



OSC GUIDE

Communication protocol and commands for the for the em64 Eigenmike® array

Date: 4/2/2026

**mh acoustics LLC
25A Summit Ave
Summit, NJ 07901**

**Phone: 908-294-1721
www.mhacoustics.com**

*This document is the property of mh acoustics LLC and is Confidential and Proprietary.
© 2026 mh acoustics LLC. All rights reserved.*

Document Change Notes

Version	Description	Date	Changed by
0.1.0	Initial version.	4/2/26	slb

Contents

1.0 Introduction	4
2.0 Tools	5
3.0 Updating the em64 Eigenmike	6
4.0 OSC Messages	8
4.1 System Messages	9
4.1.1 Serial Number	9
4.1.2 LEDs	10
4.2 Audio Messages	11
4.2.1 Boost	11
4.2.2 PGA.....	12

1.0 Introduction

Open Sound Control (OSC) is a communication protocol commonly used to exchange data and messages between computer applications and sound and music hardware on a shared network. It was developed at the Berkeley CNMAT lab and has become a widely adopted standard in audio applications and hardware.

The em64 Eigenmike® microphone array supports OSC communication for control and monitoring of features such as PGA level and LED color. This document describes the steps required to upgrade the em64 Eigenmike firmware to support OSC, along with the OSC messages available for use.

More information about the OSC protocol can be found at:

<https://ccrma.stanford.edu/groups/osc/index.html>

2.0 Tools

To transmit and receive OSC messages to and from the em64 Eigenmike, a number of third-party software applications exist. Some of these allow OSC messages to be exchanged at a somewhat low-level (e.g. type specifiers required), whereas others offer a slightly higher level interface.

- ocsend (via [liblo](#))
- [python-osc](#)
- [Protokol](#)
- [Max/MSP](#)
- [TouchOSC](#)
- Matlab ([oscmex](#))
- [oscpack](#) (see examples)

This list of third-party software is a small subset of recommendations for working with OSC and the em64 Eigenmike. A more exhaustive list of hardware and software applications that support OSC can be found at:

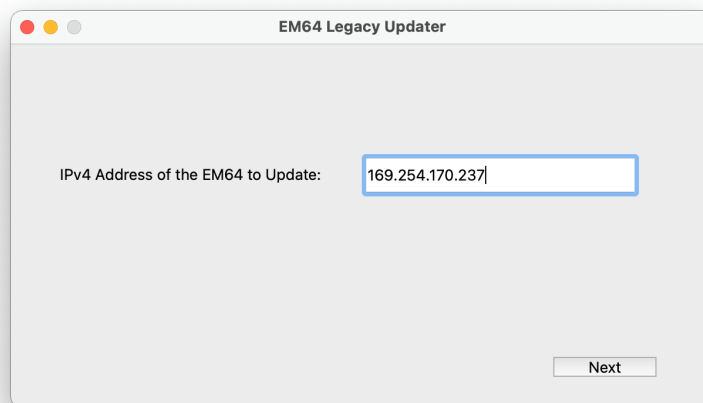
<https://ccrma.stanford.edu/groups/osc/page-list.html#implementations>

3.0 Updating the em64 Eigenmike

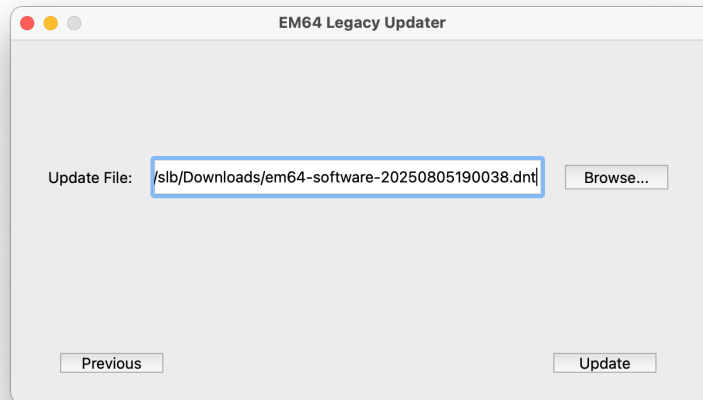
OSC support requires em64 Eigenmike software version **20250805190038** or later. To check the current software version, open the “About EigenStudio 3” menu item in EigenStudio 3 while the mic is live and connected. The dialog box will display the current software version. If you have a recent version that supports OSC you can skip the remainder of this section and go the section 4.0 .

