MXA910

Command Strings

MXA910 command strings for control systems, such as Crestron or Extron.

Version: 3 (2019-F)

Table of Contents

MXA910
Command Strings

Using a Third-Party Control System

The microphone receives logic commands over the network. Many parameters controlled through the web application can be controlled through a third party control system, using the appropriate command string.

Common applications:

- Mute
- LED color and behavior
- Loading presets
- Adjusting levels

A complete list of command strings is available in the device help or from www.shure.com.

MXA910 Microflex® Advance™ 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 MXA910 responds with a REPORT string

SET

Changes the status of a parameter. After the AMX/Crestron sends a SET command, the MXA910 will respond with a REPORT string to indicate the new value of the parameter.

REP

When the MXA910 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 device when a parameter is changed on the MXA910 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 MXA910 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 MXA910 and can be ASCII numbers 0 through 9 as in the following table.

0	All channels
1 through 8	Individual channels
9	Automix output
10	**Firmware 3.x:** Echo Reduction reference channel **Firmware 4.x:** AEC reference channel

Command Strings (Common)

Get All	
Command String: `< GET x ALL >`	*Where x is ASCII channel number: 0 through 9. Use this command on first power on to update the status of all parameters.*
MXA910 Response: `< REP ... >`	*The MXA910 responds with individual Report strings for all parameters.*
Get Model Number	
Command String: `< GET MODEL >`	
MXA910 Response: `< REP MODEL {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy is 32 characters of the model number. The MXA910 always responds with a 32 character model number.*
Get Serial Number	
Command String: `< GET SERIAL_NUM >`	
MXA910 Response: `< REP SERIAL_NUM {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy is 32 characters of the serial number. The MXA910 always responds with a 32 character serial number.*

Get Firmware Version	
Command String: `< GET FW_VER >`	
MXA910 Response: `< REP FW_VER {yyyyyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyyyyy is 18 characters. The MXA910 always responds with 18 characters.*
Get Audio IP Address	
Command String: `< GET IP_ADDR_NET_AUDIO_PRIMARY >`	
MXA910 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 >`	
MXA910 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 >`	
MXA910 Response: `< REP IP_GATEWAY_NET_AUDIO_PRIMARY {yyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyy is a 15 digit gateway address.*
Get Channel Name	
Command String: `< GET x CHAN_NAME >`	*Where x is ASCII channel number: 0 through 9.*
MXA910 Response: `< REP x CHAN_NAME {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy is 31 characters of the channel name. The MXA910 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 device.*
MXA910 Response: `< REP DEVICE_ID {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy is 31 characters of the device ID. The MXA910 always responds with a 31 character device ID.*
Get Audio Gain	
Command String: `< GET x AUDIO_GAIN_HI_RES >`	*Where x is ASCII channel number: 1 through 9. Channel number 0 (all channels) is not valid for this command.*
MXA910 Response: `< REP x 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 x 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.*
MXA910 Response: `< REP x AUDIO_GAIN_HI_RES yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1400.*
Increase Audio Gain by n dB	
Command String: `< SET x AUDIO_GAIN_HI_RES INC nn >`	*Where nn is the amount in one-tenth of a dB to increase the gain. nn can be single digit (n), double digit (nn), triple digit (nnn).*
MXA910 Response: `< REP x 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 x 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).*
MXA910 Response: `< REP x AUDIO_GAIN_HI_RES yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1400.*
Get Post-Gate Audio Gain (firmware > v3.0)	

Command String: `< GET x AUDIO_GAIN_POSTGATE >`	*Where x is ASCII channel number: 1 through 8. Channel number 0 (all channels) is not valid for this command.*
MXA910 Response: `< REP x AUDIO_GAIN_POSTGATE yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1400. yyyy is in steps of one-tenth of a dB.*
Set Post-Gate Audio Gain (firmware > v3.0)	
Command String: `< SET x AUDIO_GAIN_POSTGATE yyyy >`	*Where x is ASCII channel number: 1 through 8. Where yyyy takes on the ASCII values of 0000 to 1400. yyyy is in steps of one-tenth of a dB.*
MXA910 Response: `< REP x AUDIO_GAIN_POSTGATE yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1400.*
Increase Post-Gate Audio Gain by n dB (firmware > v3.0)	
Command String: `< SET x AUDIO_GAIN_POSTGATE INC nn >`	*Where x is ASCII channel number: 1 through 8. Where nn is the amount in one-tenth of a dB to increase the gain. nn can be single digit (n), double digit (nn), triple digit (nnn).*
MXA910 Response: `< REP x AUDIO_GAIN_POSTGATE yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1400.*
Decrease Post-Gate Audio Gain by n dB (firmware > v3.0)	
Command String: `< SET x AUDIO_GAIN_POSTGATE DEC nn >`	*Where x is ASCII channel number: 1 through 8. 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).*
MXA910 Response: `< REP x AUDIO_GAIN_POSTGATE yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1400.*
Get Echo Reduction Reference Channel Audio Gain (firmware v3.0 only)	
Command String: `< GET AUDIO_GAIN_ECHO_RED >`	
MXA910 Response: `< REP AUDIO_GAIN_ECHO_RED yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1400 representing gain from -109.9 dB to 30.0 dB. yyyy is in steps of one-tenth of a dB .*

