

illunis Canon Lens Controller



illunis Manual
Canon Lens Controller



INTRODUCTION

This document details the setup and operation of the illunis CLC Canon Lens Controller.

Rev	Date	Modification
A	4/26/19	First Revision
B	5/17/19	Revised to simplify Programming Interface
C	10/16/19	Added .dll
D	1/24/2020	Updated
E	8/27/2020	Updated for release firmware
F	10/09/2020	Updated for firmware Version :3 Rev :6
G	08/05/21	SerialLogPath added to iSDK
H	08/11/21	Updated for firmware Version: 3 Rev 9 Added RS232 version
I	11/03/21	Added .dll commands from iSDK 9.2.6, Added mechanical drawings
J	05/12/22	Updated Hardware images for Rev B hardware
K	10/10/22	Updated I2C info including connector and descriptions of encoder
L	10/19/22	Updated for firmware version 3 Rev 12 Updated Firmware loading directions Updated iSDKLC commands for iSDK 9.4.4.4
M	10/31/22	Updated Firmware loading directions
N	11/29/22	Updated firmware loading directions
O	03/14/23	Added new iSDK commands
P		

Revisions

This document explains the command interface to the of the Canon Lens controller. Both versions use identical firmware.

The RS232 version supports the Teledyne Genie cameras with external serial communication.

The following table outlines the features.

Feature	RS232	UART	I2C	USB
Power	7-40V Max	5.0V	5.0V	USB (5.0V)
Connector	Internal : JST 4 pin External: Depends on product	JST 4 pin	Internal JST 8 pin	USB Type C Light proof seal JST 8 pin : I2C
Serial Port Baud	57,600 only	57,600 only	100Kbs	57,600 only
Field programmable	Yes	Yes	Yes	Yes
Internal Table EEPROM*	Yes (64K)	Yes (64K)	Yes (64K)	Yes (64K)
Use	Embedded system	Embedded system	Embedded system	Windows PC

* Some models this is optional.

Other documents

CLC AppNote Canon Lens Control using I2C.pdf

For I2C interface use.

CLC AppNote Canon Lens Control Commands.pdf

For CMV cameras with built in CLC (illunis product).



Table of Contents

Quick Start	5
Lens Control using Software	7
Lens Control using Terminal Emulator	8
Mechanical Drawings.....	9
CLC Command Overview	12
CLC Commands	16
Lens Info Commands	17
Aperture Commands	22
Focus Commands	24
Zoom Commands	27
EEPROM Commands	27
Firmware Update	29
Example Lens	33
.dll Commands	35
RS232 Connections	40
UART Connections	40
USB Connections	40
I2C Connections	41
Teledyne Gene Cable	42

Important Note:

Lens controller version 3.14 and above checks the lens AF/MF switch position before adjusting the focus position. If the focus position is not moving, check the position of the lens AF/MF switch.



General

The illunis Canon Lens Controller (CLC) is a mechanical lens mount for a Canon EF lens with an integrated lens controller circuit board. The lens controller USB Version uses a virtual communication port to send and receive commands via a USB 2.0 connection to a computer. The RS232/UAR/I2C implementations are intended for embedded use.

USB Drivers

The lens controller USB 2.0 interface uses the FTDI FT231X USB to UART interface chip.

Drivers can be downloaded from this link:

<https://www.ftdichip.com/Drivers/VCP.htm>

Driver installation guides are available from this link:

<https://www.ftdichip.com/Support/Documents/InstallGuides.htm>

Comm Port Setup

The lens controller port settings are as follows:

Baud Rate: 57600

Parity: None

Data Bits: 8

Stop Bits: 1

Flow Control: None

Cables

The CLC uses a USB type C connector; any commercial USB type C cable may be used to connect the lens controller to the PC.



Lens Control using Software

Lens Control using Software

To assist with writing lens control software, illunis provides a lens control program example for Visual Studio C# as well as an installable executable version. The project source code and executable are available in the illunis.com Help Center. A .Net .dll is available simplifying the configuration and communication to the lens.

Lens Control using a Terminal Program

Any lens command may simply be typed into a Terminal program such as Tera Term which is available here:

<https://osdn.net/projects/ttssh2/downloads/70691/teraterm-4.102.exe/>



Lens Control using Software Continued

Lens Control Application and Source Code

Step 1 Install USB Drivers

The first step in communicating with the lens controller is to install the USB communication drivers. The lens controller USB 2.0 interface uses the FTDI FT231X USB to UART interface chip.

Drivers can be downloaded from this link:

<https://www.ftdichip.com/Drivers/VCP.htm>

Driver installation guides are available from this link:

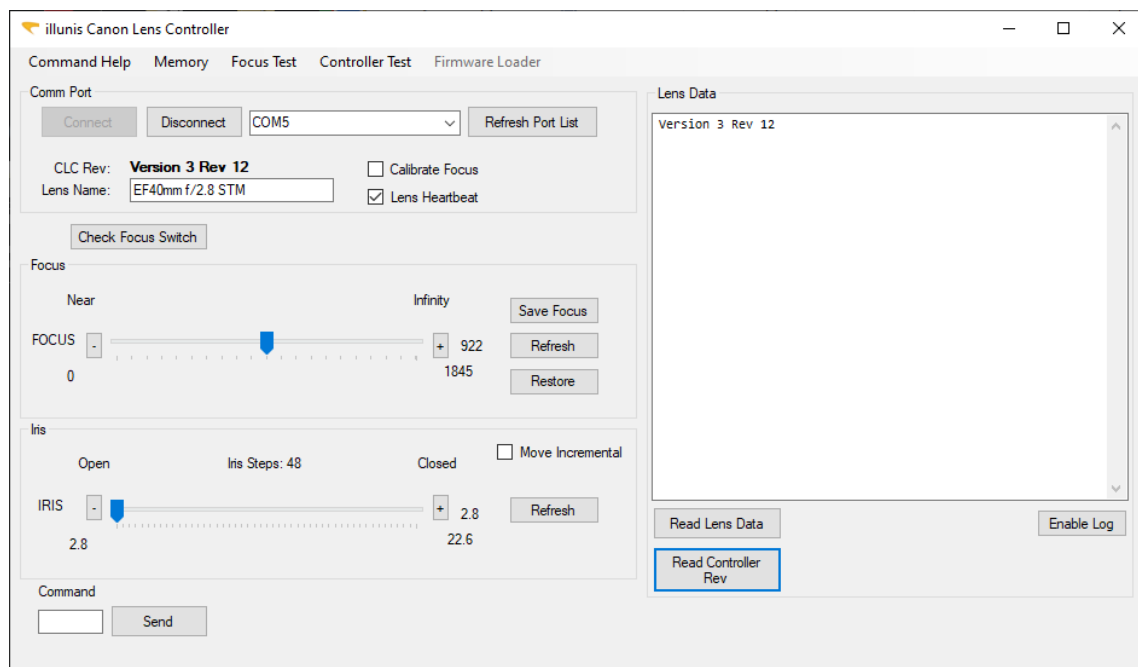
<https://www.ftdichip.com/Support/Documents/InstallGuides.htm>

Step 2 Download illunis Lens Control Software

To assist with writing lens control software, illunis provides a lens control program example for Visual Studio C# as well as an installable executable version and lens control SDK. The sample Visual Studio Project may be opened directly in Visual Studio and compiled. It is provided to show examples of the software interface implemented in order to reduce the time spent on writing application software. A directly executable version of the application may be found in the /bin/x64/Release folder as CanonController.exe.

