Starr Labs Clipper
Clip-Based Live Interactive Performance controller
USER Manual R1.1 12/27/12
User Manual R0.2, Beta release 12/7/2012.
(Firmware Revision 3.0 12/1/2012)

The Clipper fingerboard is a programmable music control surface with multicolor LED visual feedback in each key. The key-action is a standard Ztar fully polyphonic, touch-sensitive fingerboard. The Clipper's embedded software is designed to act as a remote Ableton Live * clip control matrix, as well as offering a variety of dynamic lighting effects. In addition, the Clipper integrates the standard Starr Labs MIDI guitar functionality with many MIDI control setup options. Using our standardized LED controller MIDI command set you can create many types of interactive applications such as games and teaching software.

This document primarily describes the detailed User setups to the Clipper for use with Ableton Live. These setups are pre-loaded in your instrument, and this is your reference to the menu screens pertinent to the interactive lighting features.

- "Ableton Live" property of Ableton Inc.

Contents:

Overview 1
Ableton Live to Clipper: Detailed Interface Specification 2
MIDI message Format 6
Setting up Ableton Live to work with the Clipper 7
Setting up the Clipper to work with Ableton Live 7
Zone Setup 8
Keymap setup 9
MIDI Merge settings 10
Color settings 11
Images 12
Tuning Maps 12
LED Controller Command Specification 14
Appendices and Charts 21
QUICK Ableton Mac/PC setup 23
Index 24

Hardware setup

- The Clipper can be connected to your computer with a MIDI cable to the Clipper's MIDI Out Port, or a standard USB A/B cable.
- At full brightness the LEDs require more power than a standard USB2.0 port can provide. The Clipper must be powered by "Phantom power" through the MIDI cable in order to run the lighting. Power is supplied to MIDI cable by our PB-1 interface adapter and the AC/9VDC wall-adapter. You can still connect to your computer via the USB cable which will carry only data. USB power will only support the MIDI control functions of the instrument.
- A 3-position power switch is provided: *USB<OFF>MIDI*

Overview of the Ableton Live interface

When configured as an Ableton Live controller the Clipper functions much like other commercially available Clip grid controllers, but with added functionality. Other grid controllers have a limited amount of visual feedback, usually representing only the state of a clip with LED blinking/brightness and three colors. For instance Yellow for standby, blinking Yellow for cued, Green for playing, Red for recording.

However, on the Clipper the color of each LED represents the color of each respective clip in Ableton's Session View GUI with brightness and blinking representing the clip states. When a clip is loaded it is at half brightness. When a clip is cued to play it blinks between half brightness and off. When the clip launches it lights up at full brightness. This allows the user to color code a Live set and not have to look at the computer screen throughout a performance. Thus the Clipper liberates the computer musician from the laptop stand, allowing for greatly improved virtuosity. ---Imagine how much cooler you will be when you aren't hidden behind a computer screen twisting knobs and pushing buttons on a tabletop control surface… SO COOL!

If you're interested to try programming the Clipper for your own lighting effects it's easy with our dedicated MIDI control command-set. Using standard MIDI Note and Continuous Control messages you can light the LEDs to any color, program and stream text and images, scroll the display, blink individual LEDs and other functions. Using the setups in the Ztar OS you can create Zones of color, Tuning maps for different guitar tunings, load and select images, and create animations.

Now that you're excited about your new Clipper fingerboard, let's get into the specifics.

Ableton Live to Clipper: Detailed Interface Specification

Ableton Live (currently at Rev.8.3) is a popular music software application that allows a user to record and playback clips of music in a wide variety of ways, all synced to a master beat. It is also popular to use a Remote Control surface, a small console with buttons and knobs, and often LEDs for stratus feedback, using semi-proprietary command sets developed specifically for each peripheral.
The Ztar is a MIDI guitar controller with a keyboard-like fingerboard having a touch-sensitive key for each note position. The keys can easily be mapped to control a Live 'Session', or instance of Ableton Live running in a host PC or Mac. With some modifications the Ztar can be made to reflect the status of the session's display screen as well. The Ztar or any MIDI controller directs Ableton Live by sending MIDI commands which are interpreted and re-mapped by Live to perform various internal functions of the software.

These MDI commands are standard and need no special coding from the device side other than a simple user-setup to make the specific assignments. When there are a lot of controls to be mapped, it is more convenient to create a profile, or Control Surface script using Python that is loaded into the program when the device is plugged in. Also, when it becomes necessary for the remote device to show the Host's programs status, host-to-device output software must be created, also using Python scripts which are loaded into the same master profile for that device.

Requirement for Host-to-Device MIDI communication:

An Ableton Live "Session" displays on the computer screen an array of colored boxes that form the Clip Grid and change dynamically as the music performance proceeds. In order to have a remote control surface display this same dynamic status information from the current Session in the host computer, Ableton communicates the various color changes to the remote peripheral control surface via a USB/MIDI connection. As the activity of the Live session changes, with different audio and MIDI clips turning on and off, playing, recording, and cueing for the next operation, the computer display is continually changing to reflect the current status, and the remote surface will update to correspond with the computer screen including color-matching the status LEDs for the Live clip performance matrix, or Clip Grid".

