HARDWARE COMMAND MANUAL

NIO222

Network I/O mixing breakout box Dante™/AES67 - 2 x Mic/Line inputs + 2 x Balanced Line outputs

Table of contents

NIO222 Commands List

ASCII Commands (Nio222 V1.1.0)

This is the list of ASCII Commands supported by this device. An ASCII command always follows the same structure:

```
#|Destination|Source|Type^Target^Command|Arguments|CRC|CRLF
```

This format uses 3 separator characters for different levels of separating each value in the message:

- Message separator '|':

  This separates a message in 7 blocks (if you include the start '#' and end <CRLF>)

- Block separator '^':

  This splits a block into logical elements. This is used to split the command in the message type, 'command' and 'target'

- Value separator '>':

  This splits up a logical single value in their primitives, for example: a target consists of a channel type and a channel index, split by '>'

Messages are Case sensitive, if the example shows the text in uppercase, this should always be uppercase!

Destination

The target device. This consists of 2 parts: `Device>Address`.

- **Device**

  This is the device type: **NIO222**

- **Address**

  This is the user configurable device address, default: **1**. You can also leave this field empty, this results in all Nio222 devices that receive this command to respond.

Examples	Destination
default destination	`NIO222>1`
broadcast to all Nio222 devices	`NIO222`

Device Matching

If the device type or address does not match, the message will be ignored. Device Address 0 is a special address and will always match (this can be seen as a broadcast)

Destination	Device address: NIO222>2	Remarks
NIO222>2	Destination Matches Device	this is an exact match
NIO222>1	Message ignored	the destination address does not match
NIO222	Destination Matches Device	the destination address will always match
NIO222>0	Destination Matches Device	Equivalent to NIO222
CLIENT>2	Message ignored	the device type does not match
CLIENT	Message ignored	the device type does not match

Source (optional)

The source address is optional when sending, but the device will always fill this field with its own address.

Examples	Sent message	Response message NIO222>2
broadcast to a Nio222	#|NIO222||...|<CRLF>	#||NIO222>2|...|<CRLF>
send to a specific Nio222	#|NIO222>2||...|<CRLF>	#||NIO222>2|...|<CRLF>
use a source address in the request message	#|NIO222|CLIENT>1|...|<CRLF>	#|CLIENT>1|NIO222>2|...|<CRLF>

Type

The type explains what the message wants to do. There are 3 supported message types:

Type	From	To	Explanation
SET_REQ	CLIENT	Nio222	Change a setting in the Nio222
GET_REQ	CLIENT	Nio222	Request the current status of a setting in the Nio222
GET_RSP	Nio222	CLIENT	Response to either a GET_REQ or SET_REQ, if the request was valid

Command, Target, Arguments

These 3 parameters are explained together, because they influence each other. The command dictates the meaning of the argument, while the target distinguishes which exact setting you want to change. the target can also influence the valid range of the argument.

Some commands (like the mixer) can have a range arguments (for the mixer: all mixer volumes are an individual argument). In this case, the argument looks like: `idx>val[^idx2>val2]`, where the part in between the brackets `[]` can appear 0 or more times.

- idx, idx2, ...: the argument index
- val, val2, ...: the value at the specified index

This device supports special `ALL_*` commands that allow you to set multiple values in a single command. They look similar to their single counterparts, but the *Target* starts with `ALL_`. The *Argument* is an array of elements, separated by "block separators", instead of a single value.

For example: if there's a device that supports grouping 2 MUTE commands, the following can be shortened:

```
#|NIO222>1||SET_REQ^TARGET>1^MUTE|TRUE|U|<CRLF>
#|NIO222>1||SET_REQ^TARGET>2^MUTE|TRUE|U|<CRLF>
```

becomes

```
#|NIO222>1||SET_REQ^ALL_TARGET^MUTE|TRUE^TRUE|U|<CRLF>
```

Note: For backwards compatibility reasons there might be gaps in the arguments sent, it is important to leave these gaps in the command you build! A gap is created by putting 2 block separators next to each other: "^^".

You can leverage this as well, if there are parts of this grouped command you don't want to change. You can then leave this value empty like the gaps, and you can only change the values you want, leaving the rest as is.

- `TRUE^^FALSE`: Set the first value to true, leave the second value as is, set the third value to false.