The Canon Lens Controller installer and source code can be found in the help center at illunis.com

Note: Canon Lens Controller 4.0.8 or above required for Teledyne Genie cameras.





Lens Control using a Terminal Emulator

Lens Control using a Terminal Emulator

Step 1 Obtain and install a Terminal Emulation Program

Tera Term is a free Terminal Emulator for windows available here:

<https://osdn.net/projects/ttssh2/downloads/70691/teraterm-4.102.exe/>

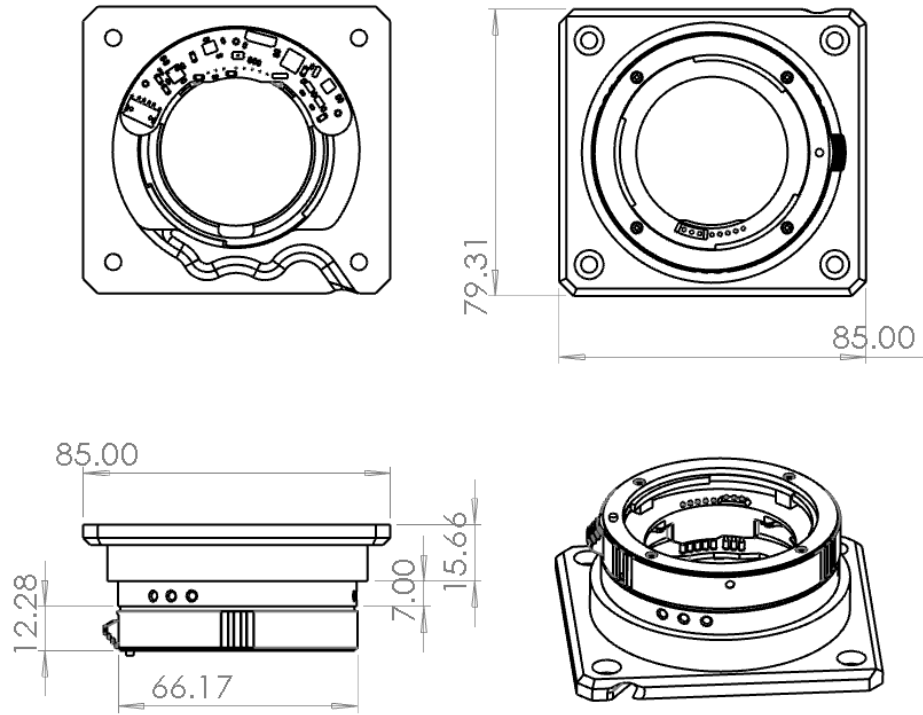
Drivers can be downloaded from this link:

<https://www.ftdichip.com/Drivers/VCP.htm>

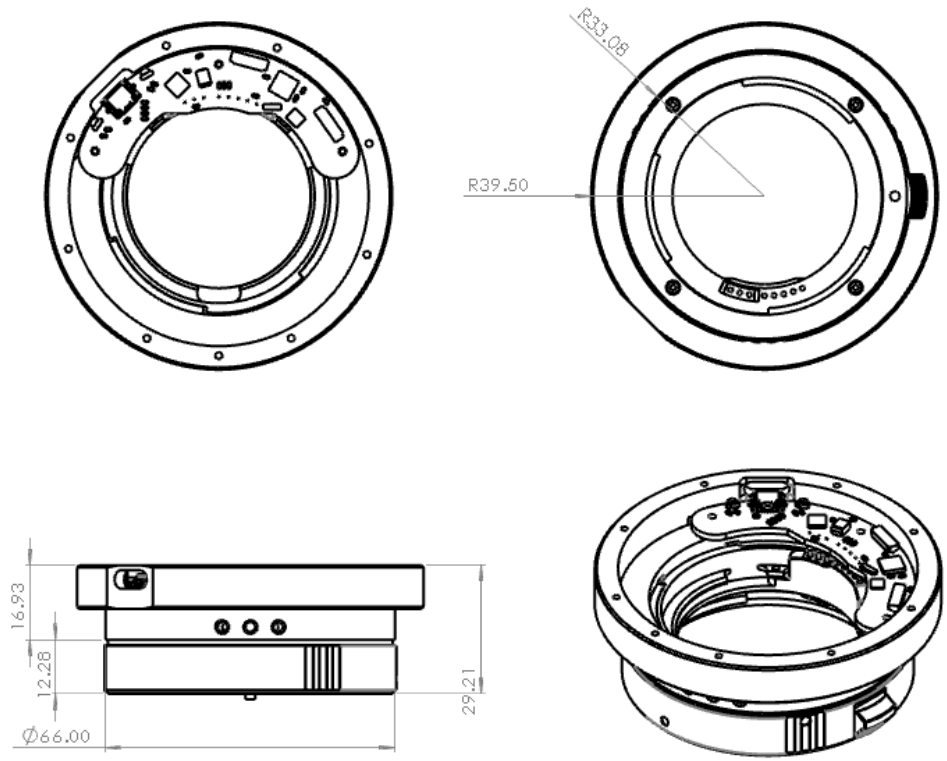
Driver installation guides are available from this link:

<https://www.ftdichip.com/Support/Documents/InstallGuides.htm>

Step 2 Run the Terminal program and issue commands from this manual to control the lens

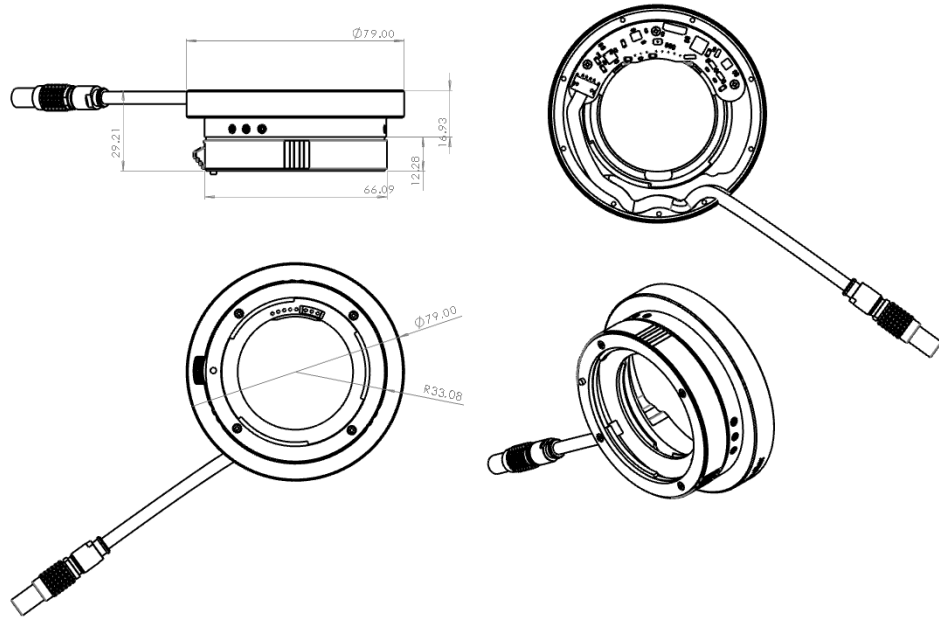


CMV Canon Lens Controller Front
RS232 and USB versions available



Base Canon Lens Controller Front
Various mounting plate options available

Mechanical Drawings



Base RS232 Lens Controller Front
 Various mounting plate options available
 RS232 Connector can be customized (Lemo shown)
 Other options: Bare Wires, Custom connector, etc.