Ableton Live provides an API for creating modifications to the operation of Live. When a new hardware Control Surface is defined for Live, a set of assignments maps the MIDI messages from the Control Surface to the internal functions of the Live software. This mapping is created by use of a Python script that loads manually or when Live recognizes the connected (enumerated) USB- MIDI device. MIDI Notes and MIDI CC messages are commonly used to control the Live software.

In addition to mapping the MIDI-notes input to control Live, the Python script can extract the dynamic status of the individual virtual “buttons” and “pot/encoders” on the Live GUI, including color information for the buttons. When the current color-state of a button is determined, the corresponding location on the remote Control Surface can be made to light up to match the appearance of the software.

That said, we can most easily define the necessary control stream to the LEDs in the StarrLabs peripherals by using standard MIDI messages from the host computer. The format is simple, and it carries enough data payload in a simple message to light any arbitrary LED. Also, it affords some unique opportunities for direct musical control of the LEDs, aside from their use in the Ableton Live context.

The **Clipper** fingerboard is a guitar-format keyboard with 6 “Strings” (columns) x 24 “frets” (rows) of touch-sensitive keys for a total of 144 keys. We may add more Strings to cover expanded arrays. Each of the 144 keys is fitted with a 3-color LED for a total of 432 LEDs. This array of LEDs may be controlled in full color by sending an encoded color-value to each pixel, or triad, located under each key.

The colors themselves are encoded in 15 bits allowing a total of over 32000 colors. These color values are arbitrary and when we’re connected to Live we use Live’s color table so that you can match the LEDs to the clip-colors in your session. The colors are then re-mapped inside the Clipper so they can be selected by a 7-bit MIDI note-velocity value. From the point of view of the host computer, the array of LEDs is a list containing a 7-bit color value for each of 144 locations.

Ableton Live Clip-array:

This is a Ztar zone of arbitrary size that reflects the RedBox within the Clip-grid on the Ableton Live screen. In performance mode, clips are loaded, recorded or launched.

When we set up the Ableton "Live Zone" on the controller we will define X-Y boundaries on the fingerboard. That becomes a static box for your performance and also defines the boundaries for the RedBox. Thus the RedBox size is arbitrary. Maximum size is 5x23 for Clips with a sixth row for StopClips, and a twenty-fourth column for Scene Launch. Live only controls the “Live” Zone on the Clipper. All other Zones are unaffected.

Automatically the Stop Clips Row is lit at the Bottom in Red and the Scene Launch Column is lit in Green.

Launch Scene Keys:
This is the column of Green-lit keys placed to the right of the clip keys array for the LiveZone on the Ztar for a Zone of any size. This is also called the master channel. The function of each of these keys is to trigger all keys in the corresponding row in the clip key array.
The bottom key in the far right column is Stop All Clips.

Stop Keys:
This is a row of keys placed below the clip keys and the launch scene keys. The function of each of a Stop key is to stop all clips in the corresponding column in the clip key array. The rightmost key stops all clips in all columns.

Blinking stop clip button:
The stop clip buttons should blink during pre-roll to indicate that a button push on that stop clip has been initiated. The stop clip button should blink until the downbeat where it stops the playing clip (bringing it to half brightness) and returns the stop clip button to full brightness (non-blinking).

Visual Feedback

The lighting for the keys enclosed by the RedBox is controlled by the Ableton script according to the LiveZone setup in the controller. Ableton only controls what's in the Live Zone. The colors of the various Clips will scroll through the LiveZone as the RedBox scrolls on the Ableton screen. Likewise, visual feedback from scrolling the RedBox from the controller is shown on the computer screen. Other Zones on the fingerboard that are not controlled by Ableton can have their color data controlled separately either from the Ztar or an external computer.

To be clear, where the boundaries of the RedBox on the fingerboard are fixed but the LiveZone is scrolled, the lighting values of the clips outside of the Live Zone do **not** show on the fingerboard. The Non-LiveZone areas are for other performance purposes to be colored separately. The Ztar OS has utilities for creating zones with a base color.

1. When a Clip location is empty, the color is a low-intensity White to highlight the LiveZone.
2. When a Clip is loaded in Ableton the fingerboard key-color assumes the clip's color.
3. When the Clip is idle the color is half-intensity.
4. When the Clip is playing the color is full intensity.

Clip Launch modes:
(These are ordinarily set in the Ableton Clip Launch Mode setup dialog)

Trigger - once the button is pressed it latches and plays the clip until it reaches it's end or a **Stop**

button is pressed. This will usually be the stop clip button at the bottom of the track, but can be any blank clip with an active stop button in that track. *Note-offs are ignored.*

Gate - F*ull intensity when active.* Momentary operation. You must hold down the button to play the clip. This is handled by Note-On on button-press and Note-Off issued button-release. Normal keyboard mode.

Toggle - *Full intensity when active.* Press once for the clip to play and repeat, press again for it to stop. Ignore Note-Off. Successive Note-On's toggle the function.