Set Echo Reduction Reference Channel Audio Gain (firmware v3.0 only)	
Command String: `< SET AUDIO_GAIN_ECHO_RED yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1400 representing gain from -109.9 dB to 30.0 dB. yyyy is in steps of one-tenth of a dB.*
MXA910 Response: `< REP AUDIO_GAIN_ECHO_RED yyyy >`	
Increase Echo Reduction Reference Channel Audio Gain (firmware v3.0 only)	
Command String: `< SET AUDIO_GAIN_ECHO_RED INC yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1400 representing gain from -109.9 dB to 30.0 dB. yyyy is in steps of one-tenth of a dB. The resulting gain must be within the allowed range.*
MXA910 Response: `< REP AUDIO_GAIN_ECHO_RED yyyy >`	
Decrease Echo Reduction Reference Channel Audio Gain (firmware v3.0 only)	
Command String: `< SET AUDIO_GAIN_ECHO_RED DEC yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1400 representing gain from -109.9 dB to 30.0 dB. yyyy is in steps of one-tenth of a dB. The resulting gain must be within the allowed range.*
MXA910 Response: `< REP AUDIO_GAIN_ECHO_RED yyyy >`	
Get Echo Reduction Level (firmware v3.0 only)	
Command String: `< GET ECHO_RED >`	
MXA910 Response: `< REP ECHO_RED sts >`	*Where sts is the desired Echo Reduction state:* *OFF* *SOFT* *MED* *HARD*
Set Echo Reduction Level (firmware v3.0 only)	

Command String: `< SET ECHO_RED sts >`	*Where sts is the desired Echo Reduction state:* *OFF* *SOFT* *MED* *HARD*
MXA910 Response: `< REP ECHO_RED sts >`	
Get Channel Audio Mute	
Command String: `< GET x AUDIO_MUTE >`	*Where x is ASCII channel number: 0 through 9. Channel Audio Mute is pre-meter*
MXA910 Response: `< REP x AUDIO_MUTE ON >` `< REP x AUDIO_MUTE OFF >`	*The MXA910 will respond with one of these strings.*
Mute Channel Audio	
Command String: `< SET x AUDIO_MUTE ON >`	
MXA910 Response: `< REP x AUDIO_MUTE ON >`	
Unmute Channel Audio	
Command String: `< SET x AUDIO_MUTE OFF >`	
MXA910 Response: `< REP x AUDIO_MUTE OFF >`	
Toggle Channel Audio Mute	
Command String: `< SET x AUDIO_MUTE TOGGLE >`	
MXA910 Response: `< REP x AUDIO_MUTE ON >`	*The MXA910 will respond with one of these strings.*

`< REP x AUDIO_MUTE OFF >`	
Get Device Audio Mute	
Command String: `< GET DEVICE_AUDIO_MUTE >`	*Device Audio Mute is post-meter.*
MXA910 Response: `< REP DEVICE_AUDIO_MUTE ON >` `< REP DEVICE_AUDIO_MUTE OFF >`	*The MXA910 will respond with one of these strings.*
Mute Device Audio	
Command String: `< SET DEVICE_AUDIO_MUTE ON >`	
MXA910 Response: `< REP DEVICE_AUDIO_MUTE ON >`	
Unmute Device Audio	
Command String: `< SET DEVICE_AUDIO_MUTE OFF >`	
MXA910 Response: `< REP DEVICE_AUDIO_MUTE OFF >`	
Toggle Device Audio Mute	
Command String: `< SET DEVICE_AUDIO_MUTE TOGGLE >`	
MXA910 Response: `< REP DEVICE_AUDIO_MUTE ON >` `< REP DEVICE_AUDIO_MUTE OFF >`	*The MXA910 will respond with one of these strings.*
Get Output Clip Status	
Command String: `< GET x AUDIO_OUT_CLIP_INDICATOR >`	*Where x is ASCII channel number: 0 through 9. It is not necessary to continually send this command. The MXA910 will send a REPORT message whenever the status changes.*