You can also skip the last elements in the list, if you don't want to change them. For example: the command groups 4 volumes together, all. If you only want to change the first two values, you have 2 options for the "Arguments" field:

- `-15^-30^^`
- `-15^-30`

Both are equivalent, but in the first example you explicitly define the third and fourth values as "gaps", in the second example they are implicitly defined as gaps.

BT_PAIR

this command is used to enable the pairing process in a bluetooth device

Argument (enable): enable the pairing process

Target	Argument	Example (default value)
INPUT_BLUETOOTH>1>BLUETOOTH>1	options: TRUE,FALSE	`#|NIO222>1||SET_REQ^INPUT_BLUETOOTH>1>BLUETOOTH>1^BT_PAIR|FALSE|U|<CRLF>`

BT_DISCONNECT

this command is used to disconnect the connected BT device

Argument (enable): disconnect the connected device

Target	Argument	Example (default value)
INPUT_BLUETOOTH>1>BLUETOOTH>1	options: TRUE,FALSE	`#|NIO222>1||SET_REQ^INPUT_BLUETOOTH>1>BLUETOOTH>1^BT_DISCONNECT|FALSE|U|<CRLF>`

VOLUME

Set a single Volume in dB

Argument (volume): the requested Volume in dB

Target	Argument	Example (default value)
INPUT_LINE>1>VOLUME>1	min: -90, max: 0	`#|NIO222>1||SET_REQ^INPUT_LINE>1>VOLUME>1^VOLUME|0|U|<CRLF>`
INPUT_LINE>2>VOLUME>1	min: -90, max: 0	`#|NIO222>1||SET_REQ^INPUT_LINE>2>VOLUME>1^VOLUME|0|U|<CRLF>`
INPUT_BLUETOOTH>1>VOLUME>1	min: -90, max: 0	`#|NIO222>1||SET_REQ^INPUT_BLUETOOTH>1>VOLUME>1^VOLUME|0|U|<CRLF>`
INPUT_BLUETOOTH>2>VOLUME>1	min: -90, max: 0	`#|NIO222>1||SET_REQ^INPUT_BLUETOOTH>2>VOLUME>1^VOLUME|0|U|<CRLF>`
INPUT_DANTE>1>VOLUME>1	min: -90, max: 0	`#|NIO222>1||SET_REQ^INPUT_DANTE>1>VOLUME>1^VOLUME|0|U|<CRLF>`
INPUT_DANTE>2>VOLUME>1	min: -90, max: 0	`#|NIO222>1||SET_REQ^INPUT_DANTE>2>VOLUME>1^VOLUME|0|U|<CRLF>`
INPUT_DANTE>3>VOLUME>1	min: -90, max: 0	`#|NIO222>1||SET_REQ^INPUT_DANTE>3>VOLUME>1^VOLUME|0|U|<CRLF>`
INPUT_DANTE>4>VOLUME>1	min: -90, max: 0	`#|NIO222>1||SET_REQ^INPUT_DANTE>4>VOLUME>1^VOLUME|0|U|<CRLF>`
OUTPUT_DANTE>1>VOLUME>1	min: -90, max: 0	`#|NIO222>1||SET_REQ^OUTPUT_DANTE>1>VOLUME>1^VOLUME|0|U|<CRLF>`
OUTPUT_DANTE>2>VOLUME>1	min: -90, max: 0	`#|NIO222>1||SET_REQ^OUTPUT_DANTE>2>VOLUME>1^VOLUME|0|U|<CRLF>`
OUTPUT_DANTE>3>VOLUME>1	min: -90, max: 0	`#|NIO222>1||SET_REQ^OUTPUT_DANTE>3>VOLUME>1^VOLUME|0|U|<CRLF>`
OUTPUT_DANTE>4>VOLUME>1	min: -90, max: 0	`#|NIO222>1||SET_REQ^OUTPUT_DANTE>4>VOLUME>1^VOLUME|0|U|<CRLF>`
OUTPUT_LINE>1>VOLUME>1	min: -90, max: 0	`#|NIO222>1||SET_REQ^OUTPUT_LINE>1>VOLUME>1^VOLUME|0|U|<CRLF>`
OUTPUT_LINE>2>VOLUME>1	min: -90, max: 0	`#|NIO222>1||SET_REQ^OUTPUT_LINE>2>VOLUME>1^VOLUME|0|U|<CRLF>`