CLC Command Overview

The CLC is a mechanical lens mount for Canon EF lenses with an integrated controller. Please note the following important items:

The CLC can be configured with USB, RS232, UART or I2C interfaces.

The USB interface is controlled through a Windows Com Port at 57600 baud.

The USB 2.0 interface uses the FTDI FT231X USB to UART interface chip.

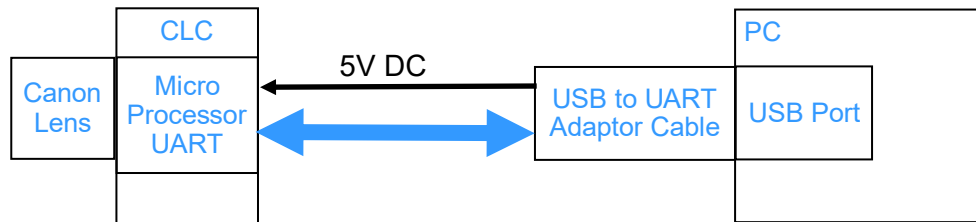
Drivers can be found here:

<https://www.ftdichip.com/Products/ICs/FT231X.html>

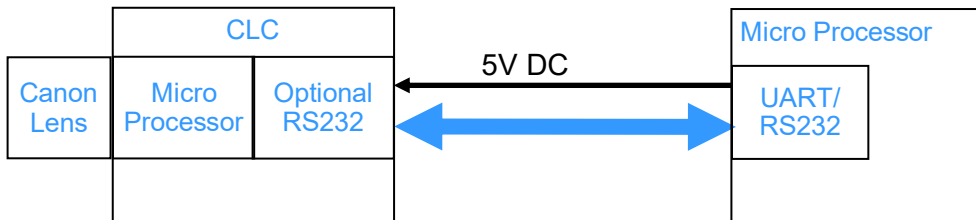
The USB connector powers the CLC-USB.

The I2C interface can be used with 3.3V and 5V busses. Ask for 1.8/2.5V.

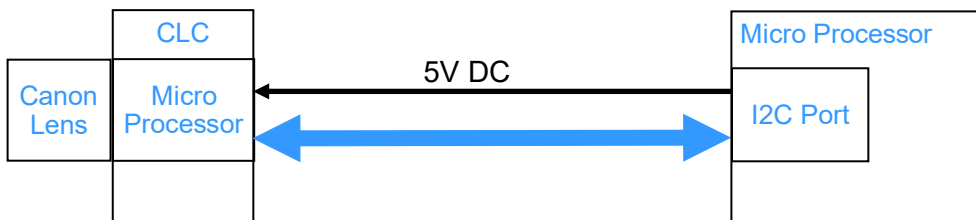
UART COM 57,600 is supported



CLC USB Block Diagram



CLC UART/RS232 Block Diagram



CLC I2C Block Diagram



CLC Command Overview Continued

The CLC interfaces with the Canon EF mounted lens through a command protocol using a micro processor. The micro processor reads data from the EF lens, and commands the lens based on this information. The native lens data is described as follows:

Attached lens is detected by the micro processor and causes the lens to be initialized by the CLC. This initialization performs the following: 1) The zero and infinity positions are set and the encoder/motor positions are measured. 2) The lens status, flags and aperture information are read. 3) The lens internal type code and protocol is read and decoded.

Aperture data is measured in 1/8th F-Stop increments. The F-Stop data is accessed as 10x the value of the F-Stop; thus the value reported from the lens as F28 is actually F2.8. The aperture of the fully open and fully closed positions are provided by the lens.

Focus data is measured in lens encoder and motor units. Individual lens types will have different encoder/motor ranges reported by the lens. The encoder/motor counts for infinity focus and zero focus and is measured from the lens when it is attached.

General purpose control signals are provide by the CLC. There are two GPIO signals that can be set to an input or output at a TTL (5V) level. GPIO signals are accessed through an internal connector on the CLC controller PCB.

The internal EEPROM in the CLC microprocessor can be accessed by the user. EEPROM data is read and written as bytes. An EEPROM dump command is provided. The first 64 bytes of EEPROM is reserved for CLC use. DO NOT WRITE to these locations.



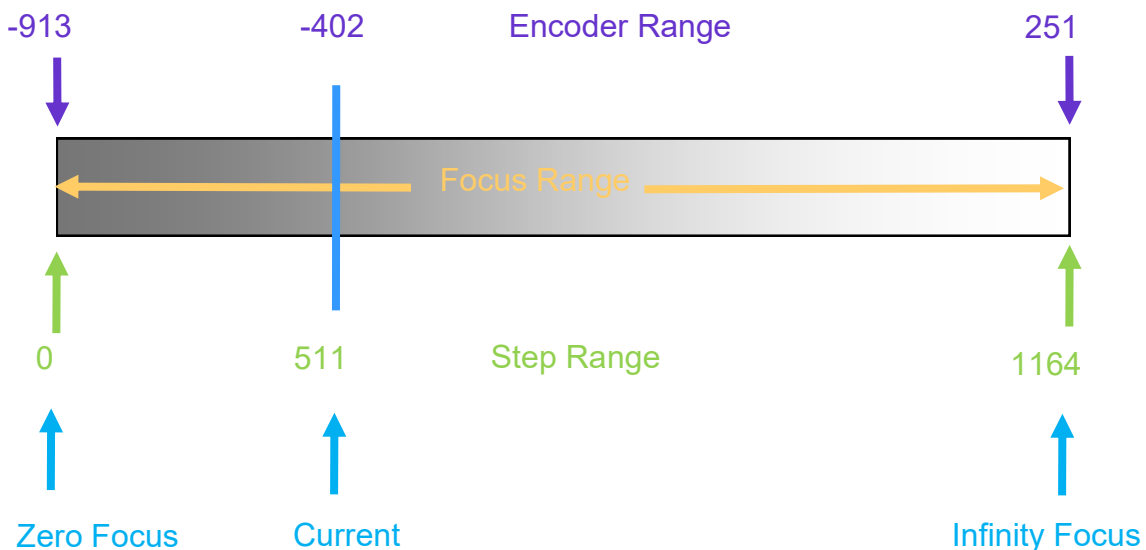
CLC Command Overview Continued

Focus Control:

The lens internal focus mechanism controls the focus position through a stepper or ultrasonic motor in steps. The mechanism uses an encoder to determine its absolute position. The CLC reads the encoder values and reports them in the 'f#' command. Encoder values can be negative and thus confusing to use. Please note that every lens has different encoder values. **(Note All Commands are in normalized motor units, from 0 to N)**

To simplify focus control, the CLC calculates the focus range in steps of motor position. This allows for control in the step based numerical range. The maximum step value can be read with the 'fs' and 'fp' commands which returns #steps and range. A step value of 0 is equal to 'focus zero' and a step value of #steps is equal to 'focus infinity' using the 'fa' focus absolute command.

