ISSI · INNOVATIVE SCIENTIFIC SOLUTIONS

ISSI LCI-EF-232

Canon® EF Lens Controller, RS-232

APPLICATION PROGRAMMING INTERFACE

Document	**Programmer's Reference Manual**
Version	**1.0**
Firmware	**5.8.6 and later**
Date	**October 2026**

Notice

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

ISSI does not warrant that the ISSI LCI-EF-232 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 connection & status
6. Focus
7. Aperture
8. Zoom
9. Image stabilization
10. Controller settings & info
11. Unsolicited messages
12. Error replies
13. Status LEDs
14. Programming examples
15. Revision history

1 Overview

The ISSI LCI-EF-232 drives a Canon EF-mount lens (and compatible third-party EF lenses) from a host computer or embedded system over a standard RS-232 serial port. The host sends short ASCII text commands and the controller answers with one text line per command. You can control focus and aperture, read zoom and focus distance, and use the lens's image stabilization.

> **Already using the ISSI LC-2 Ethernet Lens Controller** (innssi.com/canon)? The LCI-EF-232 is its RS-232 counterpart and uses the same style of text commands, so existing LC-2 integrations carry over with little effort.

Units used in this guide

QUANTITY	UNIT	NOTES
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.
Zoom	Lens zoom-encoder value	Read-only. EF lenses have a manual zoom ring.
Distance	Meters	The lens reports a near/far bracket, not an exact distance.
Time	ms or s, as stated	

2 Connection & message format

Serial settings

PARAMETER	VALUE
Interface	RS-232, DB-9 connector
Baud rate	**115200**
Data bits / parity / stop bits	8 / none / 1 (8N1)
Flow control	None
Character set	ASCII

Commands

- Each command is one line of text ending in **CR** (`\r`). LF or CR+LF is also accepted. The command runs when the line ending arrives, so the characters may arrive at any pace (for example typed in a terminal). An unfinished line is discarded after 10 s without new characters.
- Commands are **case-sensitive** and must match exactly: `getFocus` , not `getfocus` or `getFocus1` .
- Parameters follow an equals sign with no spaces: `setFocus=1200` . Numbers are plain decimal (`-200` , `8.0`); anything else returns `errorParam` and the command does nothing.
- The maximum line length is 128 characters. A longer line is discarded and answered with `errorParam` .

Replies

- Every reply is one line ending in **CR+LF** (`\r\n`).
- Most replies have the form `Name=value` (`Focus=1200`) or a status word (`OK` , `MIN_OK`).
- Errors are single words that start with `error` , or `nolensfound` (see §12). An unrecognized command returns `Unknown cmd: <text>` .
- A focus move interrupted by `stop` replies `STOPPED` instead of its normal reply.

Timing

Send **one command at a time** and wait for its reply before sending the next one. Lens commands run the focus or aperture motor and answer once the motor has stopped. Suggested host timeouts:

COMMANDS	TYPICAL	SUGGESTED TIMEOUT
Queries (`ver` , `getFocus` , `getZoom` , `isIS` , …)	< 50 ms	2 s
`setAper` , `moveAper` , `openAper` , `ping`	50–300 ms	3 s
`lensInit`	1–2 s	5 s
`setFocus` , `moveFocus` , `goMIN` , `goMAX` , `fRange`	0.1–4 s, depends on the lens	**20 s**

> While a focus move is running, the controller answers any command other than `stop` with `errorBusy` right away and does not queue it. A focus motor that does not stop within 15 s is reported as `errorFocus` , so the controller never stays silent indefinitely.

Power-up

After power-up or `Reboot` the controller needs about 3 seconds to start. It then sends two lines:

```
← ISSI LCI-EF-232=5.8.6 :: OK
← Ready. AutoDetect=OFF
```

Wait for the `Ready.` line, or wait 3 seconds, before sending commands.

3 Quick start

A typical session: identify the controller, initialize the lens, read its ranges, then move.

```
→ ver
← ISSI LCI-EF-232=5.8.6 (s/n:1042)
→ lensInit
← lensfound
→ fRange
← fRange=0,2911
→ ping
← zRange=24,105 fRange=0,2911 aRange=4.0,22.0 Current=50,2911,4.0 AF=1 IS=1 ISactive=0
→ setFocus=1500
← Focus=1500
→ setAper=8.0
← Iris=8.0
→ getDist
← Dist=1.20,1.50
```

> **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` .

