ISSI · INNOVATIVE SCIENTIFIC SOLUTIONS

ISSI LC-PZAE

Canon® EF Zoom Lens Controller (LC-PZA2), Ethernet

APPLICATION PROGRAMMING INTERFACE

Document	**Programmer's Reference Manual**
Version	**4.9**
Firmware	**4.9.0 and later (notes for 4.7.x)**
Date	**November 2026**

Notice

Copyright © 2026 Innovative Scientific Solutions Inc (ISSI). All rights reserved.

ISSI does not warrant that the ISSI LC-PZAE API will function properly in every hardware/software environment. This software is inherently complex, and users are cautioned to verify the results of their work. ISSI has tested the software and reviewed the documentation. ISSI MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESSED OR IMPLIED, WITH RESPECT TO THIS SOFTWARE OR DOCUMENTATION, THEIR QUALITY, PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS SOFTWARE AND DOCUMENTATION ARE LICENSED "AS IS" AND YOU, THE LICENSEE, ARE ASSUMING THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE. IN NO EVENT WILL ISSI BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE OR DOCUMENTATION, even if advised of the possibility of such damages. In particular, ISSI shall have no liability for any programs or data stored or used with ISSI software, including the costs of recovering such programs or data.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the above disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the above disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this API without specific prior written permission.

Canon is a registered trademark of Canon Inc. ISSI is not affiliated with Canon Inc.

Innovative Scientific Solutions Inc.
7610 McEwen Road, Dayton, OH 45459
Ph.: (937) 630-3012 · Fax: (937) 630-3015
www.innssi.com · issi-sales@innssi.com

Contents

1. Overview
2. Connection & message format
3. Quick start
4. Command quick reference
5. Lens status
6. Focus
7. Aperture
8. Zoom (motorized)
9. Image stabilization
10. Controller settings & info
11. Network settings
12. Unsolicited messages
13. Error replies
14. Programming examples
15. Revision history

1 Overview

The ISSI LC-PZAE drives a Canon EF-mount zoom lens (and compatible third-party EF lenses) over an Ethernet network. It is the ISSI LC-2 lens controller with an added **zoom motor**: a stepper motor coupled to the lens's zoom ring, so that zoom can be set remotely as well as focus and aperture. The host sends short ASCII text commands in UDP datagrams and the controller answers each one with one or more UDP datagrams. Several controllers can share one network, each with its own IP address.

> **Same API as the LC-2.** Every LC-2 command works on the LC-PZAE with the same replies; the zoom commands in §8 are the addition. Software written for the LC-2 (including the ISSI LC-2 GUI) recognizes the LC-PZAE from the `ver` reply.

Units used in this guide

QUANTITY	UNIT	NOTES
Zoom	Focal length, mm	Set with `setZoom`, read with `getZoom`. The lens reports its focal length in whole millimetres; the range is lens-specific, read it with `zRange` or `ping`.
Zoom motor	Motor steps	`moveZoom` only. One unit is 50 motor steps, about 0.2 mm of focal length on the Canon EF-S 18-135 (depends on the lens and the coupling).
Focus	Motor encoder counts	Integer. The range is lens-specific; read it with `fRange` or `ping`.
Aperture	f-number	One-third-stop scale, see §7. The range depends on the lens and the current zoom.
Distance	Meters	The lens reports a near/far bracket, not an exact distance.
Time	ms, µs or s, as stated	

2 Connection & message format

Network settings

PARAMETER	VALUE
Interface	10/100 Ethernet, RJ-45
Protocol	**UDP**, port **1339**
IP address	Static. Factory default `192.168.2.252`, mask `255.255.255.0` (see §11 to change it)
Replies	Sent to the IP address and port the command came from
Character set	ASCII

Commands

- Send **one command per datagram**. No line ending is needed; a trailing CR, LF or CR+LF (and leading or trailing spaces) is ignored, so `ver` and `ver\r` are the same command.
- Commands are **case-sensitive** and must match exactly: `getZoom`, not `getzoom` or `getZoom1`. A datagram that is not a known command gets **no reply**.
- Parameters follow an equals sign with no spaces: `setZoom=50`. Numbers are plain decimal (`-200`, `8.0`); anything else returns `errorParam` and the command does nothing (firmware 4.9.0 and later).
- The maximum command length is 255 characters; a longer datagram is answered with `errorParam`.

Replies

- Each reply is a separate datagram containing one text message, without a line ending.
- Most replies have the form `Name=value` (`Zoom=50`) or a status word (`OK`, `MIN_OK`, `zoomDone`).
- Some commands answer with **several datagrams**: `ping` sends seven, `setZoom` sends `Zoom=mm` and then `zoomDone`, `setFocus` sends `Focus=N` and then `focusDone`. Read until the last expected reply, or until no more datagrams arrive for about 0.5 s.
- Errors are single words that start with `error`, or `nolensfound` / `zoomOutOfRange` (see §13).

