THORLABS

LNC31
Low-Noise Laser Diode Driver with Dual-Channel Temperature Controller

SCPI Programmer’s Reference Manual

Version: 1.0

Date: 25 August 2025

Table of Contents

Chapter 1 Communication Interfaces

The LNC31 device has Ethernet, USB interface and RS-232 interface ports on the back panel of the system. All these three interfaces can be used for SCPI communication.

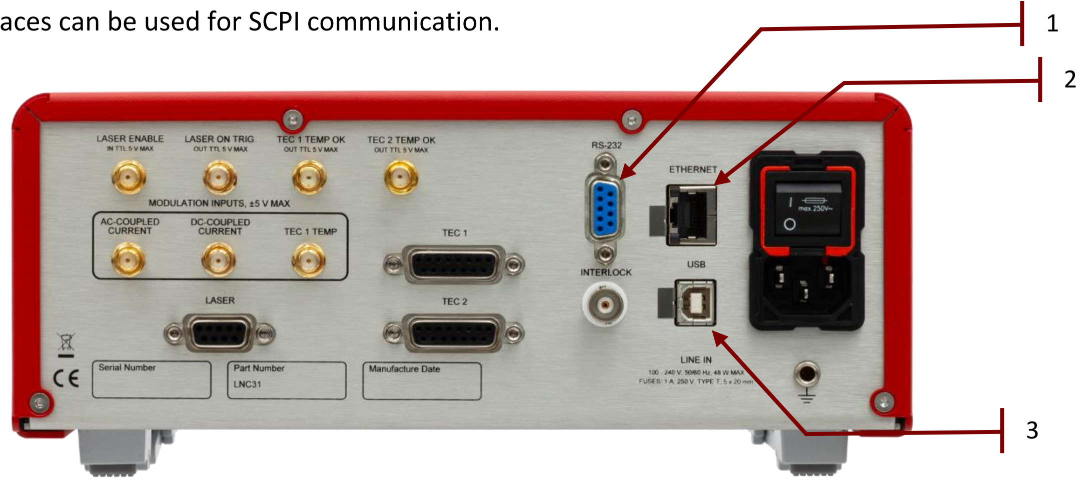


Figure 1 LNC31 Back Panel Communication Ports

1.1 USB Interface

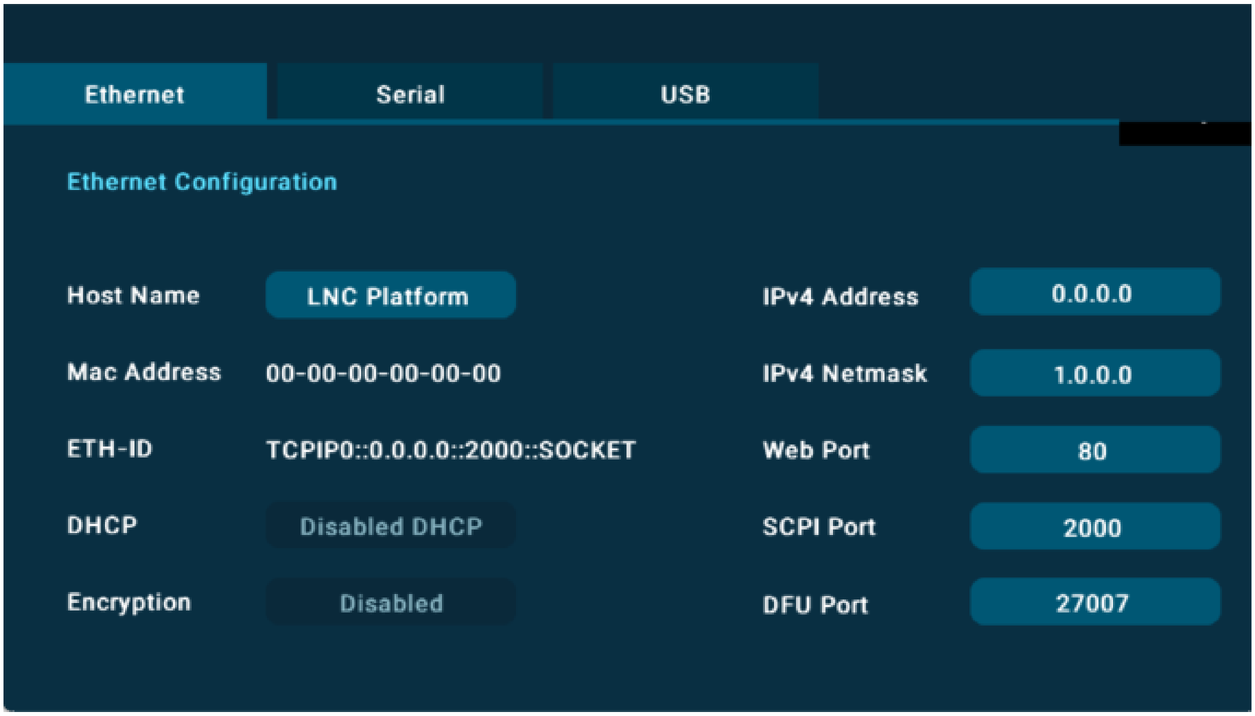
With USB connected, the device will enumerate as a Virtual COM port on the host device. For a Windows PC running Windows 10 or later, no special driver is required. For Windows PCs prior to Windows 10, the driver included as part of the firmware update utility is required.