Repeat - mouse down/note on repeatedly plays the clip, retriggering at the time interval of the clip's launch quantization. when released the clip plays out like Trigger mode

Button Flashing: Ableton buttons flash when they are Armed, that is, the clip is ready to Play or Record but is waiting for the actual Start-time, to come around before firing. The start time is usually the "downbeat" but other Launch Quantizations are available.

Recording Mode:

When recording to a Clip, color is Red. We aim to represent the status of the little Clip arrow at the left of the Scene (row), which is sometimes solid and sometimes blinking.
The fingerboard Clip button becomes Red and is steady-state or flashing depending on the Recording-status.

1. Recording may be started by pressing a Record button in which case the recording starts immediately and the arrow glows solid Red.
2. Recording may be "Armed" with a 2-bar count-in where the LED flashes and when the recording actually begins the LED glows solid Red.

Playback Mode:

The fingerboard Clip button is steady-state or flashing depending on the Playback-status. The Ableton Record/Playback Arrow shows the status.

1. Playback may be started by pressing the play button in which case the playback starts immediately and the button glows at full intensity with the assigned color.
2. Playback may be "Armed" with a 2-bar count-in where the LED flashes and when the recording actually begins the LED glows its solid color.

MIDI message format:

Each MIDI channel can address 128 note-numbers.

A full MIDI Note-On message is composed of (3) 7-bit bytes.

A full frame of color data assigns a 7-bit value to each fingerboard location. (144) 7-bit bytes.

Byte #:1 Status byte (in binary) 0101nnnn where 09h = MIDI Note-On, and nnnn is a 4-bit channel number.

Byte #2: MIDI Note-number, 0nnnnnnn, where nnnnnnn is a 7-bit note-number. This is used to address each LED location. Each key on the fingerboard has a unique channel/note address.

Send MIDI Note-Off to turn off an individual LED. Both a literal Note-Off and a Note-On velocity=0.

Byte #3: Velocity value, 0nnnnnnn, where nnnnnnn is a 7-bit value. This normally encodes the force of a key-strike. In our case this is used to encode the color-value for each LED pixel.

These color values are index-numbers into an array on the device that holds a list of values to create 128 unique colors (including OFF = 0 = no light)*.

**Note there are two ways to turn OFF an LED.*

Additionally, once a MIDI status byte has been issued from a MIDI controller, the channel number is then established, and most controllers will enter what is called **"Running Status"** which dispenses with the status byte and sends only 2-byte messages containing just the note-number and velocity information. When the channel-number must be changed or a different type of message must be sent, then the controller drops out of Running Status, sends a new Status byte as necessary, and re-enters Running Status. In the case where there are more than 127 LEDs in the array, a channel switch will be necessary to address the full array.

We can use, arbitrarily, MIDI channels 15 and 16 which are rarely used in most MIDI setups. Two channels yield 256 programmable locations which will easily cover our requirements.

The array is ordered starting with "String#1, Fret#1" which is the lowest fretted note on a guitar for a right-handed player. The highest note-location on String#1 is LED-location #24.

Setting up Ableton Live to work with the Clipper

The Clipper communicates with Ableton's Live API using MIDI Remote Scripts. These Python scripts are available on our website, and interpret incoming standard MIDI messages and translate them to API calls within Live. You can edit these if you want, but if you just want to get things going right at the start we recommend that you simply copy the folder containing the remote scripts into the following operating system specific locations.

Windows: /Program Files/Ableton /Live 8.x.x/Resources/MIDI Remote Scripts

Mac OSX: /Applications/Live 8.x.x/Live.app/Contents/App-Resources/MIDI Remote Scripts

The name of this folder will be the name of the control surface in Ableton Live. Currently the standard name is “Ztar” but feel free to rename it something you like and will remember for the next step.

Once this folder has been copied, (re)start Ableton Live, open up the Preferences menu and navigate to the MIDI tab. There you will be presented with a grid of dropdown menus. In the control column open one of the dropdown menus and scroll down to Ztar (or whatever you renamed it). Then in the next two columns to the right, select your input and ouput ports for the instrument.

In the lower half of the MIDI preferences window you will see a grid with options for all of the active MIDI ports. Make sure that both “Track” and “Remote” are set to “On” for your previously assigned input and output ports. The “Sync” option is for sending/receiving MIDI clock messages and can be left off for most use cases. We recommend leaving it off to reduce the amount of messages the Ztar has to handle.

Now Ableton is ready to interact with your Clipper enabled Ztar, but is your Ztar ready? We have Ableton ready setups available on our website which can be programmed into your Ztar via System Exclusive MIDI messages (SysEx). You can also set up the necessary fingerboard zones with using the following instructions

Setting up the Ztar/Clipper to work with Ableton Live

- *These setups have already been included in the Ztar-ABL-OS version. The following instructions explain how the setups are created and the way the Ztar interacts with Ableton Live.*
- The Ztar and the Clipper LED Controller communicate with any external MIDI device, including Ableton Live running in a host computer, by means of standard MIDI note and Continuous Control messages, as defined in the Clipper LED interface specification. For this purpose we reserve MIDI channels 15 and 16 for our various MIDI control functions.