4 Command quick reference

COMMAND	REPLY	DESCRIPTION
LENS CONNECTION & STATUS		
`lensInit`	`lensfound` / `nolensfound`	Power up and initialize the lens
`ping`	`zRange=… AF=… IS=…`	All ranges and current positions in one line
`isLens`	`LENS_HERE` / `LENS_NONE`	Is a lens attached?
`autoDetect=0|1`	`AutoDetect=0|1`	Initialize the lens automatically when attached
`powerON` / `powerOFF`	`POWER_ON` / `POWER_OFF`	Switch lens power
`getLens`	`Lens=<name>`	Lens model name
FOCUS		
`setFocus=N`	`Focus=N`	Move focus to an absolute position
`moveFocus=±N`	`Focus=N`	Move focus by a relative amount
`getFocus`	`Focus=N`	Current focus position
`fRange`	`fRange=min,max`	Explore the focus limits
`goMIN` / `goMAX`	`MIN_OK` / `MAX_OK`	Drive focus to near / infinity end
`stop`	`OK` (move: `STOPPED`)	Interrupt a focus move
`setZero`	`OK`	Reset the focus encoder counter
`getDist`	`Dist=near,far`	Focus distance bracket in meters
`setFocusT=ms` / `getFocusT`	`OK` / `T=ms`	Focus settle time for slow lenses
APERTURE		
`setAper=F`	`Iris=F`	Set an f-number
`moveAper=±N`	`Iris=F`	Step N one-third stops

`openAper`	`OK`	Fully open the aperture
ZOOM		
`getZoom`	`Zoom=N`	Current zoom position
`zRange`	`zRange=min,max`	Zoom range
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 LCI-EF-232=x.y.z (s/n:N)`	Firmware version and serial number
`getAPI`	`API=1.0`	API version
`getStatus`	`Lens=… LastError=…`	Controller and lens status, no side effects
`setName=xxxxxxx` / `getName`	`OK` / `Name=…`	7-character controller nickname
`getSN` / `getHW` / `getVENDOR`	`SN=` / `HW=` / `VENDOR=`	Serial number, hardware revision, dealer ID
`setProto=0|1` / `getProto`	`OK` / `Proto=0|1`	Lens compatibility mode
`setLED=0|1` / `getLED`	`OK` / `LED=0|1`	Status LEDs on/off
`getRST`	`RST=…`	Cause of the last restart
`echo`	`ECHO_OK`	Link test (no lens needed)
`Reboot`	`rebooting`	Restart the controller

5 Lens connection & status

`lensInit` Initialize the lens

LENS

ACTION — Switches on lens power, establishes communication, reads the lens parameters and opens the aperture fully. Power is switched on only if the lens is fully seated in the mount (see Lens power safety); otherwise the reply is `nolensfound` and power stays off. Required once after power-up (unless `autoDetect=1`) and after changing lenses.

REPLY — `lensfound` or `nolensfound`. The focus range is reset to unknown (`fRange=0,0`) until it is explored again.

```
→ lensInit
← lensfound
```

`ping` Lens status in one line

LENS

ACTION — Returns all ranges and current positions. `ping` never moves the focus: the focus range reads `0,0` until `fRange` (or the first `setFocus` / `moveFocus`) has explored it. Aperture limits change with zoom on many zoom lenses, so send `ping` again after zooming.

REPLY — `zRange=minZ,maxZ fRange=minF,maxF aRange=minA,maxA Current=Z,F,A AF=x IS=y ISactive=s`

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

ERRORS — `nolensfound` : no lens attached or initialized. If a lens is attached but does not respond, the controller tries to initialize it again, so send `ping` again after about 1 s.

```
→ ping
← zRange=24,105 fRange=0,2911 aRange=4.0,22.0 Current=50,0,4.0 AF=1 IS=1 ISactive=0
```

`isLens` Lens attached?

LENS

REPLY — `LENS_HERE` or `LENS_NONE`. Reflects the state set by `lensInit` or auto-detection.

