

# NLED Aurora Magician Protocol Manual – v.3

*Please also read the NLED Aurora Control Manual for definitions and more info.*

This document covers the NLED command protocol. These commands can be issued to compatible NLED controllers and devices to initiate functions. These functions include things like Sequence Change, Color Swap, Device Identification, and more. Native USB devices will accept commands over emulated USB serial port, the USB serial port can be accessed by any software or language that can access standards serial ports. Such as serial terminals, Processing, C languages, .NET, Java, etc. Some USB controllers offer the “Dual Command mode” hardware configuration, that allows the controller to accept commands over it's UART from devices such as an FTDI, ESP8266, X-Bee, or other serial based transceiver at any baud rate, but always 8-N-1.

**Sending Commands to a NLED Device:** Contact [Support@NLEDshop.com](mailto:Support@NLEDshop.com) for assistance.

**NOTE: Command requests are sent and received as ASCII characters. The command and data bytes are sent as numbers(8-bit).**

## To Request a Command:

Host Sends (ASCII Characters):	“NLED11”	As Hex: 0x4E -> 0x4C -> 0x45 -> 0x44 -> 0x31 -> 0x31
Device Responds(ASCII Characters):	“a9”	As Hex: 0x61 -> 0x39      As Decimal: 97 -> 57
Host Sends(ASCII Characters):	“nled99”	As Hex: 0x6E -> 0x6C -> 0x65 -> 0x64 -> 0x39 -> 0x39
Device Responds(ASCII Characters):	“f0”	As Hex: 0x66 -> 0x30      As Decimal: 102 -> 48

The device is now ready to receive the command.

## To Send a Command Once Request is Acknowledged:

Host Sends(As Numbers): Command ID(Byte) → Data1(Byte) → Data2(Byte) → Data3(Byte) → Data4(Byte)

## Command Reception Acknowledge:

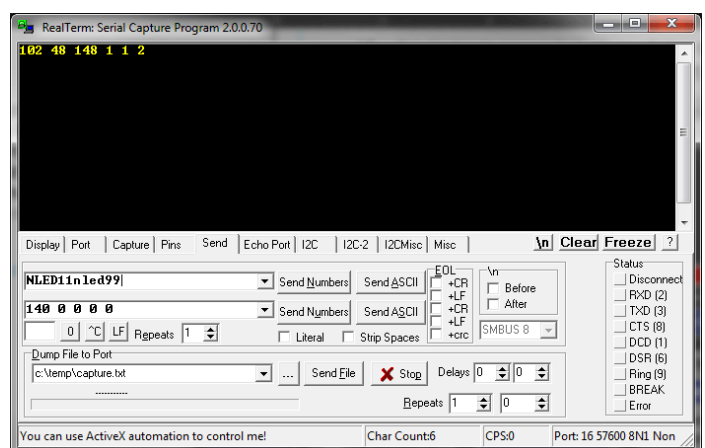
After the device receives the command it will respond

As ASCII characters “cmd” then as numbers: 0 -> Issued Command ID Number

If the issued command was an invalid ID number the device will respond as ASCII “ERR”

Data packets are acknowledged with “zACK” as a string

Sample code with command examples can be found at: <https://github.com/NLED/aurora-interface-example>



## Optional Manual Serial Terminal Command Issuing:

1. Open your serial terminal software of choice.
2. Open the device's serial port(or FTDI or similar serial port), USB devices accept any baud rate. Serial devices require the correct baud rate to be selected. Other settings are usually default, 8-N-1
3. Prepare the command request string. Waiting for the device responses is not necessary. Sending “NLED11nled99” as ASCII characters is acceptable to request a command.
4. The device will respond as ASCII “a9” & “f0”, the device is now waiting for the command bytes.
5. Send as Numbers the: Command ID followed by 4 data bytes. See later pages for command ID and data values.

## List of Available Commands:

Some command names have been altered since the previous command manual was published.