Timing

Send **one command at a time** and wait for its reply before sending the next one. Lens commands run the zoom, focus or aperture motor and answer once the motor has stopped; datagrams that arrive in the meantime are queued and run afterwards, in order. Suggested host timeouts:

COMMANDS	TYPICAL	SUGGESTED TIMEOUT
Queries (`ver` , `getFocus` , `getZoom` , `isIS` , …)	< 20 ms	1 s
`setAper` , `moveAper` , `openAper`	10–50 ms	2 s
`ping`	0.1 s; 3–5 s when it measures the focus range	**20 s**
`setFocus` , `moveFocus` , `goMIN` , `goMAX`	0.2–1 s, depends on the lens	**20 s**
`fRange` , `setFocus2`	3–5 s	**20 s**
`setZoom`	0.5–10 s, depends on the travel and the motor settings	**65 s**
`moveZoom`	Depends on N and `setSD` ; about 0.1 s per 10 units at the defaults	**65 s**

> UDP does not guarantee delivery. If a reply does not arrive within the timeout, send the command again; queries and absolute moves (`setZoom` , `setFocus` , `setAper`) can be repeated safely. A zoom move is limited to 60 s and a focus motor that does not stop within 15 s is reported as `errorFocus` , so the controller never stays silent indefinitely (firmware 4.9.0).

Power-up

After power-up or `Reboot` the controller needs about 4 seconds to start (up to 8 s without a lens). It initializes the lens by itself; there is no separate initialization command, and the zoom motor does not move at start-up. Send `ver` until it answers, then `ping` .

3 Quick start

A typical session: identify the controller, read the lens status, zoom, then focus. Each line below is one datagram.

```
→ ver
← ISSI LC-PZAE=4.9.0 (s/n:17201)
→ ping
← zRange=18,135
← fRange=0,1210
← aRange=3.5,22.0
← Current=18,0,3.5
← AF=1
← IS=1
← ISactive=0
→ setZoom=50
← Zoom=50
← zoomDone
→ ping
   … aperture range at 50 mm, focus range …
→ setFocus=800
← Focus=800
← focusDone
→ setAper=8
← Iris=8.0
```

Focus switch: the lens's AF/MF switch must be in the **AF** position for the controller to drive the focus motor. `ping` reports the switch position as `AF=1` or `AF=0`. **Zoom motor:** the motor must be engaged with the lens's zoom ring as described in the Operating Manual; the controller cannot tell whether the ring is coupled.

4 Command quick reference

COMMAND	REPLY	DESCRIPTION
LENS STATUS		
`ping`	7 datagrams: `zRange=…` … `ISactive=…`	All ranges and current positions
`Current`	`Current=Z,F,A`	Current zoom, focus and aperture
`getLens`	`Lens=<name>`	Lens model name
`lid`	`<number>`	Lens ID code
`Sleep` / `Wakeup`	`OK` / `OK`, `NOK`	Lens power saving
ZOOM (MOTORIZED)		
`setZoom=mm`	`Zoom=mm`, `zoomDone`	Drive the zoom to a focal length
`moveZoom=±N`	`Zoom=mm`	Turn the zoom motor by N units (+ wide, − narrow)
`getZoom`	`Zoom=mm`	Current focal length
`zRange`	`zRange=min,max`	Zoom range
`setSD=µs` / `getSD`	`OK` / `SD=µs`	Zoom motor step period (speed)
`setIC=N` / `getIC`	`OK` / `IC=N`	Steps between focal-length checks
FOCUS		
`setFocus=N`	`Focus=N`, `focusDone`	Move focus to an absolute position
`setFocus2=N`	`Focus=N`, `focusDone`	Absolute move after re-zeroing at the end stop
`moveFocus=±N`	`Focus=N`	Move focus by a relative amount
`getFocus`	`Focus=N`	Current focus position
`fRange` / `refRange`	`fRange=min,max`	Measure the focus range