To get started with the update, first download the em64 Legacy Updater application for macOS or Windows. Launch the em64 Legacy Updater Application.

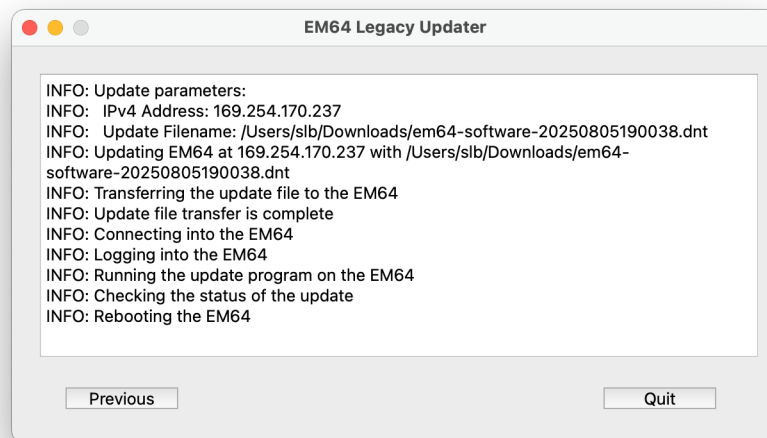
1. Enter the IP address of the em64 you wish to update. The IP address is displayed in the upper left info area in the “Live” tab in EigenStudio 3. It can also be found on the “Device Info” tab of the Dante Controller app.



2. Browse to select the .dnt file included with the Updater application (e.g.20250805190038.dnt).



3. Click update and keep an eye on the em64's LED.
4. About a minute after the message "Rebooting the EM64" the blue LED on the em64 will flash and then become solid again. Once the LED is solid the update is complete.



5. Click Quit to close the updater program.
6. You can verify the update was successful by again opening the "About EigenStudio 3" menu item in EigenStudio 3 while the mic is live and connected.
7. Note: If no mic signals are present after the update, you may need to power cycle your em64 Eigenmike.

4.0 OSC Messages

The OSC messages listed in the subsequent sections can be utilized by users of the em64 Eigenmike, either in custom applications or via one of the many third-party software programs that support sending and receiving OSC messages.

Messages are sent over the network to the Eigenmike at its IP address, which can be found in the upper left info area in the "Live" tab in EigenStudio 3, as well as on the "Device Info" tab of the Dante Controller app. Messages **sent** to the em64 Eigenmike should be directed at **port 6791**. Messages **received** from the em64 Eigenmike will be on **port 6799**. An OSC listener application should open and monitor this port.

Note only one application can bind to the listening port at a time. This is an inherent limitation of OSC/UDP. If EigenStudio 3 is opened first, it will bind to the listening port and prevent any other applications on the same computer from binding to the same port.

OSC messages typically follow the format:

[Address Pattern] [Type Tag String] [Arguments...]

[Address Pattern] specifies the message destination (e.g. /1/audio/pga_gain)

[Type Tag String] specifies the data types for the argument list. Type tags supported are:

- **i**: 32-bit integer
- **f**: 32-bit floating point number
- **s**: Null-terminated string
- **b**: Binary data blob.

[Arguments...] is a list of the argument values to be sent or received.

Specific to the em64, the first argument of all OSC messages sent to the device will be a user-defined integer value specifying a transaction ID (<id>).