1.2 RS-232

The communication settings for this interface are 115200 bps, 8 data bits, no parity, and 1 stop bit.

1.3 Ethernet

Ethernet interface settings can be configured from the LNC31 touchscreen display menu. Navigate to the menu by pressing the button in the top right corner (the icon with 3 horizontal lines) and then select Interface and Ethernet.



Ethernet Configuration

Host Name: LNC Platform

Mac Address: 00-00-00-00-00-00

ETH-ID: TCPIP0::0.0.0.0::2000::SOCKET

DHCP: Disabled DHCP

Encryption: Disabled

IPv4 Address: 0.0.0.0

IPv4 Netmask: 1.0.0.0

Web Port: 80

SCPI Port: 2000

DFU Port: 27007

By default, Dynamic Host Configuration Protocol (DHCP) is enabled, and the device uses a fixed Ethernet MAC address. If the device is connected to a local network without DHCP server, disable DHCP and set a static IPv4 address and netmask instead.

If multiple LNC31 systems are connected to the same network, the MAC address must be set to a unique address on each device.

1.4 Communication Settings

All communication commands must be terminated by a single carriage return character '\r' and a newline character '\n'. Communications originating from the device will be terminated by one or more combined carriage return and newline pairs '\r\n'. A connected host should not attempt to send additional queries until a response has been received from the device. Queries requesting the saving of configuration parameters will result in longer response times

The USB, RS-232 or Ethernet should not be used simultaneously to communicate with the device.

Chapter 2 Introduction to the SCPI Language

SCPI (Standard Commands for Programmable Instruments) is an ASCII based instrument command language designed for test and measurement instruments, based on a hierarchical tree structure. It means associated commands are grouped together under a common node. For example:

SOURce[1]

:CURRent

:LIMit

[:AMPLitude] {MIN|MAX|<amps>}

SOURce is the root level keyword of the command, CURRent is the second level keyword, LIMit is the third level keyword and AMPLitude is the fourth level keyword, which can be omitted in this case. A colon (:) separates between command levels.

For further information about SCPI please see SCPI Standard available at **www.ivifoundation.org**.

2.1 Syntax Conventions

The format used in this manual for commands is illustrated below:

```
SOURce[1]:CURRent {MINimum|MAXimum|<amps>}
```

The command syntax shows most commands (and some parameters) as a mixture of upper- and lower-case letters. The upper-case letters indicate the abbreviated spelling for the command. For shorter program lines, you can use the abbreviated form. For better program readability, you can send the long form. In the above example CURR and CURRent are both acceptable syntax spellings. Other spellings like CUR or CURRe are not allowed and will generate an error. Commands are not case sensitive.

Curly brackets ({}) enclose the parameter choices for a given command string. Braces are not sent with the command string.

A vertical bar (|) separates multiple parameter choices for a given command string. For example, {MINimum|MAXimum|<amps>} in the above command indicates that you can specify a numeric current parameter, or MINimum, or MAXimum. The bar is not sent with the command string.

Triangle brackets (<>) indicate that you must specify a value for the enclosed parameter. For example, the above syntax statement shows the <amps> parameter enclosed in triangle brackets. The brackets are not sent with the command string. You must specify a value for the parameter (for example 5.0) unless you select one of the other options shown in the syntax (for example MIN).

Some parameters and even parts of the command itself are enclosed in square brackets ([]). This indicates that the part of the command or the parameter is optional and can be omitted. The brackets are not sent with the command string. If you do not specify a value for an optional parameter, the instrument chooses a default value.