Grouped Commands

ALL_IN

example (default value): `#|NIO222>1||SET_REQ^ALL_IN^VOLUME|0^0^^^0^0^0^0^0^0|U|<CRLF>`

Index	Target
1	INPUT_LINE>1>VOLUME>1
2	INPUT_LINE>2>VOLUME>1
3	*Reserved*
4	*Reserved*
5	INPUT_BLUETOOTH>1>VOLUME>1
6	INPUT_BLUETOOTH>2>VOLUME>1
7	INPUT_DANTE>1>VOLUME>1
8	INPUT_DANTE>2>VOLUME>1
9	INPUT_DANTE>3>VOLUME>1
10	INPUT_DANTE>4>VOLUME>1

ALL_OUT

example (default value): `#|NIO222>1||SET_REQ^ALL_OUT^VOLUME|0^0^0^0^0^0^^|U|<CRLF>`

Index	Target
1	OUTPUT_DANTE>1>VOLUME>1
2	OUTPUT_DANTE>2>VOLUME>1
3	OUTPUT_DANTE>3>VOLUME>1
4	OUTPUT_DANTE>4>VOLUME>1
5	OUTPUT_LINE>1>VOLUME>1
6	OUTPUT_LINE>2>VOLUME>1
7	*Reserved*
8	*Reserved*

MUTE

mute an audio channel

Argument (enabled): is the audio channel muted

Target	Argument	Example (default value)
INPUT_LINE>1>VOLUME>1	options: TRUE,FALSE	`#|NIO222>1||SET_REQ^INPUT_LINE>1>VOLUME>1^MUTE|FALSE|U|<CRLF>`
INPUT_LINE>2>VOLUME>1	options: TRUE,FALSE	`#|NIO222>1||SET_REQ^INPUT_LINE>2>VOLUME>1^MUTE|FALSE|U|<CRLF>`
INPUT_BLUETOOTH>1>VOLUME>1	options: TRUE,FALSE	`#|NIO222>1||SET_REQ^INPUT_BLUETOOTH>1>VOLUME>1^MUTE|FALSE|U|<CRLF>`
INPUT_BLUETOOTH>2>VOLUME>1	options: TRUE,FALSE	`#|NIO222>1||SET_REQ^INPUT_BLUETOOTH>2>VOLUME>1^MUTE|FALSE|U|<CRLF>`
INPUT_DANTE>1>VOLUME>1	options: TRUE,FALSE	`#|NIO222>1||SET_REQ^INPUT_DANTE>1>VOLUME>1^MUTE|FALSE|U|<CRLF>`
INPUT_DANTE>2>VOLUME>1	options: TRUE,FALSE	`#|NIO222>1||SET_REQ^INPUT_DANTE>2>VOLUME>1^MUTE|FALSE|U|<CRLF>`
INPUT_DANTE>3>VOLUME>1	options: TRUE,FALSE	`#|NIO222>1||SET_REQ^INPUT_DANTE>3>VOLUME>1^MUTE|FALSE|U|<CRLF>`
INPUT_DANTE>4>VOLUME>1	options: TRUE,FALSE	`#|NIO222>1||SET_REQ^INPUT_DANTE>4>VOLUME>1^MUTE|FALSE|U|<CRLF>`
OUTPUT_DANTE>1>VOLUME>1	options: TRUE,FALSE	`#|NIO222>1||SET_REQ^OUTPUT_DANTE>1>VOLUME>1^MUTE|FALSE|U|<CRLF>`
OUTPUT_DANTE>2>VOLUME>1	options: TRUE,FALSE	`#|NIO222>1||SET_REQ^OUTPUT_DANTE>2>VOLUME>1^MUTE|FALSE|U|<CRLF>`
OUTPUT_DANTE>3>VOLUME>1	options: TRUE,FALSE	`#|NIO222>1||SET_REQ^OUTPUT_DANTE>3>VOLUME>1^MUTE|FALSE|U|<CRLF>`
OUTPUT_DANTE>4>VOLUME>1	options: TRUE,FALSE	`#|NIO222>1||SET_REQ^OUTPUT_DANTE>4>VOLUME>1^MUTE|FALSE|U|<CRLF>`
OUTPUT_LINE>1>VOLUME>1	options: TRUE,FALSE	`#|NIO222>1||SET_REQ^OUTPUT_LINE>1>VOLUME>1^MUTE|FALSE|U|<CRLF>`
OUTPUT_LINE>2>VOLUME>1	options: TRUE,FALSE	`#|NIO222>1||SET_REQ^OUTPUT_LINE>2>VOLUME>1^MUTE|FALSE|U|<CRLF>`