MXA910 Response: `< REP x AUDIO_OUT_CLIP_INDICATOR ON >` `< REP x AUDIO_OUT_CLIP_INDICATOR OFF >`	*The MXA910 will respond with one of these strings.*
Flash Lights on Microphone	
Command String: `< SET FLASH ON >` `< SET FLASH OFF >`	*Send one of these commands to the MXA910. The flash automatically turns off after 30 seconds.*
MXA910 Response: `< REP FLASH ON >` `< REP FLASH OFF >`	*The MXA910 will respond with one of these strings.*
Get Metering Rate	
Command String: `< GET METER_RATE >`	
MXA910 Response: `< REP METER_RATE ssssss >`	*Where ssssss is the metering speed in milliseconds. Setting ssssss= 0 turns metering off. Minimum setting is 100 milliseconds, maximum is 99999. Metering is off by default.*
Turn Metering On	
Command String: `< SET METER_RATE ssssss >`	*Where ssssss is the metering speed in milliseconds. Setting ssssss= 0 turns metering off. Minimum setting is 100 milliseconds, maximum is 99999. Metering is off by default.*
MXA910 Response: `< REP METER_RATE ssssss >` `< SAMPLE aaa bbb ccc ddd eee fff ggg hhh iii >`	*Where aaa, bbb, etc is the value of the audio level received. Levels take on values 000-060, which represent actual audio levels of -60 to 0 dBFS (value of 000 equals -60 dBFS).* *aaa= output 1* *bbb= output 2* *ccc= output 3* *ddd= output 4* *eee= output 5*

	fff= output 6 *ggg= output 7* *hhh= output 8* *iii= output 9*
Stop Metering	
Command String: `< SET METER_RATE 0 >`	*A value of 00000 is also acceptable.*
MXA910 Response: `< REP METER_RATE 00000 >`	
Get Post-Gate Metering Rate (firmware > v3.0)	
Command String: `< GET METER_RATE_POSTGATE >`	
MXA910 Response: `< REP METER_RATE_POSTGATE sssss >`	*Where sssss is the metering rate in milliseconds. Setting sssss= 0 turns metering off.*
Set Post-Gate Metering Rate (firmware > v3.0)	
Command String: `< SET METER_RATE_POSTGATE sssss >`	*Where sssss is a value from 0 to 99999 in milliseconds.* *0 = Off* *100 = Minimum value* *99999 = Maximum value*
MXA910 Response: `< SAMPLE aaa bbb ccc ddd eee fff ggg hhh >`	*Where aaa, bbb, etc is the value of the audio level received and is 000-060.* *aaa= output 1* *bbb= output 2* *ccc= output 3* *ddd= output 4* *eee= output 5* *fff= output 6* *ggg= output 7* *hhh= output 8*

Get Automixer Gain Metering Rate (firmware > v3.0)	
Command String: `< GET METER_RATE_MXR_GAIN >`	
MXA910 Response: `< REP METER_RATE_MXR_GAIN sssss >`	*Where sssss is the metering rate in milliseconds. Setting sssss= 0 turns metering off.*
Set Automixer Gain Metering Rate (firmware > v3.0)	
Command String: `< SET METER_RATE_MXR_GAIN sssss >`	*Where ssssssis a value from 0 to 99999 in milliseconds.* *0 = Off* *100 = Minimum value* *99999 = Maximum value*
MXA910 Response: `< SAMPLE aaa bbb ccc ddd eee fff ggg hhh >`	*Where aaa, bbb, etc is the value of the audio level received and is 000-060.* *aaa= output 1* *bbb= output 2* *ccc= output 3* *ddd= output 4* *eee= output 5* *fff= output 6* *ggg= output 7* *hhh= output 8*
Get or Set Pre-Compressor Metering Rate (firmware > v4.0)	
Command String: `< GET METER_RATE_PRECOMP >` `< SET METER_RATE_PRECOMP sssss >`	*Where sssss is a value from 0 to 99999 in milliseconds.* *0 = Off* *100 = Minimum value* *99999 = Maximum value*
MXA910 Response: `< REP METER_RATE_PRECOMP sssss >` `< SAMPLE aaa >`	*Where aaa is the value of the audio level received . Levels take on values 000-060, which represent actual audio levels of -60 to 0 dBFS (value of 000 equals -60 dBFS).*