`goMIN` / `goMAX`	`MIN_OK` / `MAX_OK`	Drive focus to near / infinity end
`setZero`	`OK`	Reset the focus encoder counter
`getDist`	`Dist=near,far`	Focus distance bracket in meters
`setFocusT=ms` / `getFocusT`	`OK` , `T=ms` / `T=ms`	End-stop dwell time for slow lenses
`setVerifyF=0|1` / `setVerifyF`	`OK` , `VF=x` / `VF=x`	Verified focus completion
`setMoveT=ms` / `setMoveT`	`OK` , `MT=ms` / `MT=ms`	Minimum focus-move wait
`readFocus`	`FocusR=N`	Read the focus encoder live
APERTURE		
`setAper=F`	`Iris=F`	Set an f-number
`moveAper=±N`	`Iris=F`	Step N one-third stops
`openAper`	`OK`	Fully open the aperture
`aRange`	`aRange=min,max`	Aperture range at the current zoom
IMAGE STABILIZATION		
`isIS`	`IS=0|1`	Is IS available and switched on?
`enableIS=s`	`ISactive=0|1`	Run IS for s seconds (0 = stop)
CONTROLLER SETTINGS & INFO		
`ver`	`ISSI LC-PZAE=x.y.z (s/n:N)`	Firmware version and serial number
`apiver`	`API=3.0`	API version
`setName=xxxxxxx` / `getName`	– / `NAME=…`	7-character controller nickname
`getSN` / `getHW` / `getVENDOR`	`SN=` / `HW=` / `VENDOR=`	Serial number, hardware revision, dealer ID
`setProto=0|1|2` / `getProto`	`OK` / `Proto=n`	Lens-bus mode

`Reboot`	`OK`	Restart the controller
NETWORK SETTINGS		
`getIP` / `getMASK` / `getGW` / `getMAC`	`MAC=a.b.c.d` / `MASK=` / `GW=` / `MAC=`	Read the network settings
`ChangeIP=` / `ChangeMASK=` / `ChangeGW=`	`IPchanged` / `MASKchanged` / `GWchanged`	Change them (applied after `Reboot`)

5 Lens status

The controller initializes the lens at power-up. A lens attached later is detected by the next `ping`, zoom or focus command.

`ping` Lens status

LENS

ACTION — Returns all ranges and current positions. If the focus range is not yet known (after power-up or a lens change), `ping` measures it first: the focus ring moves end to end and the reply takes a few seconds. Aperture limits and, on many lenses, the focus range change with zoom, so send `ping` again after `setZoom`.

REPLY — Seven datagrams, in this order:

- `zRange=minZ,maxZ` : zoom range, mm
- `fRange=minF,maxF` : focus range
- `aRange=minA,maxA` : aperture range (f-number) at the current zoom
- `Current=Z,F,A` : current zoom, focus and aperture
- `AF=x` : 1 if the lens switch is set to AF, 0 if MF
- `IS=y` : 1 if the lens has IS and its IS switch is on, otherwise 0
- `ISactive=s` : 1 while stabilization is running (see `enableIS`)

ERRORS — `nolensfound` (one datagram): no lens attached, the lens was removed or changed, or it does not respond. The controller then looks for a lens again, so send `ping` again after about 1 s. A newly attached lens is initialized and its focus range measured by that next `ping`.

```
→ ping
← zRange=18,135
← fRange=0,1210
← aRange=3.5,22.0
← Current=50,800,8.0
← AF=1
← IS=1
← ISactive=0
```

`Current` Current positions

LENS

REPLY — `Current=Z,F,A` : zoom (mm), focus and aperture, the same as the fourth `ping` datagram, without measuring anything. `nolensfound` without a lens.

`getLens` Lens name

LENS

REPLY `Lens=<name>`, the model name stored in the lens (up to 63 characters from firmware 4.9.0). Some lenses, mostly older ones, do not report a name; the reply is then `Lens=`.

```
→ getLens
← Lens=EF-S18-135mm f/3.5-5.6 IS STM
```

`lid` Lens ID

LENS

REPLY The lens ID code reported by the lens, as a decimal number without a prefix (for example `112`). `255` means no lens answered.

`Sleep` · `Wakeup` Lens power saving

LENS

ACTION `Sleep` switches off the lens motor power, which keeps older lenses cool when they are idle for a long time. `Wakeup` re-initializes the lens. The zoom motor is powered only while it moves and is not affected.

REPLY `Sleep`: `OK`. `Wakeup`: `OK` if the lens answered, otherwise `NOK`.

6 Focus

Focus positions are motor encoder counts within the range reported by `fRange` or `ping`. The controller measures the range on the first `ping` after power-up or a lens change; if the first command is a focus move, that command measures it first, so it takes longer and the focus ring travels end to end. Focus moves answer when the motor has stopped. Most zoom lenses are not parfocal: after `setZoom`, set the focus again.

`setFocus=N` Absolute focus position

FOCUS

PARAMETER — `N`: target position. From firmware 4.9.0 a target outside the focus range is limited to the nearest end, and the reply shows the limited value.

REPLY — Two datagrams: `Focus=N`, then `focusDone`. Non-USM lenses may take longer.

ERRORS — `errorFocus`: the lens did not complete the move, the motor did not stop within 15 s, or no lens is attached · `errorParam`: not a number · `errorRange`: outside −32768…32767.

```
→ setFocus=800
← Focus=800
← focusDone
```

`setFocus2=N` Absolute move from the end stop

FOCUS

ACTION — Drives the focus to the end stop, re-zeroes the encoder counter there, waits `setFocusT`, then moves `N` counts. Slower than `setFocus`, but it removes the drift that accumulates on some lenses over many moves.