An additional command is provided to set the focus in percent of full range. This command is 'fc' <value> where value is between 0.0 and 100.0 in a float format.



Example of focus range values for the EF 50mm f/1.8 II lens.



CLC Command Overview Continued

Aperture Control:

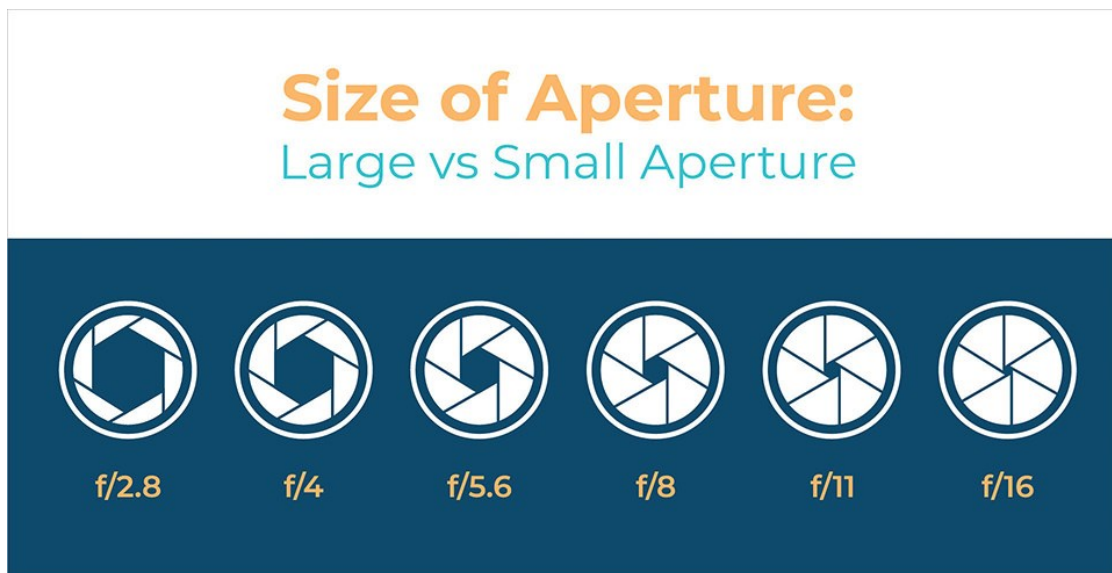
Each lens has an aperture with various lenses. The design of the lens itself determines the range of aperture settings. The CLC provides three methods of setting the aperture.

- #1 The aperture can be forced full open and full closed.
- #2 The aperture can be set in native (internal) lens steps. These steps are in 1/8th of an F-stop. The aperture can be set in absolute or incremental steps.
- #3 The aperture can be set as a F-Stop. F-Stop is specified as 10X the value requested
Example: > ms 180 <CR> = Set F-stop 1.8
Example: > ms 165 <CR> = Set F-stop 16.5

Note : Aperture display status is output as 10X the F-Stop

Example:

Aperture min	: F28	(f/2.8)
Aperture max	: F160	(f/16)
Aperture curr	: F56	(f/5.6)





CLC Commands

The illunis Canon Lens Controller uses a text based interface to command the lens and set parameters. The BAUD rate is fixed to 56,700, 8 bit, no parity. The serial interaction can be operated in the following modes

Quiet Mode : No text is sent by the CLC unless it is commanded by the host.

Normal Mode: (Non Quiet) Camera info and command help is sent on startup

Verbose Mode: This mode sends detailed text data for each command.

Setting Quiet Mode saves the setting in EEPROM and is restored at startup. This allows for a simpler command and control interface to the CLC-USB.

The CLC Normal Mode detects an attached lens and displays the following text at startup:

```
Canon EF Commands: '*'=EEPROM, @I2C=0x10/11
Ver: 3 Rev: 12 Lens Auto PowerDown: 10mins 0secs
ls          Lens status
lc          brief status: focal len,Ap:min,#steps,max
la          Lens attach
ln          Lens name
ge <#>     Get Info
in          Initialize and open aperture
ad          Print aperture info. brief
da          Print aperture info.
pa          Print aperture position
ma <stop>  Move aperture abs. 1/8stop
mc          Move aperture fully closed
mn <pos>   Move aperture inc. 1/8stop
mo          Move aperture fully open
ms          Move aperture to f-stop
mf          Move focus incremental
mi          Move focus infinity
mz          Move focus zero
fa <pos>   Move focus to abs pos.
fc <pos>   Move focus percent
pf          Print focus position
fp          Print focus positions
fs          Print focus steps
f#          Print focus #'s
ep          Print encoder positions
cm          Print focus in cm
fm          Print focus switch position
lf          focus min,max,cur
pz          Print zoom position
qm <0/1>   * Quiet Mode
bw a d     * EEPROM byte write decimal
br a       * EEPROM byte read decimal
ed         * EEPROM dump in HEX
es         * EEPROM save lens state
er         * EEPROM restore lens state
vr         * print version
sn         * print serial number
to         * Get/Set power down time in sec
?          print help
```




Lens Info Commands

The CLC detects the attached lens. The “ls” command shows lens info.:

```
>ls
```

```
Lens Name (From Lens): EF85mm f/1.8 USM  
Prime Lens           : 85mm  
Aperture min        : F18  
Aperture max        : F226  
Aperture curr       : F18  
Aperture motor steps : 58  
Focus steps         : 1695  
Focus Position      : 1692
```

```
>
```

When a lens is dynamically detached or attached a message is displayed.:

```
>Lens detached...  
Lens attached : EF85mm f/1.8 USM  
Lens EEPROM state restored  
>Lens detached...  
Lens attached : EF85mm f/1.8 USM  
Lens EEPROM state restored  
>
```



Lens Info Commands Continued

CLC Commands

- Version
- Serial number
- Quiet Mode
- Get Info

Lens Info Commands

- Lens status
- Lens attach
- Lens name
- Lens status register

Aperture Commands

- Initialize aperture
- Print aperture info
- Print aperture position
- Move aperture absolute 1/8 stop
- Move aperture fully closed
- Move aperture incremental 1/8 stop
- Move aperture fully open
- Move aperture to F-stop #

Focus Commands

- Move focus incremental
- Move focus infinity
- Move focus zero
- Move focus to absolute position
- Move focus percent
- Print focus position
- Print focus positions
- Print focus steps
- List focus min,max,current
- Focus switch position