Command Name:	Command ID:
Data Dump	3
Get Device Connection	4
Set User ID Number	5
Set Pixel Packet Clone Value	10
Set Device Output Intensity	15
Set Bulk Live Control - Hold Outputs	60
Set Outputs Blank	61
Set Single Channel Live Control Value	62
Set Single Channel Live Control Release	63
Release All Live Control Channels	64
Set Dot Correction Upload	65
Get Channel Values	69
Set Speed To Value	70
Speed Decrease by 1 or Value	71
Speed Increase by 1 or Value	72
Toggle Pause Device	75
Toggle Device On/Off	76
Set Color Reorder	81
Set Frame Stepping	82
Set Fade Control Value	85
Select Sequence by ID Number	90
Sequence Down	91
Sequence Up	92
Set Hardware Preview Sequence Upload	99
Set Full Sequence and Index Upload	100
Enable Serial Pass Through	110 - NOT DOCUMENTED
Set Device Configurations	101
Set Gamma Correction Upload	102
Get Device Configurations	120
Reset to Default Configurations Options	121
Set Device Bootloader Mode	140
External Device Bootloader Mode	141
Sync Pulse Reception/Slave	200

**The following pages describe each command, the commands are in order of their ID#**

Contact [Support@NLEDshop.com](mailto:Support@NLEDshop.com) for assistance.

**NOTE: JAVASCRIPT API IS ALMOST READY. DETAILS ARE INCOMPLETE**

## **Command Name: Data Dump**

**Command Type: NONE**

**Description:** Not for use by user. Do not issue.

**Usage:** Not Defined

**Command ID:** 3

**Data Byte 1:** Not Defined

**Data Byte 2:** Not Defined

**Data Byte 3:** Not Defined

**Data Byte 4:** Not Defined

**Javascript API:**

**Command:**

**Arguments:**

**Returns:**

## **Command Name: Get Device Connection**

**Command Type: GET**

**Description:**

Controller responds with its hardware/firmware details

**Usage:**

Responds with, HardwareID → Hardware Version → Firmware Version → Firmware Revision ->  
Bootloader HardwareID -> Bootloader Version -> User ID Number

**Command ID:** 4

**Data Byte 1:** Not Defined

**Data Byte 2:** Not Defined

**Data Byte 3:** Not Defined

**Data Byte 4:** Not Defined

**Javascript API:**

**Command:** auroraCMD.requestDeviceConnect()

**Arguments:**

**Returns:** 7 bytes as described.

## **Command Name: Set User ID Number**

### **Command Type: SET**

#### **Description:**

Set the device's User ID number to a specified value. The User ID number can be used to address and interface with specific devices, regardless of the serial or COM port ID number assigned by a computer's operating system.

#### **Usage:**

Send the command with the specified 8-bit value. Once the device receives the command, it will save the ID number to memory. This User ID Number can be polled by sending command *Get Device Connection*(Command ID #4).

If multiple serial devices are connected to a single PC, they can all be polled and specific devices can be accessed. Sample code is available for addressing up to dozens of NLED serial devices to a single computer and software application.

#### **Command ID: 5**

**Data Byte 1:** User ID value. 0 - 255

**Data Byte 2:** Not Defined

**Data Byte 3:** Not Defined

**Data Byte 4:** Not Defined

#### **Javascript API:**

**Command:** auroraCMD.setUserIDNumber(userid)

**Arguments:** userid: is the number to set.

**Returns:**

## **Command Name: Set Pixel Packet Clone Value**

### **Command Type: SET**

#### **Description:**

For pixel controllers only, allows the packet clone feature to be started by commands. Does not save to memory.

#### **Usage:**

Only enabled on pixel controllers, sets packet clone value, could be overwritten by sequences changes, including linked sequence changes. Does not save.