`autoDetect=X` Automatic lens detection

LENS

PARAMETER	`1` = on, `0` = off. Default after power-up: **off** (the setting is not stored).
ACTION	When on, the controller powers up and initializes a lens as soon as one is mounted, and switches lens power off when it is removed. It reports both events with unsolicited messages.
REPLY	`AutoDetect=1` or `AutoDetect=0`. Other values: `errorRange` / `errorParam`.

`powerON` · `powerOFF` Lens power

LENS

ACTION	Switches lens power on or off. `powerON` works only with a lens fully seated in the mount and does not initialize the lens: use `lensInit` for that. `powerOFF` disconnects the lens data lines first, then its power.
REPLY	`POWER_ON` / `POWER_OFF`. `nolensfound` if `powerON` is sent without a seated lens.

Lens power safety. The controller senses the lens through a contact in the mount. Lens power is switched on only after a lens has been seated continuously for 0.3 s, and is switched off immediately (data lines first) when the lens leaves the mount, even in the middle of a focus move. The controller then sends `LENS_POWER_OFF`; an interrupted command replies `nolensfound`. Remount the lens and send `lensInit` to continue.

`getLens` Lens name

LENS

REPLY	`Lens=<name>`, the model name stored in the lens. Some lenses, mostly older ones and some third-party lenses, do not report a name.

```
→ getLens
← Lens=EF24-105mm f/4L IS USM
```

6 Focus

Focus positions are motor encoder counts within the range reported by `fRange`. Explore the range once after `lensInit` with `fRange`. If you skip that, the first `setFocus` or `moveFocus` explores it automatically, so that first command takes longer and the focus ring travels end to end.

Every focus move (`setFocus`, `moveFocus`, `goMIN`, `goMAX`, `fRange`) can be interrupted with `stop`.

`setFocus=N` Absolute focus position

FOCUS

PARAMETER	`N`: target position, within `fRange`.
REPLY	`Focus=N`, the position reached. Non-USM lenses may take longer. `STOPPED` if interrupted by `stop`.
ERRORS	`errorRange`: `N` is outside the focus range (nothing moves) · `errorParam`: not a number · `errorFocus`: the position could not be reached, or the motor did not stop within 15 s.

```
→ setFocus=1500
← Focus=1500
```

`moveFocus=±N` Relative focus move

FOCUS

PARAMETER	`N`: counts to move, positive toward infinity, negative toward the near end. Moves that would pass a range limit stop at the limit.
REPLY	`Focus=N`, the new position. `STOPPED` if interrupted.
ERRORS	`errorParam` · `errorFocus`

```
→ moveFocus=-200
← Focus=1300
```

`getFocus` Current focus position

FOCUS

REPLY	`Focus=N`

`fRange` Explore focus limits

FOCUS

ACTION — Drives focus to both ends and measures the range. The focus ring moves through its full travel. `refRange` is accepted as an alias.

REPLY — `fRange=min,max` . `STOPPED` if interrupted (the range then stays unknown).

```
→ fRange
← fRange=0,2911
```

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

FOCUS

ACTION — `goMIN` moves focus to the closest-focus end, `goMAX` to the infinity end.

REPLY — `MIN_OK` / `MAX_OK` . `STOPPED` if interrupted.

ERRORS — `errorFocus` : the motor did not stop within 15 s.

`stop` Interrupt a focus move

FOCUS

ACTION — Send while a focus move is running: the motor halts within a few milliseconds and the interrupted command replies `STOPPED` . `stop` itself gets no separate reply in that case. `getFocus` then reports where the focus stopped.

REPLY — `STOPPED` (from the interrupted command), or `OK` if nothing was moving.

```
→ setFocus=4000
→ stop
← STOPPED
→ getFocus
← Focus=1290
```

`setZero` Reset the focus counter

FOCUS

ACTION — Sets the lens focus encoder counter to zero at the current position. Typically used after `goMIN` to remove accumulated drift.

REPLY — `OK`

`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*. Lenses without a distance encoder may report `0.00`.

```
→ getDist
← Dist=1.20,1.50
```

`setFocusT=ms` · `getFocusT` Settle time for slow lenses

FOCUS

PARAMETER `ms` : 0–10000, default 1800. Stored in non-volatile memory.