REPLY — Two datagrams: `Focus=N`, then `focusDone`.

ERRORS — `errorFocus` · `errorParam` · `errorRange`

`moveFocus=±N` Relative focus move

FOCUS

PARAMETER — `N`: counts to move, positive or negative. Moves that would pass a range limit stop at the limit.

REPLY — `Focus=N`, the new position (one datagram, no `focusDone`).

ERRORS — `errorFocus` · `errorParam`

```
→ moveFocus=100
← Focus=900
```

`getFocus` Current focus position

FOCUS

REPLY — `Focus=N` : the position of the last move. It is not read from the lens, so a focus ring turned by hand is not reflected; use `readFocus` for a live reading.

`fRange` · `refRange` Measure the focus range

FOCUS

ACTION — Drives focus to both ends and measures the range. The focus ring moves through its full travel and waits `setFocusT` at each end. The two commands are identical.

REPLY — `fRange=min,max` . `fRange=0,0` if no lens is attached.

```
→ fRange
← fRange=0,1210
```

`goMIN` · `goMAX` Drive focus to an end stop

FOCUS

ACTION — `goMIN` moves focus to the closest-focus end, `goMAX` to the infinity end. From firmware 4.9.0, `getFocus` then reports the real end position in the measured range.

REPLY — `MIN_OK` / `MAX_OK`

`setZero` Reset the focus counter

FOCUS

ACTION — Sets the focus encoder counter to zero at the current position; later positions are counted from here.

REPLY — `OK` . `nolensfound` without a lens.

`getDist` Focus distance

FOCUS

REPLY — `Dist=near,far` in meters: the lens's distance encoder reports that the current focus distance lies between *near* and *far*. `-0.01` stands for infinity (`Dist=-0.01,-0.01` at the infinity end). Lenses without a distance encoder may report `0.00` .

```
→ getDist
← Dist=1.69,1.41
```

`setFocusT=ms` · `getFocusT` End-stop dwell for slow lenses

FOCUS

PARAMETER	`ms` : 0–10000, default 900. Stored in non-volatile memory.
ACTION	How long `fRange` and `setFocus2` wait at each end of travel. Increase it for old or slow lenses (for example macro lenses) if `fRange` reports a range that is too small.
REPLY	`setFocusT=ms` : two datagrams, `OK` and `T=ms` . `getFocusT` or `setFocusT` without a value: `T=ms` .
ERRORS	`errorRange` : outside 0–10000 · `errorParam` : not a number.

Verified focus completion (advanced)

By default the controller answers a focus move as soon as the motor reports that it has stopped, and `Focus=` repeats the target. With verification switched on, it also waits until the lens's own focus counter stops changing and reports the position the lens actually reached.

`setVerifyF=X` · `setVerifyF` Verified focus completion

FOCUS

PARAMETER	`0` = off (default), `1` = on. Stored in non-volatile memory.
ACTION	When on, `setFocus` , `setFocus2` and `moveFocus` reply with the position read back from the lens (it can differ from the target by a few counts, more if the lens stopped short) and answer `errorFocus` if the counter does not settle within 3 s.
REPLY	`setVerifyF=X` : `OK` and `VF=X` . Query: `VF=X` .

```
→ setVerifyF=1
← OK
← VF=1
→ setFocus=1000
← Focus=957
← focusDone
```

`setMoveT=ms` · `setMoveT` Minimum focus-move wait

FOCUS

PARAMETER	`ms` : 0–2000, default 200. Stored in non-volatile memory.
ACTION	The shortest time a focus move waits before it can answer; the wait still extends while the motor is running. Lower values make short moves on fast lenses answer sooner. Keep the default unless the application needs the extra speed, and test with your lens.
REPLY	`setMoveT=ms` : `OK` and `MT=ms` . Query: `MT=ms` .
ERRORS	`errorRange` · `errorParam`

`readFocus` Live focus reading

FOCUS

REPLY

`FocusR=N` : the focus counter read from the lens right now, in the same units as `setFocus` . It also updates the position that the next relative move starts from.

7 Aperture

The aperture uses a one-third-stop f-number scale. The usable range depends on the lens and, for most zoom lenses, on the current zoom position (an EF-S 18-135 f/3.5-5.6 opens to f/3.5 at 18 mm and to f/5.6 at 135 mm); `ping` and `aRange` report it. Use these values:

`1.0` `1.1` `1.3` `1.4` `1.6` `1.8` `2.0` `2.2` `2.5` `2.8` `3.2` `3.5` `4.0` `4.5` `5.0` `5.6` `6.3` `7.1`
`8.0` `9.0` `10` `11` `13` `14` `16` `18` `20` `22` `25` `29` `32` `36` `40` `45` `51` `57` `64` `72` `80` `90`

`setAper=F` Set f-number

APERTURE

PARAMETER `F` : f-number from the table above, within the lens's range at the current zoom. Other values are rounded to the nearest table value.