#### **Command ID: 10**

**Data Byte 1:** Packet Clone Value(1-255), 0 = off

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** auroraCMD.setPixelPacketClone(value)

**Arguments:** value: a number 0 to 255, only lower values will work.

**Returns:**

## **Command Name: Set Device Output Intensity**

### **Command Type: SET**

#### **Description:**

Indicates to the controller to apply intensity control to all outputs, with the option to use the intensity temporarily or to save it, once saved it will be applied to all compatible modes, including stand-alone.

#### **Usage:**

Data1 indicates the intensity to apply to all the outputs. Not available on all devices.

#### **Command ID: 15**

**Data Byte 1:** 0 - 255 (equates to 0% - 100%)

**Data Byte 2:** 0 = temporary, 1 = Save Value

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** auroraCMD.setIntensity(value)

**Arguments:** value: the intensity value expressed as 0 to 255(0% to 100%)

**Returns:**

## **Command Name: Set Bulk Live Control - Hold Outputs**

### **Command Type: SET**

#### **Description:**

Starts direct live control of the outputs using bulk method. Once the command is initiated it the outputs will hold their values until updated with a full packet of bytes equal to Channel Amount

#### **Usage:**

Once Initiated, send Channel Amount of bytes. Latches sent data values to outputs once Channel Amount has been received.

#### **Command ID: 60**

**Data Byte 1:** 1 for on, 0 for off

**Data Byte 2:** 1 for 16-bit Mode, 0 or any for 8-bit mode

**Data Byte 3:** Channel Amount MSB

**Data Byte 4:** Channel Amount LSB

#### **Javascript API:**

**Command:** auroraCMD.setLiveControlMode(enable, data size, channels); after setting up Live Mode, use auroraCMD.setLiveControlPacket(buf) to send the data.

#### **Arguments:**

- Arg1: true or false.
- Arg2: 8 or 16.
- Arg3: channel size

#### **Returns:**

## **Command Name: Set Outputs Blank**

### **Command Type: SET**

#### **Description:**

Blanks all the outputs(set to 0%), with option to enter into one of two modes that will continue to hold the blank state until further commands or user interaction.

#### **Usage:**

Enter device into Bulk Live Control mode, then issue this command will turn all the outputs off/blank/0%.

While stand-alone color sequence is running, issue this command to blank/turn off all outputs to await further commands.

#### **Command ID: 61**

**Data Byte 1:** 1 for Bulk Live Control, 0 for Stall/Pause

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** auroraCMD.setOutputsBlank(mode)

**Arguments:** mode: 0 or 1. See Data Byte 1

#### **Returns:**

## **Command Name: Set Single Channel Live Control Value**

### **Command Type: SET**

#### **Description:**

Sets a single channel to live control mode and specifies the output value. Can be used concurrently with stand-alone color sequences. The value is a 8-bit number for most devices, or 16-bit number for 16-bit capable devices. See device datasheet.

#### **Usage:**

Sets a single output channel to the specified value, puts it into per channel live control mode if not already. Send the command to a channel to set the value, a single channel equates to a single color, such as red, not the the entire pixel/color channel. To set a RGB pixel to a specific color while stand-alone is running this command must be ran for R-G-B individually. Passed channel number is base 1, that means red is on 1, green on 2, blue on 3.

#### **Command ID: 62**

**Data Byte 1:** ChannelMSB

**Data Byte 2:** ChannelLSB

**Data Byte 3:** ValueMSB

**Data Byte 4:** ValueLSB

#### **Javascript API:**

**Command:** auroraCMD.setSingleLiveControl(chan,val)

#### **Arguments:**

- chan: channel number to set to per channel live control. Base 1
- Val: value to set the channel to. 8bit: 0 – 255, 16bit: 0 - 65535

#### **Returns:**

## **Command Name: Set Single Channel Live Control Release**

### **Command Type: SET**

#### **Description:**

Releases a single channel from live control mode back to default/stand-alone usage. Can be used concurrently with stand-alone color sequences.

