ISSI LC-2
Canon® Ethernet Lens Controller
Application Program Interface
Notice

Copyright © 2020 Innovative Scientific Solutions Inc (ISSI). All rights reserved.

ISSI does not warrant that the ISSI LC-2 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, API 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 listed in this license 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.

Contact Information: Innovative Scientific Solutions Inc.
ISSI LC-2 API

API based on UDP Protocol. Port 1339/UDP.

1. Commands

**Action:** Get controller f/w version  
**Command:** “ver<CR>”  
**Returns:** “ISSI LCI-EF-232=4.x.y (s/n:19xxx)”

**Action:** Initialize lens, and get current motors values and ranges (aperture ranges would change when zoom changed)  
**Command:** “ping<CR>”  
**Returns:** “zRange=minZ,maxZ fRange=minF,maxF aRange=minA,maxA Current=ZZ,FF,AA AF=X IS=Y ISactive=S”

where

- minZ– minimum zoom value, maxZ – maximum zoom value
- minF– minimum focus value, maxF – maximum focus value
- minA– minimum aperture value, maxA – maximum aperture value
- ZZ– current zoom value, FF– current focus value, AA - current aperture value
- X = 1 if the focus switch is in AF position on lens and 0 if in MF.  
- Y = 1 if Image Stabilization(IS) switch on lens is enabled, and 0 if disabled or lens have no IS  
- S = 1 if IS is activated and 0 if it deactivated
- “nolensfound” return in case no lens found

**Action:** Move Focus motor on X units
**Command:** “moveFocus=X<CR>” where X could be as positive as negative values.
**Returns:** “Focus=Y”, where Y – current value of Focus motor or “errorFocus” when focus value is not reachable;

**Action:** Move Aperture motor on X step (one-quarter-stop f-number)
**Command:** “moveAper=X<CR>” where X could be as positive as negative values. **Returns:** “Iris=Y”, where Y – current value of Aperture or “errorAperLimits” when aperture value is not reachable

**Action:** Set desired value for Focus motor
**Command:** “setFocus=X<CR>” where X is positive value
**Returns:** “Focus =XXX”, where XXX is the current value (for non USM lens could take more time), “errorFocus” - indicate focus positioning problem

**Action:** alternative way to set lens focus: lens will move focus motor to minimum position, reset encoder counter and then position the focus. Helps to suppress the accumulative lens drift effect for some lenses.
**Command:** “setFocus2=X<CR>” where X is positive value
**Returns:** “Focus =XXX”, where XXX is the current value (for non USM lens could take more time), “errorFocus” - indicate focus positioning problem

**Action:** Set desired value for aperture (in f-number)
**Command:** “setAper=X<CR>” where X is in quarter-stop f-number scale, please use pre-calculated f-stop numbers: 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

Returns: “Iris=Y”, where Y – current value of Aperture, if X is out of range will answer with “errorAperLimits” message

**Action**: detect IS (Image Stabilization) availability for connected lens
**Command**: “isIS<CR>”
**Returns**: “IS=Y” where Y could be 0 or 1, “0” - means lens does not have IS function, “1” - lens has IS function. Also will answer “0” if IS switch is in OFF position.

**Action**: activate IS (Image Stabilization) function for connected lens
**Command**: “enableIS=X<CR>”
**Returns**: “ISactive=Y” where X is the amount of seconds to keep IS active, from [0-3600]. “0” - disable IS, and Y is 0 or 1 - current IS status

**Action**: Set nickname for LC (stored in LC memory)
**Command**: “setName=XXXXXXX<CR>” where XXXXXXX is a 7-symbols name for this controller.
**Returns**: “OK”

**Action**: Get LC nickname (stored in non-volatile memory)
**Command**: “getName<CR>”
**Returns**: “Name=XXXXXX”
**Action:** soft restart LC-2  
**Command:** “`Reboot<CR>`”  
**Returns:** “OK”  

**Action:** explore focus limits  
**Command:** “`refRange<CR>`”  
**Returns:** “`fRange=minY,maxY`” where minY – minimum focus value, maxY – maximum focus value  

**Action:** get lens name  
Command: “`getLens<CR>`”  
**Returns:** “`Lens=XXXXXXX`”, where XXXXXX is the lens name stored in internal lens memory, the lens might not support this command.  

**Action:** get distance range from internal lens encoder in meters  
Command: “`getDist<CR>`”  
**Returns:** “`Dist=XX,YY`”, where XX and YY means that current lens focus distance is in range of XX-YY meters  

**Action:** determine is any lens attached to LC  
Command: “`isLens<CR>`”  
**Returns:** “`LENS_HERE`” in case lens is attached, and “`LENS_NONE`” in case no lens attached  

**Action:** set focus ring in minimum position  
Command: “`goMIN<CR>`”  
**Returns:** “`MIN_OK`”  

2020 © Innovative Scientific Solutions, Inc.
**Action:** set focus ring in maximum (infinity) position  
Command: “goMAX<CR>”  
**Returns:** “MAX_OK”

**Action:** open aperture to maximum for that lens and current zoom position  
Command: “openAper<CR>”  
**Returns:** “OK”

**Action:** reset internal encoder counter  
Command: “setZero<CR>”  
**Returns:** “OK”
2. Examples

Python

```python
import time
import serial
import datetime

COM_PORT = 'COM44'
COM_BAUD_RATE = 115200

ser = serial.Serial(COM_PORT, COM_BAUD_RATE, timeout = 2)
readOut = 0   #chars waiting from LC

MESSAGES = ["ver\r", "ping\r", "setFocus=100\r", "setAper=9\r"]

for i in range(len(MESSAGES)): # LC2 command loop
    now = datetime.datetime.now()
    print (now.strftime("%Y-%m-%d %H:%M:%S\n"), "Sent: ", MESSAGES[i])
    ser.write(str(MESSAGES[i]).encode())
    ser.flush()
    readOut = ser.readline().decode('ascii')
    print (" Answer: ", readOut)
    time.sleep(1)
```

Output:

ISSI Python3 Demo ISSI LCI-EF-232 Script
2020-01-15 12:29:22
Sent: ver
Answer: ISSI LCI-EF-232=4.7.15 (s/n:20002)
2020-01-15 12:29:23
Sent: ping

Answer: zRange=35,35 fRange=0,2508 aRange=1.4,22.0 Current=35,100,9.0 AF=1 IS=0 ISactive=0

2020-01-15 12:29:24
Sent: setFocus=100
Answer: Focus=100

2020-01-15 12:29:25
Sent: setAper=9
Answer: Iris=9.0