	aaa = output 9
Get or Set AEC Reference Signal Metering Rate (firmware > v4.0)	
Command String: `< GET METER_RATE_AECREF >` `< SET METER_RATE_AECREF sssss >`	*Where sssss is a value from 0 to 99999 in milliseconds.* *0 = Off* *100 = Minimum value* *99999 = Maximum value*
MXA910 Response: `< REP METER_RATE_AECREF sssss >` `< SAMPLE aaa >`	*Where aaais the value of the audio level received . Levels take on values 000-060, which represent actual audio levels of -60 to 0 dBFS (value of 000 equals -60 dBFS).* *aaa = output 10*
Get Audio Peak Level	
Command String: `< GET x AUDIO_IN_PEAK_LVL >`	
MXA910 Response: `< REP x AUDIO_IN_PEAK_LVL nnn >`	*Where nnn is the audio level and is 000-060.*
Get Audio RMS Level	
Command String: `< GET x AUDIO_IN_RMS_LVL >`	
MXA910 Response: `< REP x AUDIO_IN_RMS_LVL nnn >`	*Where nnn is the audio level and is 000-060.*
Get Preset	
Command String: `< GET PRESET >`	
MXA910 Response: `< REP PRESET nn >`	*Where nn is the preset number 01-10.*
Set Preset	

Command String: `< SET PRESET nn >`	*Where nn is the preset number 1-10. (Leading zero is optional when using the SET command).*
MXA910 Response: `< REP PRESET nn >`	*Where nn is the preset number 01-10.*
Get Preset Name	
Command String: `< GET PRESET1 >` `< GET PRESET2 >` `< GET PRESET3 >` **etc**	*Send one of these strings to the MXA910.*
MXA910 Response: `< REP PRESET1 {yyyyyyyyyyyyyyyyyyyyyyyyy} >` `< REP PRESET2 {yyyyyyyyyyyyyyyyyyyyyyyyy} >` `< REP PRESET3 {yyyyyyyyyyyyyyyyyyyyyyyyy} >` **etc**	*Where yyyyyyyyyyyyyyyyyyyyyyyyy is 25 characters of the preset name. The MXA910 always responds with a 25 character preset name*
Get Gate Out Status	
Command String: `< GET x AUTOMIX_GATE_OUT_EXT_SIG >`	*Where x is ASCII channel number: 0 through 8. It is not necessary to continually send this command. The MXA910 will send a REPORT message whenever the status changes.*
MXA910 Response: `< REP x AUTOMIX_GATE_OUT_EXT_SIG ON >` `< REP x AUTOMIX_GATE_OUT_EXT_SIG OFF >`	*The MXA910 will respond with one of these strings.*
Get LED State	
Command String: `< GET DEV_LED_IN_STATE >`	
MXA910 Response: `< REP DEV_LED_IN_STATE OFF >` `< REP DEV_LED_IN_STATE ON >`	*The MXA910 will respond with one of these strings.*

Set LED State	
Command String: `< SET DEV_LED_IN_STATE OFF >` `< SET DEV_LED_IN_STATE ON >`	*Send one of these commands to the MXA910.*
MXA910 Response: `< REP DEV_LED_IN_STATE OFF >` `< REP DEV_LED_IN_STATE ON >`	*The MXA910 will respond with one of these strings.*
Get LED Brightness	
Command String: `< GET LED_BRIGHTNESS >`	
MXA910 Response: `< REP LED_BRIGHTNESS n >`	*Where n can take on the following values:* 0 = LED disabled 1 = LED dim 2 = LED default **Firmware > v3.0:** 0 = LED disabled 1 = 20% 2 = 40% 3 = 60% 4 = 80% 5 = 100%
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 **Firmware > v3.0:** 0 = LED disabled