#### **Usage:**

Releases a single output channel from per channel live control. Channel number is base 1.

#### **Command ID: 63**

**Data Byte 1:** ChannelMSB

**Data Byte 2:** ChannelLSB

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** auroraCMD.setSingleLiveControlRelease(chan)

**Arguments:** chan: channel number to release from per channel live control. Will return to normal function such as stand-alone or data reception. Certain stand-alone sequence modes will not continue to work normally after releasing a channel, restart the color sequence if required. Base 1.

#### **Returns:**

## **Command Name: Release All Live Control Channels**

### **Command Type: SET**

#### **Description:**

Releases all output channels from live control mode. Returning them to default usage.

#### **Usage:**

Releases all channels from per sequence live control. And returns them to normal usage, see 'Set Single Channel Live Control Release' for additional details.

**Command ID:** 64

**Data Byte 1:** 0

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** auroraCMD.setSingleLiveControlReleaseAll()

**Arguments:**

**Returns:**

## **Command Name: Set Dot Correction Upload**

### **Command Type: SET**

#### **Description:**

For Compatible Controllers, see hardware info for details

#### **Usage:**

Send bytes equal to channel amount, values 0 – 63.

**Command ID:** 65

**Data Byte 1:** 0

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0

**Javascript API:** None – will not be implemented

**Command:**

**Arguments:**

**Returns:**



## Command Name: Get Channel Values

### Command Type: GET

#### Description:

Returns the output channel value(s), either a range of channels or a single channel value. The response is limited in the number of bytes it can return, if more are requested junk data may also be returned. Returns 8-bit or 16-bit MSB first numbers. 8-bit numbers use one channel, 16-bit values use two channels.

#### Usage:

Will report the specified number of channel values from starting from ChannelNumber Start.

**Note:** Broken in version 2 firmware. Fixed in later revisions.

**Command ID:** 69

**Data Byte 1:** ChannelNumber Start MSB

**Data Byte 2:** ChannelNumber Start LSB

**Data Byte 3:** Number of Channels MSB, in bytes. 16-bit values require two channels each

**Data Byte 4:** Number of Channels LSB, in bytes. 16-bit values require two channels each

#### Javascript API:

**Command:** auroraCMD.requestChannelValue(start,end,callback)

#### Arguments:

- start: starting channel number(base 0)
- num: number of channels to return, up to 24(MSB not currently enabled)
- callback: function to be ran once the data is ready

**Returns:** When the requested data is retrieved the callback argument function is ran.

## Command Name: Set Speed To Value

### Command Type: SET

#### Description:

Alters the Speed for current stand-alone sequence Sent as Unsigned Integer. Each tick is equivalent to 1mS. Actual speed may vary slightly between devices and modes.

#### Usage:

Send an unsigned integer(0 to 65,535) to set the Speed for stand-alone sequences. Does not save, a sequence change or power cycle will revert the change.

**Command ID:** 70

**Data Byte 1:** Speed MSB

**Data Byte 2:** Speed LSB

**Data Byte 3:** 0

**Data Byte 4:** 0

#### Javascript API:

**Command:** auroraCMD.setSpeed(val)

**Arguments:** val: 0 - 65535

**Returns:**

## **Command Name: Speed Decrease by 1 or Value**

### **Command Type: SET**

#### **Description:**

Decrease the Speed by a single value, or up to 255. Affects current stand-alone sequence, does not save.

#### **Usage:**

If Data1 is 0, it will subtract 1 from the speed, if its 1-255 it will subtract that value

#### **Command ID: 71**

**Data Byte 1:** 0 = Subtract 1, Or Subtract 1-255

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** auroraCMD.setSpeedDecrease()

**Arguments:**

**Returns:**

## **Command Name: Speed Increase by 1 or Value**

### **Command Type: SET**

#### **Description:**

Increase the Speed by a single value, or up to 255. Affects current stand-alone sequence, does not save.