2.2 Command Separators

A colon (:) is used to separate a command keyword from a lower-level keyword. You must insert a blank space to separate a parameter from a command keyword. If a command requires more than one parameter, you must separate adjacent parameters using a comma (,) as shown below:

```
MEM:STAT:NAME 1,"My setting"
```

A semicolon (;) is used to separate commands within the same subsystem and can also minimize typing. For example, sending the following command string:

```
DISP:BRIG 1;CONT 0.5
```

is the same as sending the following two commands:

```
DISP:BRIG 1
```

```
DISP:CONT 0.5
```

Use a semicolon and a colon to link commands from different subsystems. The colon resets the command tree to the root level. For example:

```
SOURce2:TEMPerature 25.0;:OUTPut ON
```

2.3 Querying Parameter Settings

You can query the current value of most parameters by adding a question mark (?) to the command. For example, the following command sets the Laser TEC temperature setpoint:

```
SOURce2:TEMPerature 25.0
```

You can then query the value by typing:

```
SOURce2:TEMPerature?
```

2.4 Command Terminators

A command string sent to the instrument must terminate with a <new line> (<NL>) character. Command string termination will always reset the current SCPI command path to the root level.

2.5 Parameter Types

The SCPI language defines several data formats to be used in program messages and response messages.

2.5.1 Numeric Parameters

Commands that require numeric parameters will accept all commonly used decimal representations of numbers including optional signs, decimal points, and scientific notation. Special values for numeric parameters such as MIN, MAX, and DEF may also be accepted. You can also send engineering unit suffixes with numeric parameters (e.g., M, k, m, or u). If a command accepts only certain specific values, the instrument will automatically round the input numeric parameter to the accepted values.

Additionally numeric parameters can be programmed using binary, decimal, hexadecimal or octal data formats for the parameter value (for a detailed description of numeric formats see chapter 7.7.4 <NONDECIMAL NUMERIC PROGRAM DATA> of IEEE488.2-1992).

Example

The four commands program the *Auxiliary Enable Register* to the same value:

- `STAT:AUX:ENAB 2081`
  Program the *Auxiliary Enable Register* with 2018 decimal.

- `STAT:AUX:ENAB #H821`
  Program the *Auxiliary Enable Register* with 821 hexadecimal.
- `STAT:AUX:ENAB #Q4041`
  Program the *Auxiliary Enable Register* with 4041 octal.
- `STAT:AUX:ENAB #B100000100001`
  Program the *Auxiliary Enable Register* with 100000100001 binary.

2.5.2 Discrete Parameters

Discrete parameters are used to program settings that have a limited number of values (like CELSius, FAHRenheit or KELVin). They have a short form and a long form just like command keywords. You can mix upper- and lower-case letters. Query responses will always return the short form in all upper-case letters.

Example

The following command requires a discrete parameter for the temperature unit:

```
UNIT:TEMPerature {C|CEL|CELSius|F|FAR|FAHRenheit|K|KELVin}
```

Discrete parameters may also have a default condition. In this manual the default conditions are underlined.

2.5.3 Boolean Parameters

Boolean parameters represent a single binary condition that is either true or false. For a false condition, the instrument will accept OFF or 0. For a true condition, the instrument will accept ON or 1. When you query a boolean setting, the instrument will always return 0 or 1. Boolean parameters may also have a default condition. In this manual the default conditions are underlined.

2.5.4 ASCII String Parameters

String parameters can contain virtually any set of ASCII characters. A string must begin and end with matching quotes; either with a single quote or a double quote. You can include the quote delimiter as part of the string by typing it twice without any characters in between.

2.6 LNC31 SCPI commands

The supported commands for LNC31 are listed under the section '**Error! Reference source not found.**'.

2.6.1 Query/Get value commands

For the query commands the following response formats are being used:

String

Some of the commands return a response as a quoted string. Some examples of commands using this response format are commands for Ethernet/IP settings and the Identification Query command.

Integer Array

The commands for setting or querying date and time use an integer array as the input and output format with 3 comma separated integers.

Float

Most of the application specific query commands return the parameter value as a float value in an application specific response format. This response format is described in the section 'Application Specific Commands'.

Bool

Some commands, e.g. the command for set or get DHCP enable/disable use a bool (a 0 or 1) as input parameter (for set command) and in the response (in case of a query command).

Chapter 3 Commands by Subsystem

The SCPI commands are grouped into 2 subsystems: System and Application specific subsystem commands.

