# NWP320

## Network input panel - 2 x XLR + 3.5 mm jack + BT

## (4 CH)

## Table of contents

NWP320 Commands List

## ASCII Commands (Nwp320 V1.2.1-RD-96)

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: **NWP320**

- **Address**

  This is the user configurable device address, default: **1**. You can also leave this field empty, this results in all Nwp320 devices that receive this command to respond.

| Examples | Destination |
|---|---|
| default destination | `NWP320>1` |
| broadcast to all Nwp320 devices | `NWP320` |

**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: NWP320>2 | Remarks |
|---|---|---|
| NWP320>2 | Destination Matches Device | this is an exact match |
| NWP320>1 | Message ignored | the destination address does not match |
| NWP320 | Destination Matches Device | the destination address will always match |
| NWP320>0 | Destination Matches Device | Equivalent to NWP320 |
| 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 NWP320>2 |
|---|---|---|
| broadcast to a Nwp320 | #|NWP320||...|<CRLF> | #||NWP320>2|...|<CRLF> |
| send to a specific Nwp320 | #|NWP320>2||...|<CRLF> | #||NWP320>2|...|<CRLF> |
| use a source address in the request message | #|NWP320|CLIENT>1|...|<CRLF> | #|CLIENT>1|NWP320>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 | Nwp320 | Change a setting in the Nwp320 |
| GET_REQ | CLIENT | Nwp320 | Request the current status of a setting in the Nwp320 |
| GET_RSP | Nwp320 | 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:

```
#|NWP320>1||SET_REQ^TARGET>1^MUTE|TRUE|U|<CRLF>
#|NWP320>1||SET_REQ^TARGET>2^MUTE|TRUE|U|<CRLF>
```