#### **Usage:**

If Data1 is 0, it will add 1 to the speed, if its 1-255 it will add that value.

#### **Command ID: 72**

**Data Byte 1:** 0 = Add 1, Or Add 1-255

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** auroraCMD.setSpeedIncrease()

**Arguments:**

**Returns:**

## **Command Name: Toggle Pause Device**

### **Command Type: SET**

#### **Description:**

Toggles device pausing of the current stand-alone sequence, with optional fade down to 0%

#### **Usage:**

Each command issued toggles Pause state with options.

Note: Setting the Pause with Fade down will not continue the sequence correctly after un-pausing.

#### **Command ID: 75**

**Data Byte 1:** 0 to toggle, 1 to pause, 2 to play

**Data Byte 2:** 1 = Fade Down all outputs to 0%, 0 = None Fade Down

**Data Byte 3:** Fade Down Speed MSB - Lower is faster

**Data Byte 4:** Fade Down Speed LSB - Lower is faster

#### **Javascript API:**

**Command:** auroraCMD.setPlayPause()

**Arguments:**

**Returns:**

## **Command Name: Toggle Output On/Off**

### **Command Type: SET**

#### **Description:**

Toggles the outputs on and off.

#### **Usage:**

Send Data Byte 1 as '0' to enable the outputs for normal usage, restores the previous color sequence when issued. Send Data Byte 1 as '1' to disable the outputs and clear them(turn the outputs off. Send Data Byte 1 as any other number to toggle enable/disable outputs.

#### **Command ID: 76**

**Data Byte 1:** 0 = Toggle Enable/Disable, 1 = Enable Outputs, 2 = Disable Outputs

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** auroraCMD.setPlayPause(callback)

**Arguments:** callback: function ran after the command is acknowledged by device.

**Returns:**

## Command Name: Set Color Reorder

### Command Type: SET

#### Description:

Swaps the colors based on 6 methods(including unaltered) See documentation for details. Data 2 should be 0 to run the function, or to reset Color Swap send anything but 0 in Data 2. Not available on all controllers, pixel controllers support this command.

#### Usage:

Send 0 to increment Color Swap Mode, or send 1 – 6 to set the mode, See image below for details.

**Command ID:** 81

**Data Byte 1:** 0 - 6, see image for ID number

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0








#### Javascript API:

**Command:** `auroraCMD.setDeviceColorOrder(val)`

**Arguments:** val: 0 through 6. See Data Byte 1

**Returns:**

#### NLED Aurora Control v.2a Color Swap Methods: By ID#

0: RGB	
1: BRG	
2: GBR	
3: RBG	
4: BGR	
5: GRB	
6: GRBW	

[www.NLEDshop.com/nledaurora](http://www.NLEDshop.com/nledaurora)

## **Command Name: Set Frame Stepping**

### **Command Type: SET**

#### **Description:**

Allows commands to step through a sequence's frames one by one, either forwards or backwards each time the command is issued. When first issues the device will Pause on the current frame. The displayed frame does not change until the sequence is changed or another command is received.

#### **Usage:**

Run CMD: 90 to load a sequence by ID number, or otherwise select a sequence. 'Instant', 'POV', or 'File' sequence modes are compatible. Then issue this command to step the output colors frame by frame.

#### **Command ID: 82**

**Data Byte 1:** 1 for Forward, 0 for Backward

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** auroraCMD.setStepForward(callback), auroraCMD.setStepPrevious(callback)

**Arguments:** callback: function ran after the command is acknowledged by device.

**Returns:**

## **Command Name: Set Fade Control Value**

### **Command Type: SET**

#### **Description:**

Stalls a Fade sequence and waits for command to allow it to cycle a the color values x amount of times. Variable should be a unsigned integer. Sending 200 will let each channel cycle normally 200 times. It takes 255 x (Amount Of Frames) for a full rotation of the setting's colors. No Linked sequences are supported.