	1 = 20% 2 = 40% 3 = 60% 4 = 80% 5 = 100%
MXA910 Response: `< REP LED_BRIGHTNESS n >`	
Get LED Mute Color	
Command String: `< GET LED_COLOR_MUTED >`	
MXA910 Response: `< REP LED_COLOR_MUTED nnnn >`	*Where nnnn can be RED, GREEN, BLUE, PINK, PURPLE, YELLOW, ORANGE, or WHITE.* **Firmware > v3.0:** *Where nnnn can be RED, GREEN, BLUE, PINK, PURPLE, YELLOW, ORANGE, WHITE, GOLD, YELLOWGREEN, TURQUOISE, POWDERBLUE, CYAN, SKYBLUE, LIGHTPURPLE, VIOLET, or ORCHID.*
Set LED Mute Color	
Command String: `< SET LED_COLOR_MUTED nnnn >`	*Where nnnn can be RED, GREEN, BLUE, PINK, PURPLE, YELLOW, ORANGE, or WHITE.* **Firmware > v3.0:** *Where nnnn can be RED, GREEN, BLUE, PINK, PURPLE, YELLOW, ORANGE, WHITE, GOLD, YELLOWGREEN, TURQUOISE, POWDERBLUE, CYAN, SKYBLUE, LIGHTPURPLE, VIOLET, or ORCHID.*
MXA910 Response: `< REP LED_COLOR_MUTED nnnn >`	
Get LED Unmute Color	
Command String: `< GET LED_COLOR_UNMUTED >`	

MXA910 Response: `< REP LED_COLOR_UNMUTED nnnn >`	*Where nnnn can be RED, GREEN, BLUE, PINK, PURPLE, YELLOW, ORANGE, or WHITE.* **Firmware > v3.0:** *Where nnnn can be RED, GREEN, BLUE, PINK, PURPLE, YELLOW, ORANGE, WHITE, GOLD, YELLOWGREEN, TURQUOISE, POWDERBLUE, CYAN, SKYBLUE, LIGHTPURPLE, VIOLET, or ORCHID.*
Set LED Unmute Color	
Command String: `< SET LED_COLOR_UNMUTED nnnn >`	*Where nnnn can be RED, GREEN, BLUE, PINK, PURPLE, YELLOW, ORANGE, or WHITE.* **Firmware > v3.0:** *Where nnnn can be RED, GREEN, BLUE, PINK, PURPLE, YELLOW, ORANGE, WHITE, GOLD, YELLOWGREEN, TURQUOISE, POWDERBLUE, CYAN, SKYBLUE, LIGHTPURPLE, VIOLET, or ORCHID.*
MXA910 Response: `< REP LED_COLOR_UNMUTED nnnn >`	
Get LED Mute Behavior	
Command String: `< GET LED_STATE_MUTED >`	
MXA910 Response: `< REP LED_STATE_MUTED nnn >`	*Where nnn can be ON, OFF, or FLASHING*
Set LED Mute Behavior	
Command String: `< SET LED_STATE_MUTED nnn >`	*Where nnn can be ON, OFF, or FLASHING*
MXA910 Response: `< REP LED_STATE_MUTED nnn >`	
Get LED Unmute Behavior	
Command String:	

`< GET LED_STATE_UNMUTED >`	
MXA910 Response: `< REP LED_STATE_UNMUTED nnn >`	*Where nnn can be ON, OFF, or FLASHING*
Set LED Unmute Behavior	
Command String: `< SET LED_STATE_UNMUTED nnn >`	*Where nnn can be ON, OFF, or FLASHING*
MXA910 Response: `< REP LED_STATE_UNMUTED nnn >`	
Get X-Axis Beam (Lobe) Steering	
Command String: `< GET x BEAM_X >`	*Where the X-Axis is parallel with the Shure logo.*
MXA910 Response: `< REP x BEAM_X nnnn >`	*Where nnnn is 0000-3048 in centimeters. The value 1524 is the centerline of the MXA910.*
Set X-Axis Beam (Lobe) Steering	
Command String: `< SET x BEAM_X nnnn >`	*Where nnnn is 0000-3048 in centimeters. The value 1524 is the centerline of the MXA910.*
MXA910 Response: `< REP x BEAM_X nnnn >`	
Get Y-Axis Beam (Lobe) Steering	
Command String: `< GET x BEAM_Y >`	*Where the Y-Axis is perpendicular to the X-Axis.*
MXA910 Response: `< REP x BEAM_Y nnnn >`	*Where nnnn is 0000-3048 in centimeters. The value 1524 is the centerline of the MXA910.*
Set Y-Axis Beam (Lobe) Steering	
Command String: `< SET x BEAM_Y nnnn >`	*Where nnnn is 0000-3048 in centimeters. The value 1524 is the centerline of the MXA910.*
MXA910 Response:	