  On the output side, the Ztar fingerboard uses a special “keymap” that assigns a unique channel and note-number combination to each note-location on the fingerboard. Keymaps are selected, edited, and stored in the Fretboard/Tunings Menu and where a unique set of eight full maps may be saved for each “Song”, or master preset in the Ztar.

On the input side, the same range of messages on Channels 15 and 16 are used to address the individual LEDs. Also there are a variety of embedded functions in the Clipper that may be accessed by other Note- and CC-messages as listed in the Clipper LED Command sheet.

Some terminology: as regards the "RedBox", within Ableton Live the RedBox is the active clip-zone.

The corresponding area of the fingerboard assigned to the RedBox is called the **"LiveZone".** The LiveZone is fixed once it's set up on the fingerboard. The RedBox will scroll through it.

- *Just to be clear, the **horizontal row is a Scene** and the **Column is a Track**. The analogy for the column is a "Bass-track" or "Piano-track", for example. The Scene is then the whole "Band" or "Composition" comprised of all Tracks.*
- *Only one clip can play per track, or column, at a given time*

Setup the Red Box:

The Ableton Red Box is a scrollable window on the Live screen that highlights the block of currently active clips in the much larger Clip Grid array. The Red Box is scrollable in both X and Y directions.

Zone setup:

Some terminology regarding Zone Boundaries:

The Clipper is modeled after a guitar fingerboard which is described in terms of strings and Frets.

- A String is comprised of the 24 consecutive keys in each of the 6 rows running the full length of the fingerboard.
- A Fret is comprised of the 6 consecutive keys in each of the 24 columns running across the width of the fingerboard.

The interactive Live setup is composed of three Zones on the fingerboard.

1) A zone to receive the lighting information from Ableton.
2) Two zones to send Clip-control commands to Ableton.
3) In the case of a 12-string fingerboard, four zones are needed for independent access to each key as we put both Channels 13 and 14 into use here.

Open the Ztar Zone menu. The "Tuning/Transposition" sub-menu has a "Color-Zone" field that may be set to *"RedBox"* or *"Local", or "Off".* Setting the soft-switch "CZone"to something other than Off simply means that the Zone will now send a ***MIDI sysex*** message that contains the Zone's boundaries and color information whenever the Boundaries or the Color are changed. Also, this Sysex message can be sent when the Ztar "Song" (onboard preset) is selected.

If *"RedBox""* is set, Zone Boundary data are sent via Sysex to MIDI-Out-A which goes both to the MIDI Out jack and also the USB port. Lighting information is returned from the Host computer on MIDI-In-B which is connected internally via MIDI-Out-B to the LED controller in the fingerboard.

If *"Local"* is set, Zone Boundary and Color-number data are sent via Sysex to MIDI-Out-B which is connected internally to the LED controller in the fingerboard.

Use the R/L switch to convert the color zones, or to tell Ableton whether you're playing Right-handed or Left-handed. Find this in the FRETBOARD > TUNE/TRANS > MORE menu.

The ***only*** function of this option is to adjust the R/L flag that's sent out in the sysex message -- it has no effect on fingerboard tuning. You now press the softkey to select the field, and then use the up and down buttons to change from R/L to L/R. Note that the left/right swap in the TUNING menu will ***not*** affect this flag.

Keymap setup:

1) Zone #1, is created with the boundaries you want to represent the RedBox for your Ableton session. This will be used by Ableton to control the lighting on the fingerboard.

The RedBox will have a re-sizeable ClipGrid, a StopClips row at the bottom, and Scene Launch buttons at the right-most column. So, if you want to use the entire 24-fret x 6-string fingerboard you will have a Clip-Grid of 23-frets x 5-strings. The 24th fret will be Scene Launch buttons for the 5 scenes.

If you are using the full fingerboard for Ableton, set:
Low Boundary= String1,Fret1; High Boundary = String6,Fret24.

Set the Zone function to **Keys: Off** so that it does not send any MIDI note messages.
Set Color-Zone=RedBox. The Color-number does not matter.

2) Create a Keymap :

[a Keymap is a map showing the note-assignment to every key on the Ztar fingerboard. When you re-tune a Zone, you are creating a new Keymap. The Map# for each Zone is shown in the Zone>Key/Trans>Tunings menu. Hit the EDIT softkey to edit the selected map by individual keys.]

- In our LED fingerboard instruments, this has been created as a default and you should not need to create this map yourself. It is also available as a Sysex file on our website.

String#1,Fret#1- Fret#24 : LED locations 1-24 ; Channel 15 MIDI Notes 24-47(*)
String#2,Fret#1- Fret#24 : LED locations 25-48 ; Channel 15 MIDI Notes 48-71
String#3,Fret#1- Fret#24 : LED locations 49-72; Channel 15 MIDI Notes 72-95
String#4,Fret#1- Fret#24 : LED locations 73-96; Channel 15 MIDI Notes 96-119
String#5,Fret#1- Fret#24 : LED locations 97-120; Channel 15 MIDI Notes 120-144(**)
String#6,Fret#1- Fret#24 : LED locations 121-144; Channel 16 MIDI Notes 24-47 (*)

*Due to the way Octaves are implemented on the Ztar, in order to enter a MIDI Note#-0 (for any channel), you must set Note-24 as the lowest available note#, then transpose the ZONE down by two octaves, or -24 half-steps to offset the entire keymap to start at MIDI Note-0..