becomes

```
#|NWP320>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 | `#|NWP320>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 | `#|NWP320>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_XLR>1>VOLUME>1 | min: -90, max: 0 | `#|NWP320>1||SET_REQ^INPUT_XLR>1>VOLUME>1^VOLUME|0|U|<CRLF>` |
| INPUT_XLR>2>VOLUME>1 | min: -90, max: 0 | `#|NWP320>1||SET_REQ^INPUT_XLR>2>VOLUME>1^VOLUME|0|U|<CRLF>` |
| INPUT_JACK>1>VOLUME>1 | min: -90, max: 0 | `#|NWP320>1||SET_REQ^INPUT_JACK>1>VOLUME>1^VOLUME|0|U|<CRLF>` |
| INPUT_JACK>2>VOLUME>1 | min: -90, max: 0 | `#|NWP320>1||SET_REQ^INPUT_JACK>2>VOLUME>1^VOLUME|0|U|<CRLF>` |
| INPUT_BLUETOOTH>1>VOLUME>1 | min: -90, max: 0 | `#|NWP320>1||SET_REQ^INPUT_BLUETOOTH>1>VOLUME>1^VOLUME|0|U|<CRLF>` |
| INPUT_BLUETOOTH>2>VOLUME>1 | min: -90, max: 0 | `#|NWP320>1||SET_REQ^INPUT_BLUETOOTH>2>VOLUME>1^VOLUME|0|U|<CRLF>` |
| INPUT_DANTE>1>VOLUME>1 | min: -90, max: 0 | `#|NWP320>1||SET_REQ^INPUT_DANTE>1>VOLUME>1^VOLUME|0|U|<CRLF>` |
| INPUT_DANTE>2>VOLUME>1 | min: -90, max: 0 | `#|NWP320>1||SET_REQ^INPUT_DANTE>2>VOLUME>1^VOLUME|0|U|<CRLF>` |
| INPUT_DANTE>3>VOLUME>1 | min: -90, max: 0 | `#|NWP320>1||SET_REQ^INPUT_DANTE>3>VOLUME>1^VOLUME|0|U|<CRLF>` |
| INPUT_DANTE>4>VOLUME>1 | min: -90, max: 0 | `#|NWP320>1||SET_REQ^INPUT_DANTE>4>VOLUME>1^VOLUME|0|U|<CRLF>` |
| OUTPUT_DANTE>1>VOLUME>1 | min: -90, max: 0 | `#|NWP320>1||SET_REQ^OUTPUT_DANTE>1>VOLUME>1^VOLUME|0|U|<CRLF>` |
| OUTPUT_DANTE>2>VOLUME>1 | min: -90, max: 0 | `#|NWP320>1||SET_REQ^OUTPUT_DANTE>2>VOLUME>1^VOLUME|0|U|<CRLF>` |
| OUTPUT_DANTE>3>VOLUME>1 | min: -90, max: 0 | `#|NWP320>1||SET_REQ^OUTPUT_DANTE>3>VOLUME>1^VOLUME|0|U|<CRLF>` |
| OUTPUT_DANTE>4>VOLUME>1 | min: -90, max: 0 | `#|NWP320>1||SET_REQ^OUTPUT_DANTE>4>VOLUME>1^VOLUME|0|U|<CRLF>` |

## Grouped Commands

**ALL_IN**

example (default value): `#|NWP320>1||SET_REQ^ALL_IN^VOLUME|0^0^0^0^0^0^^^0^0^0^0|U|<CRLF>`

| Index | Target |
|---|---|
| 1 | INPUT_XLR>1>VOLUME>1 |
| 2 | INPUT_XLR>2>VOLUME>1 |
| 3 | INPUT_JACK>1>VOLUME>1 |
| 4 | INPUT_JACK>2>VOLUME>1 |
| 5 | INPUT_BLUETOOTH>1>VOLUME>1 |
| 6 | INPUT_BLUETOOTH>2>VOLUME>1 |
| 7 | *Reserved* |
| 8 | *Reserved* |
| 9 | INPUT_DANTE>1>VOLUME>1 |
| 10 | INPUT_DANTE>2>VOLUME>1 |
| 11 | INPUT_DANTE>3>VOLUME>1 |
| 12 | INPUT_DANTE>4>VOLUME>1 |

**ALL_OUT**

example (default value): `#|NWP320>1||SET_REQ^ALL_OUT^VOLUME|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 | *Reserved* |
| 6 | *Reserved* |
| 7 | *Reserved* |
| 8 | *Reserved* |

## MUTE

mute an audio channel

**Argument (enabled)**: is the audio channel muted

| Target | Argument | Example (default value) |
|---|---|---|
| INPUT_XLR>1>VOLUME>1 | options: TRUE,FALSE | `#|NWP320>1||SET_REQ^INPUT_XLR>1>VOLUME>1^MUTE|FALSE|U|<CRLF>` |
| INPUT_XLR>2>VOLUME>1 | options: TRUE,FALSE | `#|NWP320>1||SET_REQ^INPUT_XLR>2>VOLUME>1^MUTE|FALSE|U|<CRLF>` |
| INPUT_JACK>1>VOLUME>1 | options: TRUE,FALSE | `#|NWP320>1||SET_REQ^INPUT_JACK>1>VOLUME>1^MUTE|FALSE|U|<CRLF>` |
| INPUT_JACK>2>VOLUME>1 | options: TRUE,FALSE | `#|NWP320>1||SET_REQ^INPUT_JACK>2>VOLUME>1^MUTE|FALSE|U|<CRLF>` |
| INPUT_BLUETOOTH>1>VOLUME>1 | options: TRUE,FALSE | `#|NWP320>1||SET_REQ^INPUT_BLUETOOTH>1>VOLUME>1^MUTE|FALSE|U|<CRLF>` |
| INPUT_BLUETOOTH>2>VOLUME>1 | options: TRUE,FALSE | `#|NWP320>1||SET_REQ^INPUT_BLUETOOTH>2>VOLUME>1^MUTE|FALSE|U|<CRLF>` |
| INPUT_DANTE>1>VOLUME>1 | options: TRUE,FALSE | `#|NWP320>1||SET_REQ^INPUT_DANTE>1>VOLUME>1^MUTE|FALSE|U|<CRLF>` |
