx AUDIO_GAIN_HI_RES yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1400. yyyy is in steps of one-tenth of a dB.*
Set Audio Gain	
Command String: `< SET xx AUDIO_GAIN_HI_RES yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1400. yyyy is in steps of one-tenth of a dB.*
ANIUSB Response: `< REP xx AUDIO_GAIN_HI_RES yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1400.*
Increase Audio Gain by n dB	
Command String:	*Where nn is the amount in one-tenth of a dB to increase the gain. nn can be*

`< SET xx AUDIO_GAIN_HI_RES INC nn >`	*single digit (n), double digit (nn), triple digit (nnn).*
ANIUSB Response: `< REP xx AUDIO_GAIN_HI_RES yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1400.*
Decrease Audio Gain by n dB	
Command String: `< SET xx AUDIO_GAIN_HI_RES DEC nn >`	*Where nn is the amount in one-tenth of a dB to decrease the gain. nn can be single digit (n), double digit (nn), triple digit (nnn).*
ANIUSB Response: `< REP xx AUDIO_GAIN_HI_RES yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1400.*
Get Analog Input Gain Switch	
Command String: `< GET xx AUDIO_IN_LVL_SWITCH >`	*Where xx is ASCII channel number: 00 or 05.*

ANIUSB Response:
`< REP xx AUDIO_IN_LVL_SWITCH LINE_LVL >`
`< REP xx AUDIO_IN_LVL_SWITCH AUX_LVL >`
	The ANIUSB will respond with one of these strings.

Set Analog Input Gain Switch	

Command String:
`< SET xx AUDIO_IN_LVL_SWITCH LINE_LVL >`
`< SET xx AUDIO_IN_LVL_SWITCH AUX_LVL >`
	Where xx is ASCII channel number: 00 or 05. Send one of these commands to the ANIUSB

ANIUSB Response:
`< REP xx AUDIO_IN_LVL_SWITCH LINE_LVL >`
`< REP xx AUDIO_IN_LVL_SWITCH AUX_LVL >`
	The ANIUSB will respond with one of these strings.

Get Channel Audio Mute	
Command String: `< GET xx AUDIO_MUTE >`	*Where xx is ASCII channel number: 00 through 10.*

ANIUSB Response:
`< REP xx AUDIO_MUTE ON >`
`< REP xx AUDIO_MUTE OFF >`
	The ANIUSB will respond with one of these strings.

Mute Channel Audio	
Command String: `< SET xx AUDIO_MUTE ON >`	*Where xx is ASCII channel number: 00 through 10.*
ANIUSB Response: `< REP xx AUDIO_MUTE ON >`	*The ANIUSB will respond with one of these strings.*
Unmute Channel Audio	
Command String: `< SET xx AUDIO_MUTE OFF >`	*Where xx is ASCII channel number: 00 through 10.*
ANIUSB Response: `< REP xx AUDIO_MUTE OFF >`	*The ANIUSB will respond with one of these strings.*
Toggle Channel Audio Mute	
Command String: `< SET xx AUDIO_MUTE TOGGLE >`	

ANIUSB Response:
`< REP xx AUDIO_MUTE ON >`
`< REP xx AUDIO_MUTE OFF >`
	The ANIUSB will respond with one of these strings.

Get Device Audio Mute	
Command String: `< GET DEVICE_AUDIO_MUTE >`	

ANIUSB Response:
`< REP DEVICE_AUDIO_MUTE ON >`
`< REP DEVICE_AUDIO_MUTE OFF >`
	The ANIUSB will respond with one of these strings.

Set Device Audio Mute	

Command String:
`< SET DEVICE_AUDIO_MUTE ON >`
`< SET DEVICE_AUDIO_MUTE OFF >`
`< SET DEVICE_AUDIO_MUTE TOGGLE >`
	Send one of these commands to the ANIUSB.

