Protocol manual

Lightning Protocol

Product no:	10440904
Versions:	2.0
Release date:	15 Aug 2019
	Vision Hardware Partner De Steiger 59 1351AD Almere The Netherlands www.VisionHardwarePartner.nl info@VisionHardwarePartner.nl

Change history / Erratum:

15th august 2019:
first release date

1 Getting started

1.1 *Connecting to the device*

Two ways can be used to connect to the device.

1. **The virtual comm port**
   With the virtual comm port the developer can use existing software for controlling serial ports. Communication settings like baud rate, data bits and parity do not need to be set.
2. **the USB direct drivers**
   FTDIs USB direct drivers can be used to directly find and connect to the device.

For both driver types the following options com port settings need to be specified:

baudrate:	115200
databits:	8
parity:	none
stopbits:	1

1.2 *protocol basics*

1.2.1 Message separation

Messages in the serial data stream are be separated by either '\n' (line feed). Or '\r' (CR).
Each message (command) to the device has to be followed by \n or \r. Each message from the device to the computer will be followed by both '\r' and '\n' .

Commands are not case sensitive. Reports from the device are always in capitals.

1.2.2 Arguments and data representation

<xxxx> = context defined value in hexadecimal.
<dddd> = context defined value in decimal.

The number of positions for xxxx or dddd must always be used. Use leading zeros for padding if necessary.

For digital I/O blocks the value represents a bit mask. Each bit represents an input / output.

2 Commands and events

This chapter describes the possible commands for the controller.

The user can send commands to the controller to set items, or to request status data.
The controller can respond to these commands. Also it can notify the user of hardware events happening.

The user can send the following types of messages to the device:

- **get** : get current value of an item
- **Assign** : assign a specified value to an item
- **Notify** : set how the controller will notify the user of hardware events
- **System** : Set / get system properties

The device can send the following types of messages:

- **confirmation**: confirms a received command
- **data** : Provides requested data
- **notification**: notifies the user of a hardware event

The following paragraphs describe the possible commands, ordered by hardware functionality.

2.1 *Lamp profile settings*

25 lamp setting profiles can be set and retrieved.

The lamp needs to have a resistor between 0V and the Ident input. The value of this resistor sets the profile number.
The user can also change the settings for each profile.

!!!Caution !!!:
both the lamp and the controller are protected only by the limits as defined in the lamp profiles.
It is extremely important that the lamp profile settings are thoroughly checked for safety before applying them.
It is strongly recommended to only use the profiles provided by Vision Hardware Partner.

Command	function	response
LAMPSELSET<dd>\n	Select the type of lamp that the controller will be expecting. <dd> : profile number, starting at 1.	LAMPSEL=<dd>\n
LAMPSELGET\n	Request the current setting for the lamp profile selection	LAMPSEL=<dd>\n
LAMPDETGET\n	Request the lamp type detected by the controller <dd> : detected lamp type (profile), where 00 means “no lamp connected”	LAMPDET=<dd>\n
LAMPAUTOLOADPRFSET<d>\n	<d>: can be 1 or 0 When 1 the controller will automatically select the profile that belongs to the detected lamp type. When 0 only lamps that match the selected profile will be accepted. If a lamp of a different type is connected the controller will disable the driver and signal an error.	LAMPAUTOLOAD PRFSET=<d>\n
LAMPAUTOLOADPRFGET\n	Request the above mentioned state. <d>: either 1 or 0	LAMPAUTOLOAD PRFSET=<d>\n

Command	function	response
LAMPPRF<dd>SET<ddddd>,<dddd>,<dd>, <dd>\n	Define a lamp profile (**See note above**) <dd> selected lamp profile (01 to 25) <ddddd> selected lamp current in ma <dddd> max on time in us <dd> cooldown factor. <dd> lamp buffer voltage	LAMPPRF<dd>=<ddddd>,<dddd>,<dd>, <dd>\n
LAMPRF<dd>GET\n	Request the contents of profile <dd>	LAMPPRF<dd>=<ddddd>,<dddd>,<dd>, <dd>\n

Lamp profiles vs resistor value

Profile no	Resistor value	
0	> 100K	In this range the lamp is assumed to be disconnected. This profile is therefore invalid.
1	100K	
2	56K	
3	39K	
4	27K	
5	22K	
6	18K	
7	15K	
8	12K	
9	10K	
10	8,2K	
11	6,8K	
12	6,2K	
13	5,6K	
14	4,7K	
15	3,9K	
16	3,3K	
17	2,7K	
18	2,2K	
19	1,8K	
20	1,5K	
21	1K	
22	680R	
23	390R	
24	180R	
25	< 180R	

2.2 Temperature control

Both the lamp and the driver are assumed to have a temperature sensor. The temperature is measured each second. For each temperature there is a temperature limit. If this limit is exceeded the driver will be disabled and and error state will be reported. Two values can be set. One to activate the temperature protection and one to reset it.