3.1 System Subsystem Commands

The set commands for date and time use an integer array as input parameters with three comma separated integers. The corresponding query commands for time and date outputs a response in the same format.

3.1.1 Command format and example for time/date

- Set time
  - Format: *SYSTem:TIME <HH, MM, SS>*
  - Example (Set time to 04:02:00 PM): *SYSTem:TIME 16,2,0*
- Set date
  - Format: *SYSTem:DATE <YYYY,MM,DD>*
  - Example (Set date to August 12th, 2025): *SYSTem:DATE 2025,8,12*

The commands for Ethernet/IPv4 settings use a string format in the input in case of a set command, and in the command response for query commands.

3.1.2 Command format and example for Ethernet/IPv4 commands

- Set the static IPv4 address to 10.0.100.1:
  - Format: SYSTem:COMMunicate:LAN:ADDRess "<IPv4 address>"
  - Example: SYSTem:COMMunicate:LAN:ADDRess "10.0.100.1"

Command *Description*	**Input**	**Output**
`*IDN?` *Identification query. Read the unit's identification string.*	N/A	String
`SYSTem:TIME` *Set system time.* *Format: SYSTem:TIME <HH, MM, SS>* *Example: Set the time to 04:02:00 PM* *SYSTem:TIME 16,02,02*	Integer	N/A
`SYSTem:TIME?` *Get system time.* *The response is an integer array in the format < HH, MM, SS >*	N/A	Integer
`SYSTem:DATE` *Set system date.* *Format: SYSTem:DATE <YYYY, MM, DD>* *Example: Set the date to March 4th, 2025* *SYSTem:DATE 2025, 03, 04*	Integer	N/A
`SYSTem:DATE?` *Get system date.* *The response is an integer array in the format < YYYY, MM, DD >*	N/A	Array
`SYSTem:COMMunicate:LAN:ADDRess` *Set system IPv4 address.* *This is the static IPv4 address used if DHCP is disabled.* *Example:* *SYSTem:COMMunicate:LAN:ADDRess "10.0.100.1"*	String	N/A
`SYSTem:COMMunicate:LAN:ADDRess?` *Get system IPv4 address.* *This is the static IPv4 address used if DHCP is disabled.* *The response is a string with the IP address.*	N/A	String
`SYSTem:COMMunicate:LAN:SMASk` *Set system IPv4 network mask.* *Example:* *SYSTem:COMMunicate:LAN:SMAsk "255.255.255.0"*	String	N/A
`SYSTem:COMMunicate:LAN:SMASk?` *Get system IPv4 network mask.* *The response is a string with the IPv4 network mask.*	N/A	String
`SYSTem:COMMunicate:LAN:GATEway` *Set IPv4 address for internet gateway.* *Example:* *SYSTem:COMMunicate:LAN:GATEway "10.0.100.1"*	String	N/A
`SYSTem:COMMunicate:LAN:GATEway?` *Get IPv4 address for internet gateway.*	N/A	String

The response is a string with the IPv4 address of the gateway.		
`SYSTem:COMMunicate:LAN:MAC?` *Get Ethernet MAC address.* *The response is a string with the Ethernet MAC address.*	N/A	String
`SYSTem:COMMunicate:LAN:DHCP` *Set DHCP enabled (1) or disabled (0).* *Example: Enable DHCP* *SYSTem:COMMunicate:LAN:DHCP 1*	Bool	N/A
`SYSTem:COMMunicate:LAN:DHCP?` *Get current setting of DHCP.* *The response is a Boolean (0 if disabled, and 1 if enabled).*	N/A	Bool

3.2 Application Specific Commands

The application specific commands are used to either query the current parameter values, or to set a parameter setpoint value. Each query command is followed by a command response from the device with the following format:

<Response_code:Response_Text>:<parameter value>,<parameter status>

For a successful command where the <Response_code:Response_Text> will typically be “000: OK” meaning the command executed successfully. The parameter status field will typically be “0x0000” for a successful command.

3.2.1 Query command and response example

This is an example of the command and resulting response for querying the laser TEC measured temperature.

Command***:*** *MEASure[:SCALar]:TEMPerature3?*

Response: *000: OK:25.306750775919273,0x0000*

- 25.306750775919273 is the measured value in degrees °C,
- 000: OK is the <Response_code:Response_Text> part of the response and,
- 0x0000 is the <parameter status> part of the response.

3.2.2 Set command example

The application specific set commands typically take a float as parameter. The only exception to this is the command to set laser state (OUTPut1) which takes either ‘on’ or ‘off’ as parameters.