Zoom Commands

- Print Zoom position

EEPROM Commands

- EEPROM dump
- Write byte
- Read byte
- Save lens state to EEPROM
- Restore lens state from EEPROM



Lens Info Commands Continued

Command: Help (menu)
Syntax: ?
Returns: Table of commands
Description: Returns table of commands and descriptions
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:

```
?  
Canon EF Commands: '*'=EEPROM  
Ver: 1 Rev: 1  
ls          Lens status  
lc          brief status: focal len,Ap min,#steps,Ap max  
la          Lens attach  
ln          Lens name  
...  
vr          * print version  
gs          * get serial number  
sn          * print serial number  
help or ?  print help  
>
```

Command: Print Version
Syntax: **vr**
Returns: Test Version :<number> Rev ::<number>
Description: Returns internal version information from CLC-USB.
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:

```
vr <CR>  
Version :3 Rev :3  
>
```

Command: Serial number
Syntax: **sn**
Returns: :<number>
Description: Returns serial number of the CLC-USB
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:

```
3  
>
```

Command: Quiet Mode
Syntax: **qm <0,1>**
Returns: nothing
Description: Sets quiet <1> or normal <0> mode.
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:

```
qm 0  
qm 1  
>
```



Lens Info Commands Continued

Command: Get Info
Syntax: **ge <#>**
Returns: CLC internal information
Description: Returns data in the form of a signed integer
Prompt is returned if in normal mode.

Request #'s

```
#define CLCD_SN 0
#define CLCD_LENS_ATTACHED 1
#define CLCD_VER_MAJOR 2
#define CLCD_VER_MINOR 3
#define CLCD_APERTURE_MIN 4
#define CLCD_APERTURE_CUR 5
#define CLCD_APERTURE_MAX 6
#define CLCD_FOCUS_MIN 7
#define CLCD_FOCUS_CUR 8
#define CLCD_FOCUS_MAX 9
#define CLCD_ZOOM_MIN 10
#define CLCD_ZOOM_CUR 11
#define CLCD_ZOOM_MAX 12
#define CLCD_MF_ON 13
#define CLCD_IS_ON 14
#define CLCD_LENS_ID 15
#define CLCD_FNUM_MIN 16
#define CLCD_FNUM_CUR 17
#define CLCD_FNUM_MAX 18
#define CLCD_EXTENDED_DATA 19
```

Example:

```
>ge 16
18
>
```



Lens Info Commands Continued

Command: Lens status
Syntax: **ls**
Returns: Table of lens status values
Description: Returns all lens data in table format
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:

```
ls <CR>
Lens Name           : EF 50mm f/1.8 II
Prime Lens         : 50mm
Aperture min       : F18
Aperture max       : F226
Aperture curr      : F32
Aperture motor steps : 58
Focus steps        : 984
Focus Position     : 50
>
```

Command: Lens attach
Syntax: **la**
Returns: nothing
Description: Moves lens focus to find endpoints, sets focus to infinity
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:

```
la <CR>
>
```

Command: Lens name
Syntax: **ln**
Returns: <string>
Description: Returns lens name if in internal data base
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:

```
ln <CR>
EF 50mm f/1.8 II
>
```

Command: Lens internal status register
Syntax: **st**
Returns: <string>
Description: AF/MF=Auto/Manual Focus, IS=Image Stabilizer On
F@Stop/F@Accl/F@Moving/F@Rest = Focus Motor status
A-MotorOn, A@FullOpen = Aperture status

Example:

```
>st <CR>
0x10:AF+F@Stop+F@Rest+A@FullOpen
>
```



Aperture Commands

Command: Initialize aperture
Syntax: **in**
Returns: nothing
Description: Initializes aperture motor and move aperture fully open
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:

```
in <CR>  
>
```

Command: Print aperture info
Syntax: **da**
Returns: <string>
Description: Returns lens aperture min, max, and current settings
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:

```
da <CR>  
Aperture min           : F18  
Aperture max           : F226  
Aperture curr          : F18  
>
```

Command: Print aperture position
Syntax: **pa**
Returns: <string>
Description: Returns lens aperture current - min, and current settings
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:

```
pa <CR>  
0, f18  
>
```

Command: Move aperture absolute 1/8 stop
Syntax: **ma <stop>**
Returns: <stop>,f<number>
Description: Moves aperture to absolute position in 1/8 stop's
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:

```
ma 22 <CR>  
22, f47  
>
```



Aperture Commands Continued

Command: Move aperture fully closed
Syntax: **mc**
Returns: <stop>,f<number>
Description: Moves aperture fully closed
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:
mc <CR>
58, f226
>

Command: Move aperture fully open
Syntax: **mo**
Returns: <string>
Description: <stop>,f<number>
Description: Moves aperture fully open.
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:
mo <CR>
0, f18
>

Command: Move aperture incremental 1/8 stop
Syntax: **mn <stops>**
Returns: <string>
Description: Returns lens aperture min, max, and current settings
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:
mn -4 <CR>
14, f33
>

Command: Move aperture to F-stop #
Syntax: **ms <fstop>**
Returns: <stop>,f<number>
Description: Moves aperture to absolute F-stop. **fstop** is 10x value
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example: Move to f-stop 2.2
ms 22 <CR>
4, f21
>

Example: Move to f-stop 11.0
ms 110 <CR>
41, f108
>



Focus Commands

Command: Move focus infinity
Syntax: **mi**
Returns: nothing
Description: Moves focus position to infinity focus
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:
`mi <CR>`
>

Command: Move focus zero
Syntax: **mz**
Returns: nothing
Description: Moves focus position to zero focus
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:
`mz <CR>`
>

Command: Move focus to absolute position
Syntax: **fa <position>**
Returns: nothing
Description: Moves focus position to absolute position
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:
`fa 333 <CR>`
>

Command: Move focus incremental
Syntax: **mf <delta position>**
Returns: nothing
Description: Moves focus position incrementally from current position
Prompt is returned if in normal mode. Nothing returned in quite mode.
Negative numbers moves focus towards zero focus.
Positive numbers moves focus towards infinity focus.
Focus motors will stop at end points.

Example:
`mf -55 <CR>`
>



Focus Commands Continued

Command: Move focus percent
Syntax: **fc <percent>**
Returns: position:focus steps
Description: Moves focus position to a percent of full range
Prompt is returned if in normal mode. Nothing returned in quite mode.
Percent is 0.0 to 100.0

Example:
`fc 44.4 <CR>`
`44.40:402`
>

Command: Print focus position
Syntax: **pf**
Returns: focus step position
Description: Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:
`pf <CR>`
`511`
>

Command: Print focus positions
Syntax: **fp**
Returns: Focus motor positions
Description: Prints focus positions in motor value
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:
`fp <CR>`
`Fmin:-913 Fmax:251 current:-402`
>

Command: Print encoder positions
Syntax: **ep**
Returns: Focus encoder positions (NOTE: Not motor step position)
Description: Prints focus positions in encoder value
Prompt is returned if in normal mode. Nothing returned in quite mode.
Works only in extended data mode

Example:
`ep <CR>`
`>EZero:19458 Einf:608 current:11568`

Command: Print encoder positions
Syntax: **cm**
Returns: Focus in cm (NOTE: Not motor step position)
Description: Prints focus positions in centimeters.
Prompt is returned if in normal mode. Nothing returned in quite mode.
Works only in extended data mode

Example:
`cm <CR>`
`>Fcm:41`



Focus Commands Continued

Command: Print focus steps
Syntax: **fs**
Returns: <value>
Description: Prints number of focus steps
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:
`fs <CR>`
1164
>

Command: List focus
Syntax: **lf**
Returns: <value>,<value>,<value>
Description: Prints focus values in simple format. (NOTE: Not step position)
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:
`lf <CR>`
-913,251,-402
>

Command: Focus switch position
Syntax: **fm**
Returns: "AF" - Auto focus or "MF" - Manual focus
Description: Firmware 3.12 or greater. Prints the state of the focus switch on the lens.
Starting with firmware 3.12 focus commands will not function if in manual focus mode.