Command	function	response
CTRLDRVTMPGET\n	Requests the temperature from the sensor near the lamp driver <+ddd> temperature: 3 digits + +/- sign	CTRLDRVTMP=<+ddd>\n
CTRLLMPTMPGET\n	Requests the temperature from the sensor on the lamp <+ddd> temperature: 3 digits + +/- sign	
CTRLSERIALGET\n	The device allows the user to store a text string that can be used as a product serial number. The serial does not have a fixed length, only a maximum of 15 characters.	CTRLSERIAL=<serial>\n
CTRLSERIALSET<serial>\n		
CTRLPROTTMPDRV LIMSSET<ddd>,<ddd> \n	Temperature protection settings for the driver <ddd>: The switch off point (maximum temperature) <ddd>: The reset point (driver is reenabled Both in 3 digits without sign	CTRLPROTTMPDRV LIMS=<ddd>,<ddd> \n
CTRLPROTTMPDRV LIMSGET\n		
CTRLPROTTMPLMP LIMSSET<ddd>,<ddd>\n	Temperature protection settings for the lamp <ddd>: The switch off point (maximum temperature) <ddd>: The reset point (driver is reenabled Both in 3 digits without sign	CTRLPROTTMPLMP LIMS=<ddd>,<ddd> \n
CTRLPROTTMPLMP LIMSGET\n		
CTRLERRGET\n	Gets the current error state of the controller. <xx> a byte of which each bit represents an error state. See also: “error states”	CTRLERR=<xx>\n

2.3 General properties

Command	function	response
CTRLSERIALGET\n	The device allows the user to store a text string that can be used as a product serial number. The serial does not have a fixed length, only a maximum of 15 characters.	CTRLSERIAL=<serial>\n
CTRLSERIALSET<serial>\n		CTRLSERIAL=<serial>\n
CTRLERRGET\n	Gets the current error state of the controller. <xx> a byte of which each bit represents an error state. See also: “error states”	CTRLERR=<xx>\n

Error states

Bit no.	error	description
0	Wrong lamp	The lamp connected does not match the selected profile
1	Lamp profile	There is an error in the settings of the selected lamp profile. Either it has not been defined, or there are illegal values in the profile
2	No lamp	No lamp was detected. This means that either no lamp is connected, or there is a problem with the detection mechanism
3	Temperature Measurement	The temperature sensor gave an implausible value. Usually that would mean that there is a faulty connection to the sensor
4	DA converter	The controller cannot access the DA converter. This would usually indicate a defect on the board.
5	reserved	
6	reserved	
7	Overheat	The temperature of either the lamp or its driver exceeded the set limit and the driver shut down. Usually the driver would automatically cool down and restart when the temperature gets below the reset point.

3 System commands

Command	function	response
RS\n	Requests the sampling speed. The return value tells how many times per second the main loop is executed	RS=<ddddd>\n
RB\n	Reboot. Reboots the device. The protocol defines no response for this command, but after the reboot the bootloader will signal its presence by sending “^”	^
TYPE\n	Requests the device type. Possible codes for <text> are: 10440904: Lightning OEM10-3	TYPE=<text>\n
PROTVER\n	Request the current version. PROTVER is the version of the command protocol SWVER is the version of the current firmware <d.d.d> = Major, Minor, Bugfix **Major version**: Major functional changes. Version may not be compatible to previous major versions. **Minor version**: additional functionality. Version will be downwards compatible. **Bugfix version**: no protocol/functional changes. Just bugfixes	PROTVER= <d.d.d>\n
SWVER\n		SWVER= <d.d.d>\n

4 Error messages

4.1 Error responses

If a command was not understood, the reply will always start with '?'.

When the command is not understood at all the controller will report: *?CMD*

If a command is partially understood. The controller will first send '?' and then echo the request up to the point where it was not understood, followed by a '?'
Checking if a command was handled ok can be done by scanning replies for an echo of the command which was sent both with and without the '?'.

For example:

The command DIA000000 sets all inputs of a 24 input module to 0.

reply for accepted:	Possible error response:
DIA=000000\n	*?DIA?\r\n*

If the user would specify a command for digital inputs (DI) which does not exist. For example DIX, the controller would report: *?DI?\n*