ANIUSB Response:
`< REP DEVICE_AUDIO_MUTE ON >`
`< REP DEVICE_AUDIO_MUTE OFF >`
	The ANIUSB will respond with one of these strings.

Get Analog Output Gain Switch	
Command String: `< GET xx AUDIO_OUT_LVL_SWITCH >`	*Where xx is ASCII channel number: 00 or 09.*

ANIUSB Response:
`< REP xx AUDIO_OUT_LVL_SWITCH LINE_LVL >`
`< REP xx AUDIO_OUT_LVL_SWITCH AUX_LVL >`
`< REP xx AUDIO_OUT_LVL_SWITCH MIC_LVL >`
	The ANIUSB will respond with one of these strings.

Set Analog Output Gain Switch	

Command String:
`< SET xx AUDIO_OUT_LVL_SWITCH LINE_LVL >`
`< SET xx AUDIO_OUT_LVL_SWITCH AUX_LVL >`
`< SET xx AUDIO_OUT_LVL_SWITCH MIC_LVL >`
	Where xx is ASCII channel number: 00 or 09. Send one of these commands to the ANIUSB.

ANIUSB Response:
`< REP xx AUDIO_OUT_LVL_SWITCH LINE_LVL >`
`< REP xx AUDIO_OUT_LVL_SWITCH AUX_LVL >`
`< REP xx AUDIO_OUT_LVL_SWITCH MIC_LVL >`
	The ANIUSB will respond with one of these strings.

Flash Lights on ANIUSB	

Command String:
`< SET FLASH ON >`
`< SET FLASH OFF >`
	Send one of these commands to the ANIUSB. The flash automatically turns off after 30 seconds.

ANIUSB Response:
`< REP FLASH ON >`
`< REP FLASH OFF >`
	The ANIUSB will respond with one of these strings.

Turn Metering On	
Command String: `< SET METER_RATE sssss >`	*Turns metering on/off and sets rate. Where sssss is a value from 00000 to 99999 representing milliseconds.*

	00000= off *00100 = minimum value* *99999= maximum value* *Note: values 00001 to 00099 are not valid and result in response.*

ANIUSB Response:
`< REP METER_RATE sssss >`
`< SAMPLE aaa bbb ccc ddd eee fff ggg hhh iii jjj >`
	sssss = rate in milliseconds. Value 00000 means metering is off. Where the list that follows is the sample for each channel. Audio Levels take on values 000-060, which represent actual audio levels of -60 to 0 dBFS.
	aaa= channel 1 data
	bbb= channel 2 data
	ccc= channel 3 data
	ddd= channel 4 data
	eee = channel 5 data
	fff = channel 6 data
	ggg = channel 7 data
	hhh = channel 8 data
	iii = channel 9 data
	jjj = channel 10 data

Stop Metering	
Command String: `< SET METER_RATE 0 >`	*A value of 00000 is also acceptable.*
ANIUSB Response: `< REP METER_RATE 00000 >`	
Get LED Brightness	
Command String: `< GET LED_BRIGHTNESS >`	
ANIUSB Response: `< REP LED_BRIGHTNESS n >`	*Where n can take on the following values:*

	0 = LED disabled 1 = LED dim 2 = LED default
Set LED Brightness	
Command String: `< SET LED_BRIGHTNESS n >`	*Where n can take on the following values:* 0 = LED disabled 1 = LED dim 2 = LED default
ANIUSB Response: `< REP LED_BRIGHTNESS n >`	
Get Audio Clip Indicator	
Command String: `< GET xx AUDIO_OUT_CLIP_INDICATOR >`	*Where xx is ASCII channel number: 00 through 10.*

ANIUSB Response:
`< REP xx AUDIO_OUT_CLIP_INDICATOR ON >`
`< REP xx AUDIO_OUT_CLIP_INDICATOR OFF >`
	The ANIUSB will respond with one of these strings.