**While ordinarily "127" (decimal) is the highest allowable note# in MIDI, because of the aforementioned reason, the top note on String#5 becomes "144" and the Zone-offset of -24 adjusts the top-note back to a correct value of "120".

3) **Create two Zones that use this special Keymap.** These will be used to send the control commands for each key as it corresponds to a clip in the Ableton session.

Zone2: Low Boundary= String1,Fret1; High Boundary = String5,Fret24.
Set this Zone to MIDI Channel 15.
Zone3: Low Boundary= String6,Fret1; High Boundary = String6,Fret24.Set this Zone to MIDI Channel 16.

If you are not using Ableton, or simply want to play guitar on the Clipper, but have the LEDs light the keys as you play:

Set the Fingerboard to use three Zones -

1) Zone #1 uses a keymap with an assigned "Guitar" or other selected tuning of your choice and directed to MIDI Channel#1 or any channel other than 15 or 16. This will map the notes that you hear.
2) Set two other Zones to use the special Lighting keymap that assigns a unique note-message to each key/LED using channels 15 and 16. This map assigns Notes 1-120 for channel 15 and Notes 1-24 for channel 16.

Moving the RedBox

Up ----- Note 110 Channel 16

Down	-----	Note 109	Channel 16
Right	-----	Note 108	Channel 16
Left	-----	Note 107	Channel 16
Rotate 90°	-----	Note 84	Channel 16

[Note: the RedBox boundaries may also be changed within the Clipper's onboard user-interface by changing the Zone boundaries for the 'RedBox' zone.]

MIDI Merge settings

You can assign any of the following Merge settings to each of the MIDI ports independently.

- The Ztar has (2) full MIDI Ports, Left and Right. (You can think of them as #1 and #2).

  The Left MIDI port Input is our standard MIDI In connector on the Ztar.
  The Left MIDI port output drives our normal MIDI Out connector and the USB Output.

  The Right MIDI port input is attached to the USB input.
  The Right MIDI port output drives the input to the LED controller board.

Go the **UTILITIES/More>MIDI-IN** menu and set the Merge settings. *The RedBox Zone setting should make these settings for you automatically.]*

Merge: ***Off/Notes/Sysex/Both/NotesX,SysexX/BothX***.
Notes: Only merge incoming notes from the selected input port to the same corresponding output port. (In#1-Out #1 or In#2- Out#2.)
Sysex: Only merge incoming Sysex data to the selected output.
Both: Merge both Notes and Sysex data to the selected output.
NotesX: Merge only incoming Notes to the opposite MIDI Port number. (In#1-Out #2 or In#2- Out#1.)
SysexX: Merge only incoming Sysex data to the opposite MIDI Port number.
BothX: Merge both Notes and Sysex data to the opposite output.

Note: **If *<u>any</u>* zone is enabled as a Redbox zone, the MIDI Merge assignments will be set so that both MIDI In and USB In will communicate with the LEDs.**

That automatic Merge setup is as follows: (**MIDI Left Input = crossed, MIDI Right Input = normal**). If all zones are either "Off" or "Local", Merge is disabled.

The automatic setting is only applied when you change the CZone type in the Fretboard > Tune/Trans menu, so once you have set up your zones, you can then go to the MIDI-RX menu and change the merge options manually and independently to the two ports if you wish. But the next time you change a CZone setting it will reset the merge options.

Color Settings

Terminology

Color Map: A defined palette of colors.

Indexed Color: An arbitrary palette of colors whose various colors may be selected by an index number. Color Map Index: A number that selects a given color in a color map. In the case of the Clipper the indexed color map consists of 127 colors and the MIDI velocity is the color-index. The Ableton Live color map is an indexed color map of just 60 colors.

Direct Color: The color hue is directly defined by its component Red, Green, and Blue intensity values. The Clipper uses 15-bit color resolution which may define 32,000 different colors. An arbitrary selection of these colors may be loaded into an indexed color map.

Ableton Live currently uses a color-picker from either the Mac or Windows OS that offers 60 color choices. These 60 values are mapped into our available 7-bit color value and selected from an embedded table in the Clippper. When the Live session starts the Ableton color-map is sent to the Clipper as the colors are used. Other color maps may be loaded into the Clipper as well, and several alternative maps are included.

Ableton Live carries internal variables that store all of its dynamic status including current color-status for each on-screen clip-location among other things. That information is available using calls to the Python-based LiveAPI. Using this information we are able to extract the dynamic clip/color status and output to the Clipper via MIDI data to update its LED array.