REPLY `Iris=F` , the aperture now set.

ERRORS Out of range: `errorAperLimits` followed by `errorAper` (two datagrams) · `errorParam` : not a number.

```
→ setAper=8
← Iris=8.0
```

`moveAper=±N` Step the aperture

APERTURE

PARAMETER `N` : one-third-stop steps. Positive closes the aperture (larger f-number), negative opens it.

REPLY `Iris=F`

ERRORS `errorAperLimits` : the step would leave the lens's range · `errorAper` : the lens did not complete the move · `errorParam` .

```
→ moveAper=1
← Iris=9.0
```

`openAper` Fully open

APERTURE

ACTION Opens the aperture to the widest value available at the current zoom.

REPLY `OK`

`aRange` Aperture range

APERTURE

REPLY `aRange=min,max` at the current zoom, for example `aRange=3.5,22.0` .

8 Zoom (motorized)

The LC-PZAE turns the lens's zoom ring with a stepper motor. EF lenses have no zoom encoder that the controller could count steps against; instead the lens reports its **current focal length in whole millimetres**, and the controller uses that reading as feedback: `setZoom` runs the motor until the lens reports the requested focal length, `moveZoom` runs it a given number of steps.

What to expect

- **Resolution.** The lens reports whole millimetres, and on some lenses the reported value changes in steps of 2 mm or more near the long end (an EF-S 18-135 goes from 134 to 136 mm). `setZoom` therefore stops within **±1 mm** of the target for targets inside the range; the reply shows the focal length actually reached.
- **End stops.** A target equal to the minimum or maximum of `zRange` is driven all the way into the mechanical stop, so `setZoom=18` and `setZoom=135` always reach the ends.
- **Direction.** The travel direction is chosen once, at the start of the move, and the controller never reverses within a move; it stops as soon as the target is reached or crossed. This avoids hunting on lenses whose focal-length readout skips values, at the cost of a possible overshoot of one feedback cycle.
- **Stall.** If the reported focal length stops changing although the motor is stepping (mechanical stop, slipping coupling, motor not engaged), the move ends and the reply shows where the lens is. Compare the reply with the target.
- **After zooming** the aperture range and, on most lenses, the focus change. Send `ping` or `aRange`, and set the focus again.
- The motor is powered only while it moves. No zoom command is run at power-up.

`setZoom=mm` Drive the zoom to a focal length

ZOOM

PARAMETER	`mm` : target focal length within `zRange` . A decimal value is rounded to the nearest millimetre.
ACTION	Runs the zoom motor until the lens reports the target focal length (see *What to expect*). The move is limited to 60 s. A prime lens, or a lens whose zoom range is not known, answers at once without moving.
REPLY	Two datagrams: `Zoom=mm` , the focal length now reported by the lens, then `zoomDone` . `zoomDone` **is always the last datagram**, also after an error, so a host can wait for it in every case.
ERRORS	`zoomOutOfRange` : target outside `zRange` , nothing moved · `nolensfound` : no lens · `errorParam` : not a number. Each is followed by `zoomDone` .

```
→ setZoom=50
← Zoom=50
← zoomDone
→ setZoom=200
← zoomOutOfRange
← zoomDone
```

`moveZoom=±N` Turn the zoom motor

ZOOM

PARAMETER	`N` : −10000…10000 units of 50 motor steps. **Positive = wide** (towards the shortest focal length), **negative = narrow** (towards the longest). On an EF-S 18-135 one unit is about 0.2 mm of focal length; the ratio depends on the lens and the coupling.
ACTION	Steps the motor the requested amount in bursts of `setIC` steps, reading the focal length between bursts, and stops early at the end of the zoom range. Use it for fine adjustment or jog buttons; use `setZoom` for absolute positions.
REPLY	`Zoom=mm` , the focal length after the move (one datagram). At a zoom limit the motor does not move and the current value is returned.
ERRORS	`nolensfound` · `errorParam` · `errorRange` : |N| over 10000.

```
→ moveZoom=-10
← Zoom=52
```

`getZoom` Current focal length

ZOOM

REPLY	`Zoom=mm` , read from the lens. A reading outside the zoom range (a rare bus error) is re-read, and the last good value is reported if it persists, so the value does not jump. Prime lenses report their fixed focal length.

zRange Zoom range — ZOOM

REPLY — `zRange=min,max` in mm, for example `zRange=18,135`. The range is read from the lens at detection and accepted only when two consecutive reads agree.