#### **Usage:**

First run CMD: 90, to load the a sequence by ID number or otherwise select a sequence. 'Fade', 'Gradient', sequence modes are compatible. Then Pause the sequence using command CMD: 75. Then issue this command when required to fade the sequence forwards a defined amount of steps. After the device cycles the specified steps it will pause and hold the current output colors.

**Command ID:** 85

**Data Byte 1:** Steps Amount MSB

**Data Byte 2:** Steps Amount LSB

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** `auroraCMD.setControlFade(val, callback)`

##### **Arguments:**

- `val`: number of steps to fade/cycle.
- `Callback`: function ran after the command is acknowledged by device.

##### **Returns:**

## **Command Name: Select Sequence by ID Number**

### **Command Type: SET**

#### **Description:**

Selects a stand-alone Sequence by ID number. The ID number is the the sequence's position in the Index.

#### **Usage:**

Loads a sequence by ID indexed ID number, with mode. Start playing or start paused. Data2 with a value of 1 will start the Idle Sequence, otherwise send 0.

**Command ID:** 90

**Data Byte 1:** Sequence Number(0-255, base zero, means #0 is the first sequence)

**Data Byte 2:** 0 = Ignore, 1 = Idle Sequence

**Data Byte 3:** 1 for Pause, 0 for play

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** auroraCMD.setSequenceByID(id), auroraCMD.setIdleSequence()

**Arguments:** id: sequence ID number, which is the same as the index slot number.

**Returns:**

## **Command Name: Sequence Down**

### **Command Type: SET**

#### **Description:**

Decrements Sequence number, as if the button was pressed. The device will then start playing the previous sequence in the index, wrapping to the beginning if required.

#### **Usage:**

Send command, instantly changes sequence, saves to memory.

**Command ID:** 91

**Data Byte 1:** 0

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** auroraCMD.setSequencePrevious(callback)

**Arguments:** callback: function ran after the command is acknowledged by device.

**Returns:**

## **Command Name: Sequence Up**

### **Command Type: SET**

#### **Description:**

Increments Sequence number, as if the button was pressed. The device will then start playing the next sequence in the index.

#### **Usage:**

Send command, instantly changes sequence, saves to memory.

**Command ID:** 92

**Data Byte 1:** 0

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** auroraCMD.setSequenceNext(callback)

**Arguments:** callback: function ran after the command is acknowledged by device.

**Returns:**

## **Command Name: Set Hardware Preview Sequence Upload**

### **Command Type: SET**

#### **Description:**

Uploads a single sequence to the controller for Hardware Previewing.

#### **Usage:**

Not for user, details proprietary.

**Command ID:** 99

**Data Byte 1:** 0

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** PROPRIETARY

**Arguments:**

**Returns:**



## **Command Name: Set Full Sequence and Index Upload**

### **Command Type: SET**

#### **Description:**

Uploads the index and all indexed color sequences and their values.

#### **Usage:**

Not for user, details proprietary.

**Command ID:** 100

**Data Byte 1:** 0

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** PROPRIETARY

**Arguments:**

**Returns:**

## **Command Name: Set Device Configurations**

### **Command Type: SET**

#### **Description:**

Uploads the user selected configuration settings from NLED Aurora Control or other software. These configuration settings are specific to the device.

#### **Usage:**

Not for user, details proprietary.

**Command ID:** 101

**Data Byte 1:** 0

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** auroraCMD.requestConfigUpload(buf)

**Arguments:** buf: is the configuration data to be uploaded as an array

**Returns:**

## **Command Name: Set Gamma Correction Upload**

### **Command Type: SET**

#### **Description:**

Uploads the optional Gamma Correction table values. Not available on all devices.

#### **Usage:**

Depends on controller, contact for assistance. More details released in the future.