Setting the Color Number:

The Color number is used by the ZtarOS and the Clipper software to assign a given color to a zone of keys when playing the Clipper as a musical instrument, other than the Ableton session usage. Simply set the **CZone= Local** and change the color number to view the colors in your selected color map.

- *The Color-number adjustment in this menu is always visible but you cannot adjust it when in the Redbox mode. You need to have Czone:Local or Czone:Off in order to adjust the color value.*

Color-Zone Sysex Message format

When the Zone-boundaries are changed for the RedBox Zone the ZtarOS will send a Sysex to Ableton Live to reset the size of the RedBox to match. The following information is sent to Live:

Transmits the boundaries of zone 'z' (1-32) in the current song, as a Sysex message in the format:

$F0 $00 $01 $42 pp nn nn ll ll ZZ LS LF HS HF RL CC csum $F7

where:

pp = packet type, (7-bit unpacked data) = $40

nn nn = subtype number (14-bit) = $00 $03

ll ll = data length (14-bit) = $07

ZZ = zone number

LS = lo string

LF = lo fret

HS = hi string

HF = hi fret

RL = R/L switch , 0=right, 1=left

CC = color index (7-bit)

IMAGES

The Clipper currently can hold 10 full fingerboard-frame images which may be uploaded using the companion LED control software for the PC. Images are stored as MIDI SysEx files which may be dumped to the Clipper using any standard MIDI SysEx utility. The capacity will be upgraded to 32 images and more in the future.

TUNING MAPS

The Clipper stores (5) Tuning maps which are used to represent “fixed” images that may reside on the fingerboard underneath any dynamic lighting effects that may be active.

1) Fret Marks: shows guitar fret marks at the standard fret locations.
2) Tertiary guitar: 12 colors, 1 per chromatic scale note-name. Guitar tuning.
3) WhiteBlack (2 color): The fingerboard is tuned to standard Guitar tuning and the keys are coded according to the Black and White leys of the piano
4) Tertiary fourths: 12 color, 1 per chromatic scale note-name. Straight Fourths tuning.
5) WhiteBlack Fourths: Two colors as on the Piano. Straight Fourths tuning starting at a low Open E natural.

Using the proper MIDI commands, select the desired Tuning Map and then use the SetToBackground command.

NOTE-OFFS

The Clipper deals with note-off messages according to the mode of operation.

In normal run-mode, depending on assertion of the Enable-Background command, a Note-off corresponds to either the selection of Color-#0, or the pixel value of a previously set background. Color-#0 is normally Black(OFF), however any other color may be set into the Color#0 location of the Clipper by loading a new color map.

Fade-time: when the Fade-time value is set to >0, the MIDI Note-Off triggers a Fade-timer for each pixel location. When the Fade-timer expires the lit LED will extinguish.

Latch-Mode*:* When Latch-Mode is set the Note-Offs are ignored.

LED Fingerboard Command Specification

Special MIDI-Note Functions on Channel 16 :

[Notice that the minimum velocity value available for your use is the number 1. Velocity = 0 is reserved for the Note-Off function.]

Command	#	Comments
CLEAR FRAME	127	Ignore velocity, clears to 0 unless clear to background mode set
INIT ZONES	126	ignore velocity
REFRESH	125	ignore velocity
ENABLE ZONE	124	velocity = ZONE 1 - 31, bit6 1 = enable 1 = disable (Displays Zone lighting. Bring to foreground)
LEFTRIGHT	123	velocity 2 = LEFT, 1 = RIGHT, mirror about the Y-axis
SHIFT RIGHT	122	velocity offset 0:no effect, =>1:Shift Right one position Bit6=0: Rotate (Wrap) image, Bit6=1: Do not wrap image. Rotate Right: vel=>1; Shift Right: vel =>64D
SHIFT LEFT	121	velocity offset 0:no effect, =>1:Shift Left one position Bit6=0: Rotate (Wrap) image, Bit6=1: Do not wrap image. Rotate Left: vel=>1; ShiftLeft: vel =>64D
SCROLL UP	120	velocity offset: 0:no effect, =>1:Shift Up one position Bit6=0: Rotate (Wrap) image, Bit6=1: Do not wrap image. Rotate Up: vel=>1; Shift Up: vel =>64D
SCROLL DN	119	velocity offset: 0:no effect, =>1:Shift Down one position Bit6=0: Rotate (Wrap) image, Bit6=1: Do not wrap image. Rotate Down: vel=>1; Shift Down: vel =>64D
BLINK ON(upper 16)	118	velocity + 128 = led#
BLINK ON(lower 128)	117	velocity = led#
BLINK OFF(upper 16)	116	velocity + 128 = led#
BLINK OFF(lower 128)	115	velocity = led#
BRIGHT(upper 16)	114	velocity + 128 = led#
BRIGHT(lower 128)	113	velocity = led#
DIM(upper 16)	112	velocity + 128 = led#
DIM(lower 128)	111	velocity = led#