Grouped Commands

ALL_IN

example (default value): `#|NIO222>1||SET_REQ^ALL_IN^MUTE|FALSE^FALSE^^^FALSE^FALSE^FALSE^FALSE^FALSE^FALSE|U|<CRLF>`

Index	Target
1	INPUT_LINE>1>VOLUME>1
2	INPUT_LINE>2>VOLUME>1
3	*Reserved*
4	*Reserved*
5	INPUT_BLUETOOTH>1>VOLUME>1
6	INPUT_BLUETOOTH>2>VOLUME>1
7	INPUT_DANTE>1>VOLUME>1
8	INPUT_DANTE>2>VOLUME>1
9	INPUT_DANTE>3>VOLUME>1
10	INPUT_DANTE>4>VOLUME>1

ALL_OUT

example (default value): `#|NIO222>1||SET_REQ^ALL_OUT^MUTE|FALSE^FALSE^FALSE^FALSE^FALSE^FALSE^^|U|<CRLF>`

Index	Target
1	OUTPUT_DANTE>1>VOLUME>1
2	OUTPUT_DANTE>2>VOLUME>1
3	OUTPUT_DANTE>3>VOLUME>1
4	OUTPUT_DANTE>4>VOLUME>1
5	OUTPUT_LINE>1>VOLUME>1
6	OUTPUT_LINE>2>VOLUME>1
7	*Reserved*
8	*Reserved*

MIXER

mixer slider for zones

Argument (volume): mixing volume

<table>
<tr><th>Target</th><th>Argument Index</th><th>Argument</th><th>Example (default value)</th></tr>
<tr><td>OUTPUT_DANTE>1>MIXER>1</td><td>min: 1, max: 16</td><td>min: -90, max: 0</td><td>#|NIO222>1||SET_REQ^OUTPUT_DANTE>1>MIXER>1^MIXER|1>0^2>-90^3>-90^4>-90^5>-90^6>-90^7>-90 <CRLF></td></tr>
<tr><td>OUTPUT_DANTE>2>MIXER>1</td><td>min: 1, max: 16</td><td>min: -90, max: 0</td><td>#|NIO222>1||SET_REQ^OUTPUT_DANTE>2>MIXER>1^MIXER|1>-90^2>0^3>-90^4>-90^5>-90^6>-90^7>-90 <CRLF></td></tr>
<tr><td>OUTPUT_DANTE>3>MIXER>1</td><td>min: 1, max: 16</td><td>min: -90, max: 0</td><td>#|NIO222>1||SET_REQ^OUTPUT_DANTE>3>MIXER>1^MIXER|1>-90^2>-90^3>-90^4>-90^5>0^6>-90^7>-90 <CRLF></td></tr>
<tr><td>OUTPUT_DANTE>4>MIXER>1</td><td>min: 1, max: 16</td><td>min: -90, max: 0</td><td>#|NIO222>1||SET_REQ^OUTPUT_DANTE>4>MIXER>1^MIXER|1>-90^2>-90^3>-90^4>-90^5>-90^6>0^7>-90 <CRLF></td></tr>
<tr><td>OUTPUT_LINE>1>MIXER>1</td><td>min: 1, max: 16</td><td>min: -90, max: 0</td><td>#|NIO222>1||SET_REQ^OUTPUT_LINE>1>MIXER>1^MIXER|1>-90^2>-90^3>-90^4>-90^5>-90^6>-90^7>0^ <CRLF></td></tr>
<tr><td>OUTPUT_LINE>2>MIXER>1</td><td>min: 1, max: 16</td><td>min: -90, max: 0</td><td>#|NIO222>1||SET_REQ^OUTPUT_LINE>2>MIXER>1^MIXER|1>-90^2>-90^3>-90^4>-90^5>-90^6>-90^7>-9 <CRLF></td></tr>
</table>

ROUTE

change the routing of a zone

Argument (input): the input that is selected in that zone. -1 = Mixed (not settable), O = OFF, 1 = input 1 ,...