ACTION How long `fRange` waits at each end of travel. Increase it for old or slow lenses whose encoder feedback is unreliable, if `fRange` reports a range that is too small.

REPLY `OK` / `T=ms`

ERRORS `errorRange` : value outside 0–10000 · `errorParam` : not a number.

7 Aperture

The aperture uses a one-third-stop f-number scale. The usable range depends on the lens and, for many zoom lenses, on the current zoom position; `ping` reports it as `aRange`. 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.

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

ERRORS `errorAperLimits`: `F` is outside the lens's range.

```
→ setAper=8.0
← 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.

```
→ moveAper=3
← Iris=5.6
```

`openAper` Fully open

APERTURE

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

REPLY `OK`

8 Zoom

EF lenses are zoomed by hand. The controller reads the lens's zoom encoder.

`getZoom` Current zoom

ZOOM

REPLY `Zoom=N`. Prime lenses report a constant value.

`zRange` Zoom range

ZOOM

REPLY `zRange=min,max`

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. Negative values return `errorRange` .

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

```
→ 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 LCI-EF-232=x.y.z (s/n:N)`

```
→ ver
← ISSI LCI-EF-232=5.8.6 (s/n:1042)
```

`getAPI` API version

INFO

REPLY `API=1.0` : the version of this command set. Host software can check it before using commands added in later versions.

`getStatus` Status in one line

INFO

ACTION Reports the controller and lens state without moving anything (unlike `ping`, it never re-initializes the lens).

REPLY `Lens=l Mount=m AF=a IS=i ISactive=s Range=r AutoDetect=d LastError=e`

- `Lens` : 1 if a lens is powered, initialized and answering
- `Mount` : 1 if the mount contact senses a seated lens (whether powered or not)
- `AF`, `IS`, `ISactive` : as in `ping` (0 without a lens)
- `Range` : 1 once the focus range has been explored
- `AutoDetect` : the `autoDetect` setting
- `LastError` : the last error reply since power-up, or `none`

```
→ getStatus
← Lens=1 Mount=1 AF=1 IS=1 ISactive=0 Range=1 AutoDetect=0 LastError=none
```

`setName=xxxxxxx` · `getName` Controller nickname

SETTING

PARAMETER — Exactly 7 characters are stored. Longer names are cut to 7, shorter ones are padded with spaces. Printable ASCII only. Stored in non-volatile memory and kept through firmware updates.