RedBox Scroll Up	110	Velocity: Ignore - not used by ledfb firmware, talks to Ableton
RedBox Scroll Down	109	Velocity: Ignore - not used by ledfb firmware, talks to Ableton
RedBox Scroll Right	108	Velocity: Ignore - not used by ledfb firmware, talks to Ableton
RedBox Scroll Left	107	Velocity: Ignore - not used by ledfb firmware, talks to Ableton
- Unused -	106	- nused -
REVERSE LEDS	105	Velocity: Ignore , Toggles the LED states
TABLE TOP GUITAR	104	Mirror about X-axis. Velocity 7F=Tabletop: body is low LED number Velocity 00=Guitar: head is low LED number
Set Char (lower 128)	103	velocity = text character in lower 128 of ASCII table
Set Char (upper 128)	102	velocity + 128 = text character in upper 128 of ASCII
SET CHAR DIRECTION	101	velocity sets the direction Chars scroll from, 0 from head, >0 from body
SELECT SCRIPT [Available scripts are listed in a table below.]	100	Velocity selects a Script, 0 aborts. A Script is an embedded macro that can display images, text, and/or lighting effects. The opening splash screen is a Script. This, or any, script is aborted by asserting Select-Script with a value=0.
ENABLE BACKGROUND	99	Enable Background: velocity 1 = clear to black, velocity 2 = Enable background. When an LED is turned on, then turned off according to some dynamic function, the off-state may be either an image saved to the background or simply black.
SET COLOR MAP	98	velocity is color map number
SELECT IMAGE	97	velocity is the number of the image to display. Image# 0-31 are RAM, images 32-128 are ROM based images
SET STRING 6	96	velocity = color index
SET STRING 5	95	velocity = color index
SET STRING 4	94	velocity = color index
SET STRING 3	93	velocity = color index
SET STRING 2	92	velocity = color index
SET STRING 1	91	velocity = color index
Display Tuning Map	90	velocity = zone number (???)
Send To Background	89	send the foreground to the background
Set Tuning	88	Pre-defined Tuning Maps are stored in ROM. The full list is provided below. Velocity = Tuning map number. Guitar, Piano, etc. 1=ClearTuning, which clears the background. The full list is described on Page 12.

SCROLL-IMAGE	87	A selected image is scrolled over the short dimension of the fingerboard. Velocity = step timing for the Scroll delay. (0=?)
SET FOREGROUND COLOR	86	Velocity = color map index# of the foreground color.
SET BACKGROUND COLOR	85	Velocity = color map index# of the background color.
Rotate RedBox 90°	84	Velocity: Ignore - not used by ledfb firmware, talks to Ableton. Rotates the Ableton RedBox
SYNC	83	Reset the Blink timer and other internal step-timers.

MIDI-Note Messages on Channel 16 continued:

OFFSET FOREGOUND COLOR INDEX	82	Velocity is added to all color selections for indexed-color pixels, i.e., foreground colors. If the resulting color-index exceeds the length of the color map, the value wraps around to the beginning of the table.
SET FRETMARKS OVERLAY COLOR	81	Velocity = color for the Fretmarks
ENABLE FRETMARKS OVERLAY	80	Velocity = 1 disables the Fretmarks. Velocity = 2 enables the Fretmarks

MIDI-CC Messages on Channel 16 (Channel 15- Unused):

Latch LEDs	64	Off < 65, On > 64; Once set, all LEDs will remain lit until the command is reset.

MIDI-CC Messages used by the LED fingerboard. Only on channel 16

BRIGHTNESS	7	Scale global brightness from Master Volume, Channel 16
SCALE RED	20	Adjust the brightness for all Red LEDs on the
SCALE GREEN	21	Adjust the brightness for all Green LEDs on the
SCALE BLUE	22	Adjust the brightness for all Blue LEDs on the
SCALE SATURATION	23	0-127; Scales the intensity of all colors. Applies to images, tuning maps, and foreground colors.
VU String 1 *	24	Fires all of the LEDs on String #1 in sequence (Data value= length of the lit-string. All LEDs are lit from Fret1 to the scaled

		data-level) CC-data values of 0-127 scale from no lights to all 24 columns lit. All LEDs will light from Fret1 up to the scaled data-value. Each Fret (column) corresponds to a data increment of a little more than 5.
VU String 2	25	Fires all of the LEDs on String #2 (in sequence, value= length) lights sequential number of frets according to MIDI data 0-127
VU String 3	26	Fires all of the LEDs on String #3 (in sequence, value= length) lights sequential number of frets according to 0-127
VU String 4	27	Fires all of the LEDs on String #4 (in sequence, value= length) lights sequential number of frets according to 0-127
VU String 5	28	Fires all of the LEDs on String #5 (in sequence, value= length) lights sequential number of frets according to 0-127
VU String 6	29	Fires all of the LEDs on String #6 (in sequence, value= length) lights sequential number of frets according to 0-127
VU All Strings	30	Fires all of the LEDs on all Strings (in sequence, value= length) lights sequential number of frets according to 0-127
VU Fret 1 **	31	Lights LEDs sequentially up fret 1 like a VU meter. CC-data values of 0-127 scale from no lights to all 6 rows lit in 12 increments. From bottom to top: Rows 1-4= green, Row5=yellow, Row6=Red. Each key lights at half-brightness for the lower half of its own range. Any given data value will light all keys from the bottom to the scaled level. Each lighting increment corresponds to a data increment of about 10 and a half. For example, a datum110 (decimal) lights Row6 (Red) at half-brightness, Row5 (Yellow) full-brightness, and Rows4-1 (Green) full-brightness.
VU Fret 2	32	Lights LEDs sequentially up fret 2 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 3	33	Lights LEDs sequentially up fret 3 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 4	34	Lights LEDs sequentially up fret 4 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 5	35	Lights LEDs sequentially up fret 5 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 6	36	Lights LEDs sequentially up fret 6 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and