<table>
<tr><th>Target</th><th>Argument</th><th>Example (default value)</th></tr>
<tr><td>OUTPUT_DANTE>1>MIXER>1</td><td>min: -1, max: 21</td><td>#|NIO222>1||SET_REQ^OUTPUT_DANTE>1>MIXER>1^ROUTE|0|U|<CRLF></td></tr>
<tr><td>OUTPUT_DANTE>2>MIXER>1</td><td>min: -1, max: 21</td><td>#|NIO222>1||SET_REQ^OUTPUT_DANTE>2>MIXER>1^ROUTE|0|U|<CRLF></td></tr>
<tr><td>OUTPUT_DANTE>3>MIXER>1</td><td>min: -1, max: 21</td><td>#|NIO222>1||SET_REQ^OUTPUT_DANTE>3>MIXER>1^ROUTE|0|U|<CRLF></td></tr>
<tr><td>OUTPUT_DANTE>4>MIXER>1</td><td>min: -1, max: 21</td><td>#|NIO222>1||SET_REQ^OUTPUT_DANTE>4>MIXER>1^ROUTE|0|U|<CRLF></td></tr>
<tr><td>OUTPUT_LINE>1>MIXER>1</td><td>min: -1, max: 21</td><td>#|NIO222>1||SET_REQ^OUTPUT_LINE>1>MIXER>1^ROUTE|0|U|<CRLF></td></tr>
<tr><td>OUTPUT_LINE>2>MIXER>1</td><td>min: -1, max: 21</td><td>#|NIO222>1||SET_REQ^OUTPUT_LINE>2>MIXER>1^ROUTE|0|U|<CRLF></td></tr>
</table>

Grouped Commands

ALL_OUT

example (default value): `#|NIO222>1||SET_REQ^ALL_OUT^ROUTE|0^0^0^0^0^0^^|U|<CRLF>`

Index	Target
1	OUTPUT_DANTE>1>MIXER>1
2	OUTPUT_DANTE>2>MIXER>1
3	OUTPUT_DANTE>3>MIXER>1
4	OUTPUT_DANTE>4>MIXER>1
5	OUTPUT_LINE>1>MIXER>1
6	OUTPUT_LINE>2>MIXER>1
7	*Reserved*
8	*Reserved*

CRC

The CRC block is calculated over the message starting from and including the first pipe "|", up to and including the last pipe **before** the CRC Block. These CRC's can ensure message integrity if desired.

<table>
<tr><th>CRC Type</th><th>Configuration</th><th>Format</th><th>Example</th><th>notes</th></tr>
<tr><td>None</td><td>/</td><td>U</td><td>`#|ALL||SET_REQ^INPUT_LINE>1^VOLUME|0|U|<CRLF>`</td><td>'U' means unused</td></tr>
<tr><td>CRC16-ARC</td><td>• input reflected
• output reflected
• polynomial: 0x8005
• initial value: 0x0000
• final exor: 0x0000</td><td>XXXX</td><td>`#|ALL||SET_REQ^INPUT_LINE>1^VOLUME|0|C06C|<CRLF>`</td><td>calculator</td></tr>
<tr><td>CRC32</td><td>• input reflected
• output reflected
• polynomial: 0x04C11DB7
• initial value: 0xFFFFFFFF
• final exor: 0xFFFFFFFF</td><td>XXXXXXXX</td><td>`#|ALL||SET_REQ^INPUT_LINE>1^VOLUME|0|D887125C|<CRLF>`</td><td>calculator</td></tr>
</table>

The examples in the table above are example for calculating the CRC, they may not be a valid command for the Nio222

The CRC can ensure data integrity across unreliable data channels (RS232, RS485), but they are by no means a security measure! If someone has the knowledge and means to maliciously alter a message, correcting the CRC becomes trivial for the attacker. We support different kinds of CRC for maximum flexibility, but we recommend not using any so you do not get a false sense of security.

Stop Bytes

The final 2 characters are denoted as <CRLF>, they mean "Carriage Return, Line Feed" or simply put a new line. Depending on the tool used to create the command, you can have different representations:

- CRLF
- \r\n
- 0x0D 0x0A

We support both CRLF and LF only