| INPUT_DANTE>2>VOLUME>1 | options: TRUE,FALSE | `#|NWP320>1||SET_REQ^INPUT_DANTE>2>VOLUME>1^MUTE|FALSE|U|<CRLF>` |
| INPUT_DANTE>3>VOLUME>1 | options: TRUE,FALSE | `#|NWP320>1||SET_REQ^INPUT_DANTE>3>VOLUME>1^MUTE|FALSE|U|<CRLF>` |
| INPUT_DANTE>4>VOLUME>1 | options: TRUE,FALSE | `#|NWP320>1||SET_REQ^INPUT_DANTE>4>VOLUME>1^MUTE|FALSE|U|<CRLF>` |
| OUTPUT_DANTE>1>VOLUME>1 | options: TRUE,FALSE | `#|NWP320>1||SET_REQ^OUTPUT_DANTE>1>VOLUME>1^MUTE|FALSE|U|<CRLF>` |
| OUTPUT_DANTE>2>VOLUME>1 | options: TRUE,FALSE | `#|NWP320>1||SET_REQ^OUTPUT_DANTE>2>VOLUME>1^MUTE|FALSE|U|<CRLF>` |
| OUTPUT_DANTE>3>VOLUME>1 | options: TRUE,FALSE | `#|NWP320>1||SET_REQ^OUTPUT_DANTE>3>VOLUME>1^MUTE|FALSE|U|<CRLF>` |
| OUTPUT_DANTE>4>VOLUME>1 | options: TRUE,FALSE | `#|NWP320>1||SET_REQ^OUTPUT_DANTE>4>VOLUME>1^MUTE|FALSE|U|<CRLF>` |

Grouped Commands

**ALL_IN**

example (default value): `#|NWP320>1||SET_REQ^ALL_IN^MUTE|FALSE^FALSE^FALSE^FALSE^FALSE^FALSE^^^FALSE^FALSE^FALSE^FALSE|U|<CRLF>`

| Index | Target |
|---|---|
| 1 | INPUT_XLR>1>VOLUME>1 |
| 2 | INPUT_XLR>2>VOLUME>1 |
| 3 | INPUT_JACK>1>VOLUME>1 |
| 4 | INPUT_JACK>2>VOLUME>1 |
| 5 | INPUT_BLUETOOTH>1>VOLUME>1 |
| 6 | INPUT_BLUETOOTH>2>VOLUME>1 |
| 7 | *Reserved* |
| 8 | *Reserved* |
| 9 | INPUT_DANTE>1>VOLUME>1 |
| 10 | INPUT_DANTE>2>VOLUME>1 |
| 11 | INPUT_DANTE>3>VOLUME>1 |
| 12 | INPUT_DANTE>4>VOLUME>1 |

**ALL_OUT**

example (default value): `#|NWP320>1||SET_REQ^ALL_OUT^MUTE|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 | *Reserved* |
| 6 | *Reserved* |
| 7 | *Reserved* |
| 8 | *Reserved* |

## MIXER

mixer slider for zones

**Argument (volume)**: mixing volume

| Target | Argument Index | Argument | Example (default value) |
|--------|----------------|----------|-------------------------|
| OUTPUT_DANTE>1>MIXER>1 | min: 1, max: 16 | min: -90, max: 0 | `#|NWP320>1||SET_REQ^OUTPUT_DANTE>1>MIXER>1^MIXER|1>0^2>-90^3>-90^4>-90^5>-90^6>-90^7>-90` `<CRLF>` |
| OUTPUT_DANTE>2>MIXER>1 | min: 1, max: 16 | min: -90, max: 0 | `#|NWP320>1||SET_REQ^OUTPUT_DANTE>2>MIXER>1^MIXER|1>-90^2>0^3>-90^4>-90^5>-90^6>-90^7>-90` `<CRLF>` |
| OUTPUT_DANTE>3>MIXER>1 | min: 1, max: 16 | min: -90, max: 0 | `#|NWP320>1||SET_REQ^OUTPUT_DANTE>3>MIXER>1^MIXER|1>-90^2>-90^3>0^4>0^5>-90^6>-90^7>-90^8` `<CRLF>` |
| OUTPUT_DANTE>4>MIXER>1 | min: 1, max: 16 | min: -90, max: 0 | `#|NWP320>1||SET_REQ^OUTPUT_DANTE>4>MIXER>1^MIXER|1>-90^2>-90^3>-90^4>-90^5>0^6>0^7>-90^8` `<CRLF>` |

## 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 ,...

| Target | Argument | Example (default value) |
|--------|----------|-------------------------|
| OUTPUT_DANTE>1>MIXER>1 | min: -1, max: 21 | `#|NWP320>1||SET_REQ^OUTPUT_DANTE>1>MIXER>1^ROUTE|0|U|<CRLF>` |
| OUTPUT_DANTE>2>MIXER>1 | min: -1, max: 21 | `#|NWP320>1||SET_REQ^OUTPUT_DANTE>2>MIXER>1^ROUTE|0|U|<CRLF>` |
| OUTPUT_DANTE>3>MIXER>1 | min: -1, max: 21 | `#|NWP320>1||SET_REQ^OUTPUT_DANTE>3>MIXER>1^ROUTE|0|U|<CRLF>` |
| OUTPUT_DANTE>4>MIXER>1 | min: -1, max: 21 | `#|NWP320>1||SET_REQ^OUTPUT_DANTE>4>MIXER>1^ROUTE|0|U|<CRLF>` |

## Grouped Commands

**ALL_OUT**

example (default value): `#|NWP320>1||SET_REQ^ALL_OUT^ROUTE|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 | *Reserved* |
| 6 | *Reserved* |
| 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.

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

*The examples in the table above are example for calculating the CRC, they may not be a valid command for the Nwp320*

*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