Get Audio IP Address	
Command String: `< GET IP_ADDR_NET_AUDIO_PRIMARY >`	
ANIUSB Response: `< REP IP_ADDR_NET_AUDIO_PRIMARY {yyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyy is a 15 digit IP address.*
Get Audio Subnet Address	
Command String: `< GET IP_SUBNET_NET_AUDIO_PRIMARY >`	
ANIUSB Response: `< REP IP_SUBNET_NET_AUDIO_PRIMARY {yyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyy is a 15 digit subnet address.*
Get Audio Gateway Address	

Command String: `< GET IP_GATEWAY_NET_AUDIO_PRIMARY >`	
ANIUSB Response: `< REP IP_GATEWAY_NET_AUDIO_PRIMARY {yyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyy is a 15 digit gateway address.*
Get Limiter Status	
Command String: `< GET xx LIMITER_ENGAGED >`	*Where xx is ASCII output channel number: 07 through 10. Indicates if the limiter is currently reducing the signal level.*

ANIUSB Response:
`< REP xx LIMITER_ENGAGED ON >`
`< REP xx LIMITER_ENGAGED OFF >`
	The ANIUSB will respond with one of these strings.

Get Encryption Status	
Command String: `< GET xx ENCRYPTION_CH >`	

ANIUSB Response:
`< REP xx ENCRYPTION_CH ON >`
`< REP xx ENCRYPTION_CH OFF >`
	The ANIUSB will respond with one of these strings.

Reboot ANIUSB	
Command String: `< SET REBOOT >`	
ANIUSB Response: `< REP REBOOT >`	
Get Error Events	
Command String: `< GET LAST_ERROR_EVENT >`	*Gets the last error that is loged on the ANIUSB.*
ANIUSB Response: `< REP LAST_ERROR_EVENT {yyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyy is up to 128 characters.*
Get PEQ Filter Enable	

Command String: `< GET xx PEQ yy >`	*Where xx is the PEQ block 07 or 10. Where yy is the PEQ filter 01-04 within the block. 00 can be used for all blocks or all filters.*

ANIUSB Response:
`< REP xx PEQ yy ON >`
`< REP xx PEQ yy OFF >`	

Set PEQ Filter Enable	

Command String:
`< SET xx PEQ yy ON >`
`< SET xx PEQ yy OFF >`
	Send one of these commands to the ANIUSB.

ANIUSB Response:
`< REP xx PEQ yy ON >`
`< REP xx PEQ yy OFF >`
	Where xx is the PEQ block 07 or 10. Where yy is the PEQ filter 01-04 within the block. 00 can be used for all blocks or all filters.

Get Input Meter Display Mode	
Command String: `< GET INPUT_METER_MODE >`	

ANIUSB Response:
`< REP INPUT_METER_MODE PRE_FADER >`
`< REP INPUT_METER_MODE POST_FADER >`
	The ANIUSB will respond with one of these strings.

Set Input Meter Display Mode	

Command String:
`< SET INPUT_METER_MODE PRE_FADER >`
`< SET INPUT_METER_MODE POST_FADER >`
	Send one of these commands to the ANIUSB.

ANIUSB Response:
`< REP INPUT_METER_MODE PRE_FADER >`
`< REP INPUT_METER_MODE POST_FADER >`
	The ANIUSB will respond with one of these strings.

Get Output Meter Display Mode	
Command String:	

`< GET OUTPUT_METER_MODE >`	

ANIUSB Response:
`< REP OUTPUT_METER_MODE PRE_FADER >`
`< REP OUTPUT_METER_MODE POST_FADER >`
	The ANIUSB will respond with one of these strings.

Set Output Meter Display Mode	

Command String:
`< SET OUTPUT_METER_MODE PRE_FADER >`
`< SET OUTPUT_METER_MODE POST_FADER >`
	Send one of these commands to the ANIUSB.

ANIUSB Response:
`< REP OUTPUT_METER_MODE PRE_FADER >`
`< REP OUTPUT_METER_MODE POST_FADER >`
	The ANIUSB will respond with one of these strings.