Zoom motor tuning (advanced)

Two stored settings determine how fast the motor runs and how often the controller checks the lens during a move. The defaults suit the Canon EF-S 18-135; change them only if a lens overshoots, stalls or zooms too slowly, and test the full range afterwards. Both are kept in non-volatile memory and through firmware updates.

setSD=µs · **getSD** · **setIC=N** · **getIC** Motor step period and feedback interval — SETTING

`SETSD=MS` — Step half-period in microseconds, 20–8000, default 100 (one motor step every 200 µs). Smaller = faster. Too small a value and the motor loses steps or stalls.

`SETIC=N` — Motor steps between two readings of the focal length, 1–8000, default 550. Larger = faster moves but a bigger possible overshoot at the end of `setZoom`; smaller = finer stop but slower. If `setZoom` overshoots at the long end, halve it (for example 250).

REPLY — `setSD` / `setIC`: `OK`. `getSD`: `SD=µs`. `getIC`: `IC=N`.

ERRORS — `errorRange` · `errorParam`; the stored value is unchanged.

```
→ getSD
← SD=100
→ setIC=250
← OK
```

9 Image stabilization

isIS IS available? — IS

REPLY — `IS=1` if the lens has image stabilization and its IS switch is on. `IS=0` if the lens has no IS or the switch is off.

`enableIS=s` Run stabilization

IS

PARAMETER — `s` : seconds to keep IS running, 1–60. Values up to 3600 are accepted and limited to 60. `0` stops IS immediately.

REPLY — `ISactive=1` or `ISactive=0`. When the time runs out the controller sends `time_is_UP` and then `ISactive=0` on its own.

ERRORS — `errorRange` : negative or over 3600 · `errorParam` .

```
→ enableIS=30
← ISactive=1
   … 30 s later …
← time_is_UP
← ISactive=0
```

10 Controller settings & info

These commands do not need a lens.

`ver` Firmware version — INFO

REPLY — `ISSI LC-PZAE=x.y.z (s/n:N)`. The product name in this reply is how host software tells an LC-PZAE from an LC-2 (`ISSI LC-2=…`).

```
→ ver
← ISSI LC-PZAE=4.9.0 (s/n:17201)
```

`apiver` API version — INFO

REPLY — `API=3.0` : the version of this command set.

`setName=xxxxxxx` · `getName` Controller nickname — SETTING

PARAMETER — Up to 7 characters; longer names are cut to 7, shorter ones are padded with spaces, and an empty name clears it. Stored in non-volatile memory and kept through firmware updates. `setNAME` / `getNAME` are accepted as well (the only spelling before firmware 4.9.0).

REPLY — `setName` : no reply. `getName` : `NAME=xxxxxxx` .

```
→ setName=CAM-01
→ getName
← NAME=CAM-01
```

`getSN` · `getHW` · `getVENDOR` Identification — INFO

REPLY — `SN=N` : serial number · `HW=N` : hardware revision · `VENDOR=text` : dealer/vendor ID (may be empty). These values are set at the factory.

`setProto=X` · `getProto` Lens-bus mode

SETTING

PARAMETER	Firmware 4.9.0: `0` = universal (default), `1` = tail framing, `2` = legacy standard timing of firmware 4.7.x. Firmware 4.7.x: `0` = standard, `1` = tail framing (the "EF-S 18-135 workaround"). Stored in non-volatile memory.
ACTION	Selects how the controller talks to the lens. Universal mode works with the lenses tested so far, including the EF-S 18-135 and third-party lenses that needed `setProto=1` on firmware 4.7.x. Change it only if a lens is not detected (`ping` keeps answering `nolensfound`) with the lens mounted. The controller restarts to apply the new mode.
REPLY	`OK` , then the controller restarts / `Proto=n` . Other values: `errorRange` / `errorParam` , no restart.

`Reboot` Restart the controller

CONTROL

REPLY	`OK` . The controller restarts and answers again after about 4 s (see Power-up). The zoom position is not changed by a restart.

11 Network settings

The controller uses a static IP address. New settings are stored immediately and take effect after `Reboot`; the ISSI LC-2 software sends `ChangeIP`, `ChangeMASK`, `ChangeGW` and `Reboot` in that order. Make sure the new address is reachable from the host before rebooting.

`getIP` · `getMASK` · `getGW` · `getMAC` Read network settings

NETWORK

REPLY — `getIP`: `MAC=a.b.c.d` (the prefix is `MAC=` for compatibility with existing software, the value is the IP address) · `getMASK`: `MASK=a.b.c.d` · `getGW`: `GW=a.b.c.d` · `getMAC`: `MAC=a-b-c-d-e-f` with decimal octets.

```
→ getIP
← MAC=192.168.2.252
→ getMAC
← MAC=238-17-34-17-2-1
```

`ChangeIP=a.b.c.d` · `ChangeMASK=` · `ChangeGW=` Change network settings

NETWORK

PARAMETER — Four decimal numbers 0–255 separated by dots. The IP address and gateway must be ordinary host addresses (not ending in .0 or .255, not 0.x, 127.x or 224 and above); the mask must be a valid mask such as `255.255.255.0`.