		color.
VU Fret 7	37	Lights LEDs sequentially up fret 7 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 8	38	Lights LEDs sequentially up fret 8 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 9	39	Lights LEDs sequentially up fret 9 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 10	40	Lights LEDs sequentially up fret 10 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 11	41	Lights LEDs sequentially up fret 11 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 12	42	Lights LEDs sequentially up fret 12 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 13	43	Lights LEDs sequentially up fret 13 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 14	44	Lights LEDs sequentially up fret 14 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 15	45	Lights LEDs sequentially up fret 15 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 16	46	Lights LEDs sequentially up fret 16 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 17	47	Lights LEDs sequentially up fret 17 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 18	48	Lights LEDs sequentially up fret 18 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 19	49	Lights LEDs sequentially up fret 19 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 20	50	Lights LEDs sequentially up fret 20 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and

		color.
VU Fret 21	51	Lights LEDs sequentially up fret 21 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 22	52	Lights LEDs sequentially up fret 22 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 23	53	Lights LEDs sequentially up fret 23 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
VU Fret 24	54	Lights LEDs sequentially up fret 24 like a VU meter. Value of 0-127 scales from no lights to all lit, fading in brightness and color.
LATCH MODE	64	(MIDI Sustain) When Latch Mode is On, incoming Note-Off messages will be ignored rather than turning off their intended LEDs.Velocity < 65 = Off, velocity > 64 = On.
SET-SCRIPT-CLOCK-COARSE	75	Sets the coarse timing delay for clocked script-looping functions like the Swirl effect in 10 millisecond increments. Delay = velocity x .01 seconds.
SET-SCRIPT-CLOCK-FINE	76	Sets the fine timing for clocked script-looping functions like the Swirl effect in 0.1 millisecond increments. Delay = velocity x .0001 seconds. The coarse and fine values are additive.
STORE-RAM-IMAGE	77	The dynamic LED data frame buffer is copied to a RAM image location. Velocity = Image number (v = 0-9)

APPENDICES AND CHARTS

Embedded lighting Scripts:

0: Null script
1: Abort Script. Aborts any active script.
2: Grand Demo. Combines several scripts and the Starr Labs sign-on screen.
3: Product_ID. Current Clipper OS version#.
4: Null
5: Spiral Effect. Single shot event.
6: Spiral Script. Loops the spiral effect.
7: Swirl Effect. Single shot event.
8: Swirl Script. Loops the swirl effect.
9: Boustrofedon. Serpentine effect.
10: Boustrofedon Script. Loops the single shot event.
11: Show current color map.
12: Scroll RAM Images.
13: Scroill ROM images.
14: Cylon script

Embedded Color Maps [127 color locations per map]

0: Null Map
1: Ableton Live color map. 60 colors. Duplicated to (mostly) fill the map.
2: Spectral color map. Full color in the traditional rainbow sequence.
3: Tertiary color map. The basic 12 colors resulting from the simple combinations of the R,G, and B LEDs.
4: Velocity color map. A four-color map to clearly distinguish MIDI velocity levels.

Setting up Live in the Mac to work with the Clipper

1. Load Live
2. Load or create your session clips and other EFX mappings
3. Copy the Ztar-Live Python script into your Control Surfaces folder.
4. Go to " LIVE > Preferences" and select the "Ztar-MIDI" as your controller.
5. In "MIDI Input" tab, select "Starr Labs MIDI"
6. In "MIDI I Output" tab, select "Starr Labs MIDI"

Operation Notes and Bugs:

The Ableton Live color-map from your computer is not a contiguous spectral array of colors but is pretty random in its color assignments. When these colors are selected by sending ordinary MIDI note-messages the colors may appear random. A linear spectral map is included in the Clipper.

Sometimes when switching between songs with different RedBox dimensions some of the LEDs won't refresh. Usually pressing the song selection button a couple extra times will remedy this.

Sometimes your MIDI Merge setting will get messed up even though you don't remember changing them. This will become apparent if you stop seeing visual feedback on the fingerboard LEDs. Double check your Merge settings if you lose communication from Live to your Clipper.

More:

Index:

Bugs 21
Clip Array 3
Clip Lighting and Visual Feedback 4
Color-Zone Sysex Message format 13
Moving the RedBox 11
MIDI Note Lighting functions 16
MIDI Continuous Control Lighting functions 17
Note-Offs 14
Scene Launch Keys 3
Tuning Maps 14