When *querying* information from the em64, this id will be the only argument. The response from the em64 will contain a matching transaction ID, along with additional argument values reflecting the information requested.

When sending messages to the em64 to *set* values, the argument list will contain the transaction ID followed by values specific to each message (see below). The response from the em64 will contain a matching transaction ID, along with an additional 64-bit integer value reflecting success (0) or error (non-zero value).

For OSC commands that do not match any known supported address and/or syntax, a response with address pattern /unknown_osc_message will be returned.

For example, an OSC message to query the em64 Eigenmike's PGA level might look like (see details on the PGA level command below):

```
/1/audio/pga_gain ,i 1
```

Note that the exact syntax of the message will vary according to the tool you are using for OSC communication. For example, when using the *oscsend* application on the OS's command line, the above message might be sent with the command:

```
> oscsend 169.254.170.237 6791 /1/audio/pga_gain i 1
```

More information about the formatting of OSC messages can be found in the official OSC specification(s) at:

https://ccrma.stanford.edu/groups/osc/spec-1_0.html

<https://ccrma.stanford.edu/groups/osc/files/2009-NIME-OSC-1.1.pdf>

4.1 System Messages

4.1.1 Serial Number

The serial number of the Eigenmike can be queried at the OSC address `/1/sys/serial_number`. To inquire the serial number, send an OSC message to the em64 Eigenmike along with a transaction id type (int) and `<id>` value (user defined).

```
TX message: /1/sys/serial_number ,i <id>
```

The em64 Eigenmike will respond by transmitting an OSC message with the same transaction id and a string containing the microphone `<name>` in the format "mh-em64-000xyz" where xyz is the serial number.

```
RX message: /1/sys/serial_number ,is <id> <name>
```

4.1.2 LEDs

The front and rear LEDs on the em64 Eigenmike can also be controlled via OSC. Both the color and brightness can be adjusted. To set the LEDs, send an OSC message to the em64 Eigenmike along with transaction id type (int), RGB state and value types (int, int, int, int, int, int, int, int, int, int, int, int), blink rate type (float) followed by an <id> value (user defined) and RGB pairs of LED state and brightness for the rear LED and the front LED. The final argument defines the blink rate when the LED is in blinking state.

TX Message:

```
/1/sys/leds ,iiiiiiiiiiiiif <id>  
<rear_red_state> <rear_red_pwm>  
<rear_green_state> <rear_green_pwm>  
<rear_blue_state> <rear_blue_pwm>  
<front_red_state> <front_red_pwm>  
<front_green_state> <front_green_pwm>  
<front_blue_state> <front_blue_pwm>  
<blink_rate>
```

The LED state value can be specified as:

- 0:** LED Off
- 2:** LED On
- 3:** LED Blinking

The LED RGB PWM values range from 0-32.

To set the default blue state in both LEDs, the TX message would be:

```
/1/sys/leds ,iiiiiiiiiiiiif 1  
0 0 0 0 2 15  
0 0 0 0 2 15  
1.04
```

4.2 Audio Messages

4.2.1 Boost

The 15dB boost circuit in the em64 Eigenmike can be queried or set via OSC at the address `/1/audio/boost`. For the best SNR performance, boost should only be disabled in high SPL environments. Note that in the EigenStudio 3 application, this setting is referred to as a 15dB “pad” and is the opposite of the boost setting (i.e. boost off = pad on; boost on = pad off).

To query the boost value, send an OSC message to the em64 Eigenmike along with a transaction id type (int) and `<id>` value (user defined).

TX message: `/1/audio/boost ,i <id>`

The em64 Eigenmike will respond by transmitting an OSC message with the same transaction id and the boost state value.

RX message: `/1/audio/boost ,iii <id> <id2> <boost_value>`

To enable/disable boost, send an OSC message to the em64 Eigenmike along with transaction id type (int), and boost type (int) and value (0=off, 1=on)

TX message: `/1/audio/boost ,ii <id> <boost_value>`