**Command ID:** 102

**Data Byte 1:** 0

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** auroraCMD.requestGammaUpload(amount,packet)

##### **Arguments:**

- amount: expected number of packets(not bytes/values)
- packet: the gamma table data as an array

**Returns:**

## **Command Name: Get Device Configurations**

### **Command Type: GET**

#### **Description:**

Gets the controller's current configuration settings. Values the device sends back are specific to the device. More info available in the future.

#### **Usage:**

Responds with the devices configuration flags and variables.

**Command ID:** 120

**Data Byte 1:** 0

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** auroraCMD.requestConfigDownload(callback)

##### **Arguments:**

- callback: function is ran when data from device is ready. Data is passed as an array.

**Returns:** Multiple bytes based on device profile

## **Command Name: Reset to Default Configurations Settings**

### **Command Type: SET**

#### **Description:**

Resets configuration bits and most configuration bytes to default or 0. This will remove any configuration settings the user has applied to the device.

#### **Usage:**

Instantly updates the device's configurations flags and bytes to default as specified by the device's firmware.

**Command ID:** 121

**Data Byte 1:** 0

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** auroraCMD.setDeviceConfigDefault()

**Arguments:**

**Returns:**

## **Command Name: Set Device Bootloader Mode**

### **Command Type: SET**

#### **Description:**

Enters device into bootloader mode, will need to close serial/USB port connection and open the separate bootloader software. Note: After bootloading some devices may re-enter bootloader mode when reset, if your device keeps entering bootloader mode after bootloading new firmware, power cycle the device.

#### **Usage:**

Immediately after issuing command the controller will enter bootloader mode. USB connection will be dropped or stalled. And a USB HID device should be started.

**Command ID:** 140

**Data Byte 1:** 0

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** auroraCMD.setBootloaderMode()

**Arguments:**

**Returns:**

## **Command Name: External Device Bootloader Mode**

### **Command Type: SET**

#### **Description:**

For compatible devices with onboard modules or secondary microcontrollers. This command enters the device into transparent bridge serial mode which allows other devices to be bootloaded through the USB connection. Used for ESP WiFi modules, more function to be added in the future.

#### **Usage:**

After issuing the command the Aurora device will enter a bridge mode. Disconnect from the Aurora USB port and open the third-party bootloading software for the external device. Such as NodeMCU flasher. Set it up and begin the flashing process. Once complete unplug the controller's USB port, and

**Command ID:** 141

**Data Byte 1:** 0

**Data Byte 2:** 0

**Data Byte 3:** 0

**Data Byte 4:** 0

#### **Javascript API:**

**Command:** `auroraCMD.setExternalBootloaderMode()`

**Arguments:**

**Returns:**

## **Command Name: Sync Pulse Reception/Slave**

### **Command Type: SET**

#### **Description:**

Used for syncing multiple controllers together. One controller is the master that can be connected to multiple slaves. Slave devices receives the sync command will immediately set the color sequence to the received ID# and sets the frame number. The actual color data is not sent from the master to the slave, all the connected controllers must have the color sequences in memory and indexed the same. They don't need to be the same color sequences, they could be different color sequences, or could be a part of a larger amount of pixels. Such as synchronizing two of the same LED screens that are too large for a single controller. Or dividing a large screen into smaller sections and controlling each section with a single controller.

#### **Usage:**

Set one controller as a Sync Master, create, index and load the color sequences into memory. Set one or more controllers as a Sync Slave, create, index, and load the color sequences into memory. Connect the controller's together as described in their datasheets.

Only works with color sequence modes: POV, File

#### **Command ID: 200**

**Data Byte 1:** Sequence ID Number

**Data Byte 2:** 0 - No Usage

**Data Byte 3:** Frame Number MSB

**Data Byte 4:** Frame Number LSB

#### **Javascript API:**

**Command:** auroraCMD.setPulseReception(seqid, frame)

##### **Arguments:**

- seqid: sequence ID number which is the index slot number. 0 – 255
- fame: frame number to latch. 0 – 65535. If it is over the sequence's max frames, it defaults to frame 0.

##### **Returns:**