`< REP x BEAM_Y nnnn >`	
Get Beam (Lobe) Height	
Command String: `< GET x BEAM_Z >`	*Where height is the distance down from the MXA910.*
MXA910 Response: `< REP x BEAM_Z nnn >`	*Where nnn is 000-914 in centimeters.*
Set Beam (Lobe) Height	
Command String: `< SET x BEAM_Z nnn >`	*Where nnn is 000-914 in centimeters.*
MXA910 Response: `< REP x BEAM_Z nnn >`	
Get Beam (Lobe) Width	
Command String: `< GET x BEAM_W >`	
MXA910 Response: `< REP x BEAM_W nnnn >`	*Where nnnn can be WIDE, MEDIUM, or NARROW*
Set Beam (Lobe) Width	
Command String: `< SET x BEAM_W nnnn >`	*Where nnnn can be WIDE, MEDIUM, or NARROW*
MXA910 Response: `< REP x BEAM_W nnnn >`	
Reboot MXA910 (firmware > v2.0)	
Command String: `< SET REBOOT >`	
MXA910 Response:	*The MXA910 does not send a response for this command*
Get Error Events (firmware > v2.0)	
Command String:	

`< GET LAST_ERROR_EVENT >`	
MXA910 Response: `< REP LAST_ERROR_EVENT {yyyyy} >`	*Where yyyy can be up to 128 characters.*
Get Low Shelf Filter (firmware v2.0 only)	
Command String: `< GET LOW_SHELF_FILTER >`	
MXA910 Response: `< REP LOW_SHELF_FILTER ON >` `< REP LOW_SHELF_FILTER OFF >`	*The MXA910 will respond with one of these strings.*
Set Low Shelf Filter (firmware v2.0 only)	
Command String: `< SET LOW_SHELF_FILTER ON >` `< SET LOW_SHELF_FILTER OFF >` `< SET LOW_SHELF_FILTER TOGGLE >`	*Send on of these commands to the MXA910*
MXA910 Response: `< REP LOW_SHELF_FILTER ON >` `< REP LOW_SHELF_FILTER OFF >`	*The MXA910 will respond with one of these strings.*
Command String: `< SET LOW_SHELF_FILTER >`	
MXA910 Response: `< REP LOW_SHELF_FILTER ON >` `< REP LOW_SHELF_FILTER OFF >`	*The MXA910 will respond with one of these strings.*
Get Bypass All EQ (firmware > v3.0)	
Command String: `< GET BYPASS_ALL_EQ >`	
MXA910 Response: `< REP BYPASS_ALL_EQ sts >`	*Where sts can be:* *ON* *OFF*
Set Bypass All EQ (firmware > v3.0)	

Command String: `< SET BYPASS_ALL_EQ sts >`	*Where sts can be:* *ON* *OFF* *TOGGLE*
MXA910 Response: `< REP BYPASS_ALL_EQ sts >`	*Where sts can be:* *ON* *OFF*
Get EQ Contour (firmware > v3.0)	
Command String: `< GET EQ_CONTOUR >`	
MXA910 Response: `< REP EQ_CONTOUR sts >`	*Where sts takes on a value 1-4 to indicate:* *1 = OFF* *2 = HIGH PASS* *3 = LOW SHELF* *4 = MULTI-BAND*
Set EQ Contour (firmware > v3.0)	
Command String: `< SET EQ_CONTOUR sts >`	*Where sts takes on a value 1-4 to indicate:* *1 = OFF* *2 = HIGH PASS* *3 = LOW SHELF* *4 = MULTI-BAND*
MXA910 Response: `< REP EQ_CONTOUR sts >`	
Get or Set Bypass IntelliMix DSP (firmware > v4.0)	
Command String: `< GET BYPASS_IMX >` `< SET BYPASS_IMX sts >`	*Allows you to control the Bypass IntelliMix feature. Turning Bypass IntelliMix on will turn off these DSP blocks:* *AEC* *Noise Reduction* *Compressor* *Delay* *AGC*