REPLY — `IPchanged` / `MASKchanged` / `GWchanged`. Then send `Reboot`.

ERRORS — `errorParam`: invalid value, nothing stored (firmware 4.9.0).

```
→ ChangeIP=192.168.10.40
← IPchanged
→ ChangeMASK=255.255.255.0
← MASKchanged
→ ChangeGW=192.168.10.1
← GWchanged
→ Reboot
← OK
```

> **Unknown address?** The ISSI LC-2 software can search the local network for controllers. Contact ISSI support if a controller cannot be found.

12 Unsolicited messages

The controller can send these datagrams without a preceding command, to the address of the last command it received. Host software should accept and ignore any message it does not expect.

MESSAGE	WHEN
`time_is_UP`, then `ISactive=0`	The `enableIS` time has run out.

13 Error replies

REPLY	MEANING	WHAT TO DO
`nolensfound`	No lens attached, the lens was removed or changed, or it does not respond.	Check the mount, send `ping` again. If it keeps failing, try another `setProto` mode.
`zoomOutOfRange`	`setZoom` target outside the lens's zoom range. Nothing moved; `zoomDone` follows.	Read `zRange`.
`errorFocus`	Focus move not completed, the motor did not stop within 15 s, or no lens.	Check that the AF/MF switch is on AF, then `fRange` and retry.
`errorAperLimits`	Requested aperture is outside the lens's range at the current zoom.	Read `aRange`.
`errorAper`	The aperture move was not completed (also follows `errorAperLimits` for `setAper`).	Retry, or `openAper` and set again.
`errorRange`	Parameter value out of the allowed range. Nothing was changed.	See the command description.
`errorParam`	Malformed parameter (not a number, empty, wrong characters) or a command over 255 characters. Nothing was changed.	Check the command syntax.
(no reply)	Unknown command, or the datagram was lost.	Check spelling and case; resend after the timeout.

A `setZoom` that ends at a mechanical stop or a slipping coupling is not an error reply: the `Zoom=` datagram simply shows a focal length other than the target. Compare the two in the host.

`errorParam` and `errorRange` were introduced in firmware 4.9.0, which validates all parameters before a command is executed. Firmware 4.7.x does not return these replies, so with 4.7.x make sure that parameters are well-formed and within the documented ranges.

14 Programming examples

Both examples follow the same rules: one command at a time, every reply datagram collected, late replies to an earlier command discarded before the next one is sent, and only datagrams from the controller's address accepted. A command that ends with a known reply (`zoomDone` , `focusDone` , `ISactive=`) is read until that reply or an error; any other command is read until no datagram has arrived for 0.5 s. An empty result means that no reply came (unknown command or lost datagram). Replace the address with your controller's.

Python 3

Uses only the standard library.

```
import socket, time

class LCPZAE:
    """LC-PZAE client: one command at a time, replies collected per command."""

    def __init__(self, ip, port=1339):
        self.addr = (ip, port)
        self.s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    def close(self):
        self.s.close()

    def _drain(self):
        """Discard late or unsolicited replies left from earlier commands."""
        self.s.setblocking(False)
        try:
            while True:
                self.s.recvfrom(1024)
        except OSError:
            pass                            # nothing left to read
        self.s.setblocking(True)

    def cmd(self, text, until=None, timeout=None):
        """Send one command and return its replies, one string per datagram.
        until:   stop at the first reply that starts with one of these strings.
        timeout: seconds to wait; 20 s with until, otherwise 2 s.
        Without until, collecting ends 0.5 s after the last reply.
        An empty list means that no reply arrived."""
        if timeout is None:
            timeout = 20.0 if until else 2.0
        self._drain()
        self.s.sendto(text.encode("ascii"), self.addr)
        replies, end = [], time.monotonic() + timeout
        while True:
            left = end - time.monotonic()
            if left <= 0:
```

```
                break
            self.s.settimeout(left if until or not replies else min(left, 0.5))
            try:
                data, sender = self.s.recvfrom(1024)
            except socket.timeout:
                break
            if sender[0] != self.addr[0]:
                continue                    # from another device
            line = data.decode("ascii", "replace").strip("\x00\r\n ")
            replies.append(line)
            if until and line.startswith(until):
                break
        return replies

lc = LCPZAE("192.168.2.252")
try:
    print(lc.cmd("ver"))            # ['ISSI LC-PZAE=4.9.0 (s/n:17201)']
    print(lc.cmd("ping", until=("ISactive=", "nolensfound")))
                                    # 7 replies, or ['nolensfound']
    r = lc.cmd("setZoom=50", until=("zoomDone",), timeout=65)
    print(r)                        # ['Zoom=50', 'zoomDone'] (within 1 mm)
    if r[:1] != ["Zoom=50"]:
        print("zoom did not reach 50 mm:", r)
    r = lc.cmd("setFocus=800", until=("focusDone", "error"))
    if r[-1:] != ["focusDone"]:
        print("focus failed:", r)   # e.g. ['errorFocus']
    print(lc.cmd("setAper=8"))      # ['Iris=8.0']
    print(lc.cmd("getZoom"))        # ['Zoom=50']
finally:
    lc.close()
```

C# (.NET UdpClient)

Builds with .NET Framework 4.x and with .NET 5 or later (C# 5 or later).