Get USB Connection Status	
Command String: `< GET USB_CONNECT >`	

ANIUSB Response:
`< REP USB_CONNECT ON >`
`< REP USB_CONNECT OFF >`
	The ANIUSB will respond with one of these strings.

Get Matrix Mixer Routing	
Command String: `< GET xx MATRIX_MXR_ROUTE yy >`	*Where xx is input channel numbers 00-06. Where yy is output channel numbers 00 or 07-10.*

ANIUSB Response:
`< REP xx MATRIX_MXR_ROUTE yy ON >`
`< REP xx MATRIX_MXR_ROUTE yy OFF >`
	The ANIUSB will respond with one of these strings.

Set Matrix Mixer Routing	

Command String:
`< SET xx MATRIX_MXR_ROUTE yy ON >`
`< SET xx MATRIX_MXR_ROUTE yy OFF >`
	Where xx is input channel numbers 00-06. Where yy is output channel numbers 00 or 07-10. Send one of these commands to the ANIUSB.

ANIUSB Response:	*The ANIUSB will respond with one of these strings.*

`< REP xx MATRIX_MXR_ROUTE yy ON >`
`< REP xx MATRIX_MXR_ROUTE yy OFF >`	

Get Matrix Mixer Gain	
Command String: `< GET xx MATRIX_MXR_GAIN yy >`	*Where xx is input channel numbers 00-06. Where yy is output channel numbers 00 or 07-10.*
ANIUSB Response: `< REP xx MATRIX_MXR_GAIN yyzzzz >`	*Where zzzz takes on the ASCII values of 0000 to 1400. zzzz is in steps of one-tenth of a dB.*
Set Matrix Mixer Gain	
Command String: `< SET xx MATRIX_MXR_GAIN yyzzzz >`	*Where xx is input channel numbers 00-06. Where yy is output channel numbers 00 or 07-10. Where zzzz takes on the ASCII values of 0000 to 1400. zzzz is in steps of one-tenth of a dB.*
ANIUSB Response: `< REP xx MATRIX_MXR_GAIN yyzzzz >`	
Increment Matrix Mixer Gain	
Command String: `< SET xx MATRIX_MXR_GAIN yy INC nn >`	*Where xx is input channel numbers 00-06. Where yy is output channel numbers 00 or 07-10. Where nn is in steps of one-tenth of a dB.*
ANIUSB Response: `< REP xx MATRIX_MXR_GAIN yyzzzz >`	*Where zzzz takes on the ASCII values of 0000 to 1400. zzzz is in steps of one-tenth of a dB.*
Decrement Matrix Mixer Gain	
Command String: `< SET xx MATRIX_MXR_GAIN yy DEC nn >`	*Where xx is input channel numbers 00-06. Where yy is output channel numbers 00 or 07-10. Where nn is in steps of one-tenth of a dB.*
ANIUSB Response: `< REP xx MATRIX_MXR_GAIN yyzzzz >`	*Where zzzz takes on the ASCII values of 0000 to 1400. zzzz is in steps of one-tenth of a dB.*
Get or Set Logic Mute	
Command String: `< SET LOGIC_MUTE sts >`	sts is the requested gate out signal for the channel: 1. Muted

	2. Unmuted
ANIUSB Response: `< REP LOGIC_MUTE sts >`	
Enable or Disable Call Status	

Command String:
`< GET ONHOOK_ENABLE >`
`< SET ONHOOK_ENABLE state >`
	Enables or disables Call status feature. state can be:
	1. ON
	2. OFF

ANIUSB Response: `< REP ONHOOK_ENABLE state >`	
Report Call Status State	
Command String: `< GET ONHOOK_STATE >`	
ANIUSB Response: `< REP ONHOOK_STATE state >`	When Call status is enabled, state can be: 1. ONHOOK (not in a call) 2. OFFHOOK (in a call) When Call status is disabled, state is OFFHOOK.