Example:
`fm <CR>`
AF
>

Note:

In firmware version 3.12 or greater, the lens controller will honor the lens focus switch setting. If the lens switch is set to manual focus (MF), the lens controller will silently ignore all focus commands. It also will not calibrate the focus endpoints of the lens when connected in manual focus mode.

Switching back to auto focus will allow the controller to adjust the focus motor again. When switching back to auto focus, it is important to run the "la" command for the lens controller to calibrate the focus endpoints. This will move the focus to each endpoint and lose the previous focus position.

Auto focus does not mean the lens controller will focus on its own, it simply allows the lens controller to send lens focus commands.



Zoom Commands

Command: Print Zoom position
Syntax: **pz**
Returns: <value>, <value>, <value>
Description: Prints Zoom position : min, max, current (Lens is prime if all are equal)
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:
pz <CR>
50mm, 50mm, 50mm
>

EEPROM Commands

Command: Byte Write
Syntax: **bw <address> <data>**
Returns: nothing
Description: Writes byte to EEPROM, all values are decimal
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:
bw 200, 23 <CR>
>

Command: Byte Read
Syntax: **br <address>**
Returns: <value>
Description: Reads byte from EEPROM, all values are decimal
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:
br 200 <CR>
23
>

Command: EEPROM dump
Syntax: **ed**
Returns: <value>
Description: Reads all bytes from EEPROM (Output is in hexadecimal)
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:
ed <CR>
EEPROM (HEX address and data):
0: 1 FF 0 0 0 0 0 0 0 FF FF FF FF D 0 1D
10: 3 0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF
20: AF 1 98 3 98 3 D8 3 FF FF FF FF FF FF FF FF
...
3E0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
3F0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
>



EEPROM Commands Continued

Command: EEPROM Save lens state
Syntax: **es**
Returns: string
Description: Saves lens state to EEPROM, lens state is restored on power up
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:
`es <CR>`
`Lens state saved`
`>`

Command: EEPROM Restore lens state
Syntax: **er**
Returns: aperture position <CR> string
Description: Restores lens state from EEPROM (Aperture and Focus)
Prompt is returned if in normal mode. Nothing returned in quite mode.

Example:
`er <CR>`
`58,f226`
`Lens state restored`
`>`

```
// Reserved EEPROM Locations
#define EE_LENSTYPE          0x0000
#define EE_LENSAPRANGE      0x0001
#define EE_QUITEMODE        0x0002
#define EE_APWREN           0x0008
#define EE_I2CN              0x0009
#define EE_FOCUS_H          0x000A
#define EE_FOCUS_L          0x000B
#define EE_APERTURE_H       0x000C
#define EE_APERTURE_L       0x000D
#define EE_LENSID_H         0x000E
#define EE_LENSID_L         0x000F
#define EE_LENS_SERIAL      0x0010
```

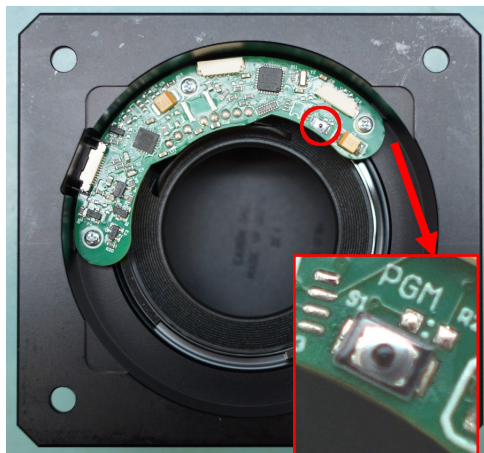


Firmware Update - Introduction

Firmware can be updated through the usb port with the illunis Lens Controller App.

Note: Lens controller shipped with firmware version 4.0 or greater do not need the program button to be pressed. No access to the back of the lens controller PCB is needed.

illunis has 2 versions of the CLC board, each has a different reset method.



The USBC version of the board (at left) has the program button on the right. In this version, the button must be pressed and held down while you click program. You may release the button after programming has begun.



In the newer version of the board the program button is in the center. This button needs to be pressed momentarily to reset the controller. To update the firmware on this board, press and release the button (don't hold) while simultaneously starting the programming.

Firmware Update



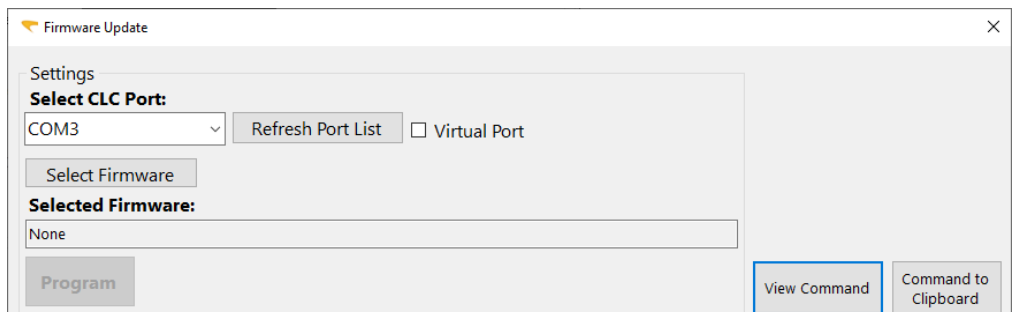
Firmware Update

Firmware Update - With Lens Controller App For installed CLC firmware Version 4 or higher

Firmware can be loaded through the USB port, without pressing the hardware programming button, using the utility illunis Lens Controller App version 4.3.3 or greater.

To upload firmware please do the following:

1. Ensure no other programs are currently connected to the COM port.
2. Open the illunis Canon Lens Controller app.
3. Without connecting the Com Port, choose “Firmware Loader” from the top menu.
4. Select the port the lens controller is using and the firmware .hex or .ill file.
5. If the lens controller is currently version 4.0 or above the firmware will load through the iSDK and a progress bar will be shown.



For installed CLC firmware earlier than Version 4

Remove lens mount assembly from camera body. Access to the back of the lens controller PCB is required for the firmware update.

Follow the directions above; the app will use AVRDUDE for older firmware and a directions will be given to press the PGM button on the controller PCB.