Command example (Set the Laser TEC temperature setpoint to 25.0 degrees *°C)*:

SOURce2:TEMPerature 25.0

3.2.3 Set parameter value commands

Command *Description*	Default	Limits
`SOURce2:TEMPerature` *Set Laser TEC temperature setpoint in °C.* *The input shall be given as a float value.* *Example: Set Laser TEC temperature setpoint to 25.0 °C.* *SOURce2:TEMPerature 25.0*	25.0	10.0 – 70.0
`SOURce3:TEMPerature` *Set FBG TEC temperature setpoint in °C*	25.0	10.0 – 70.0
`SOURce1:CURRent` *Set Laser current setpoint in A.*	0.0	0.0 – 1.0
`OUTPut1:PROTection:VOLTage` *Set Laser voltage limit setpoint in V.*	4.0	0.0 – 7.5
`OUTPut2:PROTection:VOLTage` *Set Laser TEC voltage limit setpoint in V.*	5.0	0.0 – 5.0
`OUTPut3:PROTection:VOLTage`	5.0	0.0 – 5.0

Set FBG TEC voltage limit setpoint in V.		
`SOURce1:CURRent:LIMit` *Set Laser current limit setpoint in A.*	0.5	0.0 – 1.0
`SOURce2:CURRent:LIMit` *Set Laser TEC current limit setpoint in A.*	3.0	0.0 – 3.5
`SOURce3:CURRent:LIMit` *Set FBG TEC current limit setpoint in A.*	3.0	0.0 – 3.5
`OUTPut1` *Set Laser state (on or off).* *Example: Set Laser ON* *OUTput1 on*	N/A	N/A

3.2.4 Query parameter value commands

All the query commands listed below have the response format as shown in the example under section 'Query command and response example' and return the parameter value as a float.

Command	Description
`MEASure[:SCALar]:TEMPerature3?`	*Read the Laser TEC temperature in °C.*
`MEASure[:SCALar]:TEMPerature4?`	*Read the FBG TEC temperature in °C.*
`MEASure[:SCALar]:VOLTage1?`	*Read Laser voltage in V.*
`MEASure[:SCALar]:VOLTage3?`	*Read Laser TEC voltage in V.*
`MEASure[:SCALar]:VOLTage4?`	*Read FBG TEC voltage in V.*
`MEASure[:SCALar]:CURRent1?`	*Read Laser current in A.*
`MEASure[:SCALar]:CURRent3?`	*Read Laser TEC current in A.*
`MEASure[:SCALar]:CURRent4?`	*Read FBG TEC current in A.*
`SOURce1:CURRent?`	*Read Laser current setpoint in A.*
`SOURce2:TEMPerature?`	*Read Laser TEC temperature setpoint in °C.*
`SOURce3:TEMPerature?`	*Read FBG TEC temperature setpoint in °C.*
`OUTPut1:PROTection:VOLTage?`	*Read Laser voltage limit setpoint in V.*
`OUTPut2:PROTection:VOLTage?`	*Read Laser TEC voltage limit setpoint in V.*
`OUTPut3:PROTection:VOLTage?`	*Read FBG TEC voltage limit setpoint in V.*
`SOURce1:CURRent:LIMit?`	*Read Laser current limit setpoint in A.*
`SOURce2:CURRent:LIMit?`	*Read Laser TEC current limit setpoint in A.*
`SOURce3:CURRent:LIMit?`	*Read FBG TEC current limit setpoint in A.*

Chapter 4 Thorlabs Worldwide Contacts

For technical support or sales inquiries, please visit us at www.thorlabs.com/contact for our most up-to-date contact information.

Corporate Headquarters
Thorlabs, Inc.
43 Sparta Ave
Newton, New Jersey 07860
United States

sales@thorlabs.com
techsupport@thorlabs.com

Product Manufacturer
Thorlabs, Inc.
43 Sparta Ave
Newton, New Jersey 07860
United States

sales@thorlabs.com
techsupport@thorlabs.com

EU Importer
Thorlabs GmbH
Münchner Weg 1
D-85232 Bergkirchen
Germany

sales.de@thorlabs.com
europe@thorlabs.com

UK Importer
Thorlabs Ltd.
204 Lancaster Way Business Park
Ely CB6 3NX
United Kingdom

sales.uk@thorlabs.com
techsupport.uk@thorlabs.com

THORLABS

www.thorlabs.com