REPLY — `OK` / `Name=xxxxxxx`

```
→ setName=CAM-01
← OK
→ 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 compatibility mode

SETTING

PARAMETER — `0` = standard (default), `1` = alternative lens-bus timing. Stored in non-volatile memory.

ACTION — Enables a workaround for lenses that do not initialize reliably in standard mode, for example the Canon EF-S 18-135mm and some third-party lenses. Use it only when `lensInit` repeatedly reports `nolensfound` with a lens mounted, then run `lensInit` again.

REPLY — `OK` / `Proto=0|1` . Other values: `errorRange` / `errorParam` .

`setLED=X` · `getLED` Status LEDs

SETTING

PARAMETER — `1` = LEDs show status (default), `0` = LEDs always off, for example in light-sensitive setups. Stored in non-volatile memory.

REPLY — `OK` / `LED=0|1` . Other values: `errorRange` / `errorParam` .

`getRST` Last restart cause

INFO

REPLY — `RST=POR` (power-on) · `RST=EXT` (reset button) · `RST=SYS` (`Reboot` or firmware update) · `RST=BOD` (supply voltage dropped) · `RST=WDT` (internal watchdog recovered the controller) · `RST=OTHER` . If you see `BOD` or `WDT` unexpectedly, check the power supply and contact ISSI support.

`echo` Link test

INFO

REPLY — `ECHO_OK`. Answers immediately and never touches the lens, so it is useful for checking the serial link.

`Reboot` Restart the controller

CONTROL

REPLY — `rebooting`. The controller then restarts and sends the power-up lines about 3 s later. Send `lensInit` again afterwards.

11 Unsolicited messages

The controller can send these lines without a preceding command. Host software should accept and ignore any line it does not expect.

MESSAGE	WHEN
`ISSI LCI-EF-232=x.y.z :: OK` `Ready. AutoDetect=OFF`	After power-up or restart.
`LENS_POWER_ON`, then `lensfound` / `nolensfound`	A lens was mounted (only with `autoDetect=1`).
`LENS_POWER_OFF`	The lens left the mount and its power was switched off (always, with or without `autoDetect`).
`time_is_UP`, then `ISactive=0`	The `enableIS` time has run out.

12 Error replies

REPLY	MEANING	WHAT TO DO
`nolensfound`	No lens seated in the mount, the lens was removed, or it does not respond.	Check the mount, send `lensInit`. If it keeps failing, try `setProto=1`.
`errorFocus`	Focus target not reached, or the motor did not stop within 15 s.	Check 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` with `ping`.
`errorAper`	The lens did not complete the aperture move.	Retry, or `openAper` and set again.
`errorRange`	Parameter value out of the allowed range (for `setFocus`: outside the focus range). Nothing was changed.	See the command description.
`errorParam`	Malformed parameter (not a number, empty, wrong characters) or a line over 128 characters. Nothing was changed.	Check the command syntax.
`errorBusy`	A focus move is running; the command was not executed.	Wait for the move's reply, or send `stop`.
`STOPPED`	Not an error: the move was interrupted by `stop`.	Read the position with `getFocus`.
`Unknown cmd: <text>`	The command was not recognized.	Check spelling and case.

13 Status LEDs

Two LEDs, red and blue, show the controller state at a glance. Turn them off with `setLED=0`.

PATTERN	MEANING
Both flash once at power-up	LED self-test
Blue short blink every 2 s	Running, idle, no lens
Blue solid	Lens attached and communicating
Blue solid, red wink every second	Lens OK, image stabilization active
Red slow blink (1 Hz)	Lens attached but not responding
Red fast blink for 2 s	An error reply was just sent
Both brief flash	Command received
Red/blue alternating	Firmware update in progress (do not disconnect power)
Both dark	No power, or LEDs turned off with `setLED=0`

14 Programming examples

Python 3 (pyserial)

Install with `pip install pyserial`. Replace `COM5` with your port (`/dev/ttyUSB0` on Linux).

```
import serial, time

class LCI:
    def __init__(self, port, baud=115200):
        self.s = serial.Serial(port, baud, timeout=0.2)
        time.sleep(3)                       # allow for power-up after the port opens
        self.s.reset_input_buffer()

    def cmd(self, text, timeout=3.0, until=None):
        """Send a command, return its reply line.
        until: tuple of accepted final replies (for multi-line commands)."""
        self.s.reset_input_buffer()
        self.s.write((text + "\r").encode("ascii"))
        end = time.time() + timeout
        while time.time() < end:
            line = self.s.readline().decode("ascii", "replace").strip()
            if not line:
                continue
            if until is None or line.startswith(until):
                return line
        raise TimeoutError(f"no reply to {text!r}")

lc = LCI("COM5")
print(lc.cmd("ver"))
print(lc.cmd("lensInit", timeout=5))           # lensfound / nolensfound
print(lc.cmd("fRange", timeout=20))
print(lc.cmd("ping"))
print(lc.cmd("setFocus=1500", timeout=20))
print(lc.cmd("setAper=8.0"))
print(lc.cmd("getDist"))
```

C# (.NET System.IO.Ports)

```
using System.IO.Ports;

var port = new SerialPort("COM5", 115200, Parity.None, 8, StopBits.One)
{
    NewLine = "\r\n", ReadTimeout = 20000
};
port.Open();
Thread.Sleep(3000);
port.DiscardInBuffer();

string Cmd(string c) { port.Write(c + "\r"); return port.ReadLine(); }

Console.WriteLine(Cmd("ver"));
Console.WriteLine(Cmd("lensInit"));      // lensfound / nolensfound
Console.WriteLine(Cmd("fRange"));
Console.WriteLine(Cmd("setFocus=1500"));
```

Terminal program

Any serial terminal program, such as PuTTY or Tera Term, works for manual testing: 115200 baud, 8N1, no flow control, send CR (or CR+LF) at the end of each line.

15 Revision history

DOC	FIRMWARE	DATE	CHANGES
1.0	5.8.6	Oct 2026	First edition (`API=1.0`).