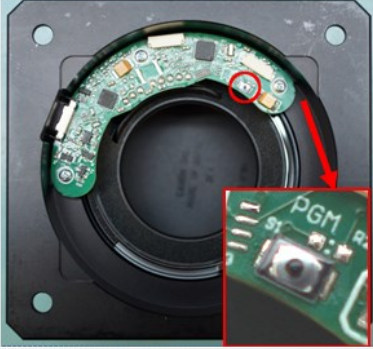
Continued on next page...

Firmware Update - With Lens Controller App


Firmware Update

Firmware Load Directions

The current firmware requires the PGM button to enter bootloader mode. Please press the PGM button according to your board type, then click "Start" to begin



The USBC version of the board (at left) has the program button on the right. In this version, the button must be pressed and held down while you click "Start". You may release the button after programming has begun.



In the newer version of the board the program button is in the center. This button needs to be pressed momentarily to reset the controller. To update the firmware on this board, press and release the button (don't hold) while simultaneously starting the programming.

Start

After clicking "Start", progress will be shown in the blue AVRDUDE output window.

Firmware Update

Settings

Select CLC Port: COM5 Refresh Port List Virtual Port

Select Firmware

Selected Firmware: `ring\FirmWare\Canon Lens Control\Canon_Lens_Control_I2C_4_1\Release\Canon_Lens_Control_Ver_4_0.hex`

Program View Command Command to Clipboard

AVRDUDE Output

```
lock          0 0 0 0 no 1 1 0 4500 4500 0x00 0x00
calibration   0 0 0 0 no 1 1 0 0 0 0x00 0x00
signature     0 0 0 0 no 3 1 0 0 0 0x00 0x00

Programmer Type : Arduino
Description : Arduino
Hardware Version: 3
Firmware Version: 8.0

avrdude.exe: AVR device initialized and ready to accept instructions

Reading | ##### | 100% 0.01s

avrdude.exe: Device signature = 0x1e950f (probably m328p)
avrdude.exe: reading input file "\\diskstation\Engineering\FirmWare\Canon Lens Control\Canon_Lens_Control_I2C_4_1\Release\Canon_Lens_Control_Ver_4_0.hex"
avrdude.exe: writing flash (28492 bytes):
```



Firmware Update Troubleshooting

Issue: avrdude ser_open() can't set com-state for "\\.\COMX"

Solution: This can have many causes; these are a few simple solutions to attempt.

1. Ensure you are selecting the correct com port for the CLC.
2. Repower the PC.
3. Plug the CLC into a different USB port.
4. Change the serial port number for the CLC in windows device manager.

Issue: avrdude.exe: stk500_getsync(): not in sync: resp=0x00

Solution: This error is shown when avrdude is not able to handshake with the CLC's bootloader.

This can be due to:

1. The "pgm" button was not pressed on the lens controller.
2. The "pgm button was not pressed or held at the appropriate time.

When you see this error, it is best to repower the clc before making another programming attempt.

If you continue to receive a sync error from avrdude, there may be an issue with the bootloader on the clc. In this case please contact support.



Example Lens

The Canon 85mm EF 1:1.8 is shown in this example
on lens attachment, normal (not quiet) mode.

```
Canon Lens Controller
(c) illunis LLC 2020
Lens attached : EF100mm f/2.8L Macro IS USM
Lens EEPROM state restored
SerialNum: 1024
```

```
Canon EF Commands: '*'=EEPROM, @I2C=0x20/21
Ver: 3 Rev: 6 Lens Auto PowerDown: 10mins 0secs
ls          Lens status
lc          brief status: focal len,Ap min,Ap #steps,Ap max
la          Lens attach
ln          Lens name
ge <#>     Get Info
in          Initialize and open aperture
ad          Print aperture info. brief
da          Print aperture info.
pa          Print aperture position
ma <stop>  Move aperture abs. 1/8stop
mc          Move aperture fully closed
mn <pos>   Move aperture inc. 1/8stop
mo          Move aperture fully open
ms          Move aperture to f-stop
mf          Move focus incremental
mi          Move focus infinity
mz          Move focus zero
fa <pos>   Move focus to abs pos.
fc <pos>   Move focus percent
pf          Print focus position
fp          Print focus positions
fs          Print focus steps
f#          Print focus #'s
ep          Print encoder positions
lf          focus min,max,cur
pz          Print zoom position
qm <0/1>   * Quiet Mode
bw a d     * EEPROM byte write decimal
br a       * EEPROM byte read decimal
ed         * EEPROM dump in HEX
es         * EEPROM save lens state
er         * EEPROM restore lens state
vr         * print version
sn         * print serial number
to         * Get/Set power down time in sec
?          print help

>
```

Example Lens



Example Lens Continued

Lens status 'ls'

```
Lens Name (From Lens): EF85mm f/1.8 USM
Prime Lens           : 85mm           <- Lens is prime (not zoom)
Aperture min        : F18
Aperture max        : F226
Aperture curr       : F18             <- Fully open
Aperture motor steps : 58
Focus steps         : 1675
Focus Position      : 1675           <- Focus @ infinity
```

>

Move closed 'mc' and Lens status 'ls'

```
Lens Name (From Lens): EF85mm f/1.8 USM
Prime Lens           : 85mm
Aperture min        : F18
Aperture max        : F226
Aperture curr       : F226           <- Aperture reports closed ~ 22
Aperture motor steps : 58
Focus steps         : 1677
Focus Position      : 1677
```

>

Move closed 'mz' and Lens status 'ls'

```
Lens Name (From Lens): EF85mm f/1.8 USM
Prime Lens           : 85mm
Aperture min        : F18
Aperture max        : F226
Aperture curr       : F18
Aperture motor steps : 58
Focus steps         : 1678
Focus Position      : 0              <- Focus reports at zero location
```

>

Version 'vr'

```
Version :3 Rev :6
```

The Canon 100mm EF 80-200mm f/4.5-5.6 USM is shown in this example

>

```
Lens Name           : EF 80-200mm f/4.5-5.6 USM
Zoom Lens min/max/cur: 80mm/200mm/195mm <- Zoom location
Aperture min        : F56
Aperture max        : F281
Aperture curr       : F56
Aperture motor steps : 37
Focus steps         : 9
Focus Position      : 0
```

>



.dll Commands

The iSDK is a .NET .dll provided to aid in connecting to and controlling the lens. It can be found in the Help Center at illunis.com. The following commands are supported.

Note: iSDK 9.2.1.1 or above required for Teledyne Genie cameras.

COM Port Commands

Function: `int` PortOpen(`string` name)

Returns: 1 for success -1 for failure

Description: Initialize the COM

Example:

```
Int err = initPort("COM4");
```

Function: `void` PortClose()

Returns: void

Description: Close the COM port connection

Example:

```
PortClose();
```

Focus Commands

Function: `void` FocusCalibrationOnConnect(`bool` Enable)

Returns: void

Description: On some lens models the encoder range can change each time it's attached. This setting must be set before **PortOpen(string name)** or the focus will calibrate. **True** - Moves the lens to 0 and infinity when attached to calibrate controller to the current encoder range. **False** - does not calibrate focus on attach, can be used on lenses with stable encoders.