	Where sts can be: *ON* *OFF* *TOGGLE*
`< REP BYPASS_IMX sts >`	
Get Network Audio Device Name	
Command String: `< GET NA_DEVICE_NAME >`	
MXA910 Response: `< REP NA_DEVICE_NAME {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} >`	*Where {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} is a text string. Most devices allow device id to be up to 31characters. Value is padded with spaces as needed to ensure that 31 char are always reported.*
Get Network Audio Channel Name	
Command String: `< GET xx NA_CHAN_NAME >`	*Where xx is channel number All channels: 0 MXA910: 1-9, 9 being automix channel*
MXA910 Response: `< REP xx NA_CHAN_NAME {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} >`	*Where xx is channel number. Where {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} is 31 char channel name. Value is padded with spaces as needed to ensure that 31 char are always reported.*
Get Control Network MAC Address	
Command String: `< GET CONTROL_MAC_ADDR >`	
MXA910 Response: `< REP CONTROL_ MAC_ADDR yy:yy:yy:yy:yy:yy >`	*Where yy:yy:yy:yy:yy:yy is a 17 char literal string formatted as 6 octets, each separated by a colon. Example: 00:0E:DD:FF:F1:63*
Get Device Mute LED State	
Command String: `< GET DEV_MUTE_STATUS_LED_STATE >`	
MXA910 Response: `< REP DEV_MUTE_STATUS_LED_STATE ON >`	*ON = MUTED OFF = UNMUTED*

`< REP DEV_MUTE_STATUS_LED_STATE OFF >`	
Restore Default Settings (firmware > v2.0)	
Command String: `< SET DEFAULT_SETTINGS >`	*Request the device to set itself to default settings.*
MXA910 Response: `< REP PRESET xx >`	*where xx = 00 if restore is successful*
Get Active Mic Channels	
Command String: `< GET NUM_ACTIVE_MICS >`	
MXA910 Response: `< REP NUM_ACTIVE_MICS x >`	*where x is number of active channels that takes on values: MXA910: channels 1-8*
Get PEQ Filter Enable (firmware > v2.0)	
Command String: `< GET xx PEQ yy >`	*Where xx is the PEQ block 01-04. Where yy is the PEQ filter 01-04 within the block. 00 can be used for all blocks or all filters.*
MXA910 Response: `< REP xx PEQ yy ON >` `< REP xx PEQ yy OFF >`	
Set PEQ Filter Enable (firmware > v2.0)	
Command String: `< SET xx PEQ yy ON >` `< SET xx PEQ yy OFF >`	*Send one of these commands to the MXA910.*
MXA910 Response: `< REP xx PEQ yy ON >` `< REP xx PEQ yy OFF >`	*Where xx is the PEQ block 01-04. Where yy is the PEQ filter 01-04 within the block. 00 can be used for all blocks or all filters.*
Get Automix Channel Solo Enable	
Command String: `< GET x CHAN_AUTOMIX_SOLO_EN >`	*where x is channel number: 0 is not valid MXA910: channels 1-8*

MXA910 Response: `< REP x CHAN_AUTOMIX_SOLO_EN ENABLE >` `< REP x CHAN_AUTOMIX_SOLO_EN DISABLE >`	*where x is channel number: 0 is not valid MXA910: channels 1-8; where sts indicates channel x's SOLO state: ENABLE DISABLE*
Set Automix Channel Solo Enable	
Command String: `< SET x CHAN_AUTOMIX_SOLO_EN ENABLE >` `< SET x CHAN_AUTOMIX_SOLO_EN DISABLE >`	*where x is channel number: 0 is not valid MXA910: channels 1-8; where sts determines the requested state of SOLO mode: ENABLE DISABLE*
MXA910 Response: `< REP x CHAN_AUTOMIX_SOLO_EN ENABLE >` `< REP x CHAN_AUTOMIX_SOLO_EN DISABLE >`	*where x is channel number: 0 is not valid MXA910: channels 1-8; where sts indicates channel x's SOLO state: ENABLE DISABLE*
Get Encryption Status (firmware > v2.0)	
Command String: `< GET ENCRYPTION >`	*Get device level encryption status;*
MXA910 Response: `< REP ENCRYPTION ON >` `< REP ENCRYPTION OFF >`	*Send one of these commands to the MXA910.*
Get Flash	
Command String: `< GET FLASH >`	
MXA910 Response: `< REP FLASH ON >` `< REP FLASH OFF >`	*The MXA910 will respond with one of these commands.*
Set Flash	
Command String: `< SET FLASH ON >` `< SET FLASH OFF >`	*Send one of these commands to the MXA910.*
MXA910 Response: `< REP FLASH ON >`	*The MXA910 will respond with one of these commands.*

`< REP FLASH OFF >`	