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Net;
using System.Net.Sockets;
using System.Text;

class LCPZAE : IDisposable
{
    readonly IPEndPoint lc;
    readonly UdpClient udp = new UdpClient(0);

    public LCPZAE(string ip, int port = 1339)
```

```
    {
        lc = new IPEndPoint(IPAddress.Parse(ip), port);
    }

    public void Dispose()
    {
        udp.Close();
    }

    // Send one command and return its replies, one string per datagram.
    // until:     stop at the first reply that starts with one of these strings.
    // timeoutMs: 20 s with until, otherwise 2 s. Without until, collecting
    //            ends 0.5 s after the last reply. An empty list means no reply.
    public List<string> Cmd(string text, string[] until = null, int timeoutMs = 0)
    {
        if (timeoutMs <= 0) timeoutMs = until != null ? 20000 : 2000;

        // discard late replies to earlier commands
        IPEndPoint from = null;
        while (udp.Available > 0)
            udp.Receive(ref from);

        byte[] b = Encoding.ASCII.GetBytes(text);
        udp.Send(b, b.Length, lc);

        var replies = new List<string>();
        var clock = Stopwatch.StartNew();
        while (true)
        {
            int left = timeoutMs - (int)clock.ElapsedMilliseconds;
            if (left <= 0) break;
            bool waitFull = until != null || replies.Count == 0;
            udp.Client.ReceiveTimeout = waitFull ? left : Math.Min(left, 500);

            byte[] data;
            try { data = udp.Receive(ref from); }
            catch (SocketException e)
            {
                if (e.SocketErrorCode == SocketError.TimedOut) break;
                throw;
            }
            if (!from.Address.Equals(lc.Address))
                continue;                  // from another device

            string line = Encoding.ASCII.GetString(data);
            line = line.Trim('\0', '\r', '\n', ' ');
            replies.Add(line);
            if (until != null && StartsWithAny(line, until)) break;
        }
```

```
        return replies;
    }

    static bool StartsWithAny(string s, string[] prefixes)
    {
        foreach (string p in prefixes)
            if (s.StartsWith(p, StringComparison.Ordinal)) return true;
        return false;
    }
}

class Program
{
    static void Main()
    {
        using (var lc = new LCPZAE("192.168.2.252"))
        {
            Console.WriteLine(string.Join(" | ", lc.Cmd("ver")));
            string[] pingEnd = { "ISactive=", "nolensfound" };
            Console.WriteLine(string.Join(" | ", lc.Cmd("ping", pingEnd)));

            string[] zoomEnd = { "zoomDone" };
            List<string> z = lc.Cmd("setZoom=50", zoomEnd, 65000);
            Console.WriteLine(string.Join(" | ", z));      // Zoom=50 | zoomDone
            if (z.Count == 0 || z[0] != "Zoom=50")
                Console.WriteLine("zoom did not reach 50 mm");

            string[] focusEnd = { "focusDone", "error" };
            List<string> r = lc.Cmd("setFocus=800", focusEnd);
            if (r.Count == 0 || r[r.Count - 1] != "focusDone")
                Console.WriteLine("focus failed: " + string.Join(" | ", r));

            Console.WriteLine(string.Join(" | ", lc.Cmd("setAper=8")));
            Console.WriteLine(string.Join(" | ", lc.Cmd("getZoom")));
        }
    }
}
```

Testing by hand

Any UDP tool works for manual tests, for example `ncat -u 192.168.2.252 1339`: type a command and press Enter (the line ending is ignored).

15 Revision history

DOC	FIRMWARE	DATE	CHANGES
4.9	4.9.0	Nov 2026	New edition for firmware 4.9.0, in the format of the LC-2 guide; replaces version 4.7.19 (June 2021). Describes how `setZoom` and `moveZoom` behave (resolution, direction, end stops, stall), the `zoomOutOfRange` reply and the `zoomDone` spelling, the tuning commands `setSD` / `setIC` / `getSD` / `getIC`, and all commands shared with the LC-2 firmware 1.9.0 (`setVerifyF`, `setMoveT`, `readFocus`, `getVENDOR`, three `setProto` modes, `errorParam` / `errorRange`). Programming examples in Python and C#.

ISSI LC-PZAE API · Version 4.9 · © 2026 Innovative Scientific Solutions, Inc.