Default is False.

Example:

```
FocusCalibrationOnConnect(false);
```

Function: `int` GetFocusNear()

Returns: Near Focus value

Description: Returns the Near Focus Value set by initFocus()

Example:

```
Int FocusNear = GetFocusNear();
```

Function: `int` GetFocusFar()

Returns: Far Focus value

Description: Returns the Far Focus Value set by initFocus()

Example:

```
Int FocusNear = GetFocusFar();
```



.dll Commands Continued

Focus Commands — Continued

Function: `int SetFocusAbsolute(int focus)`
Returns: 1 for success –1 for failure
Description: Sets focus to absolute position between FocusNear and FocusFar
Example: `SetFocusAbsolute(240);`

Function: `int GetCurrentFocus()`
Returns: Current focus value
Description: Returns the current focus value
Example: `Int Focus = GetCurrentFocus();`

Function: `int SetFocusInfinity()`
Returns: 1 for success –1 for failure
Description: Sets focus to farthest position
Example: `SetFocusInfinity();`

Function: `int SetFocusZero()`
Returns: 1 for success –1 for failure
Description: Sets focus to nearest position
Example: `SetFocusZero();`

Function: `string SaveFocusState();`
Returns: ! For success ? For error
Description: Saves current focus position to EEPROM
Example: `String success = SaveFocusState();`

Function: `string RestoreFocusState();`
Returns: ! For success ? For error
Description: Performs a focus calibration, then sets focus to value saved in EEPROM. Note: This will return after command is sent, but lens may take up to 4 seconds to complete.

Iris Commands

Function: `double GetIrisMin()`
Returns: Minimum Iris Value (Most Open value)
Description: Returns the Minimum Iris Value.
Example: `double MinIris = GetIrisMin();`

Function: `double GetIrisMax()`
Returns: Maximum Iris Value (Most Closed value)
Description: Returns the Maximum Iris Value.
Example: `double MaxIris = GetIrisMax();`



.dll Commands Continued

Iris Commands — Continued

Function: `int SetIrisAbsolute(double focus)`
Returns: 1 for success -1 for failure
Description: Sets focus to absolute position between GetIrisMin and GetIrisMax
Example: `SetIrisAbsolute(1.8);`

Function: `int SetIrisStepAbsolute(int steps)`
Returns: 1 for success -1 for failure
Description: Sets focus to step position between 0 and GetIrisSteps
Example: `SetIrisStepAbsolute(20);`

Function: `double GetIrisCurrent()`
Returns: Current Iris value
Description: Returns the current iris value
Example: `double Iris = GetIrisCurrent();`

Function: `int GetIrisCurrentStep()`
Returns: Current Iris step value
Description: Returns the current iris value in steps
Example: `int IrisStep = GetIrisCurrentStep();`

Function: `int GetIrisSteps()`
Returns: number of steps from fully open to fully closed iris
Description: Returns total stepper motor step for iris
Example: `int steps = GetIrisSteps();`

Function: `int SetIrisIncremental(int stops)`
Returns: 1 for success -1 for failure
Description: Moves iris stepper motor number of stops. Can be positive or negative.
Example: `SetIrisIncremental(1);`
`SetIrisIncremental(-2);`

Function: `int SetIrisOpen()`
Returns: 1 for success -1 for failure
Description: Fully opens iris
Example: `SetIrisOpen();`

Function: `int SetIrisClosed()`
Returns: 1 for success -1 for failure
Description: Fully closes iris
Example: `SetIrisClosed();`



.dll Commands Continued

General Commands

Function: `string` GetLensName()

Returns: Lens Name

Description: Returns Lens Name

Example:

```
String Name = GetLensName();
```

Function: `string` GetLensStatus()

Returns: Table of Lens parameters

Description: Returns Lens parameters

Example:

```
String Status = GetLensStatus();
```

Function: `string` GetVersion()

Returns: Lens Controller firmware version

Description: Returns Lens Controller firmware version

Example:

```
String Version = GetVersion();
```

Function: `void` LensHeartbeat(`bool` Enable)

Returns: void

Description: Enables or disables SDK periodic lens presence checks to raise LensPresenceChanged event.

Default: true

Example:

```
LensHeartbeat(false);
```

Function: `event` EventHandler LensPresenceChanged

Returns: none

Description: Event is raised when a lens is attached or detached from the controller if LensHeartbeat is set to true.

Example:

```
myLens.LensPresenceChanged += LensAttachDetach;
```

Function: `bool` LensPresent()

Returns: True if lens attached to controller

Description: Returns current lens status updated by LensHeartbeat. If LensHeartbeat is disabled it will query lens controller.

Example:

```
bool LensPresent = LensPresent();
```



.dll Commands Continued

General Commands — Continued

Function: `string` PortWrite(`string` command)
Returns: returns lens controller response to command (if any)
Description: Used to send any command covered earlier in the guide that does not have a SDK function. Returns lens controller response, "!" for success on commands with no response, "?" for failed or unknown command.

Example:

```
string Response = PortWrite("pz");
```

Function: `void` SerialLogPath(`string` logfile)
Returns: void
Description: Set to a full path and file name to log commands sent to lens controller and responses received. Text will be appended if the file exists or the file will be created if it doesn't exist. If the file can not be opened or created logging will be disabled, to reenale send the SerialLogPath command with a new path/file.

Example:

```
SerialLogPath("D:\Documents\COMlog.txt");
```

Variables (iSDK 9.4.4.4 or higher)

Variable: `bool` LensConnected
Values: true - lens connected; false - no lens
Description: Reports if a lens is connected

Variable: `decimal` FirmwareVersion
Values: reports firmware version as a decimal.
Description: C# only, this is not COM visible.
Example: Version 3 Rev 12 would be 3.12

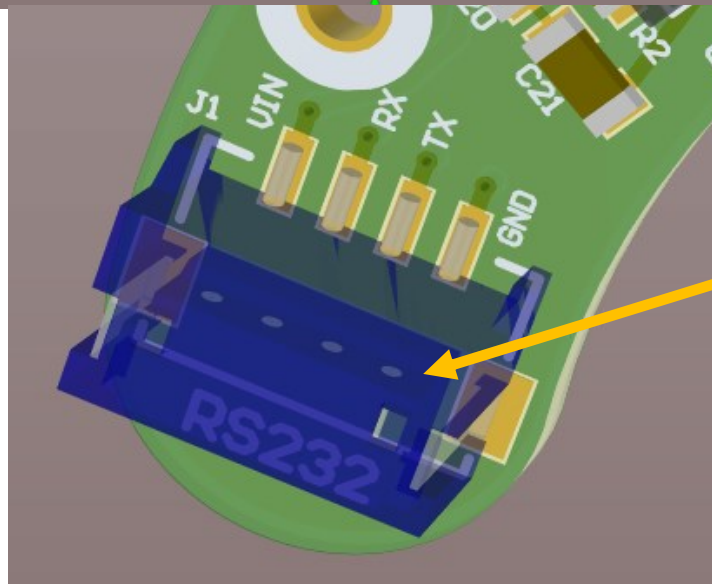
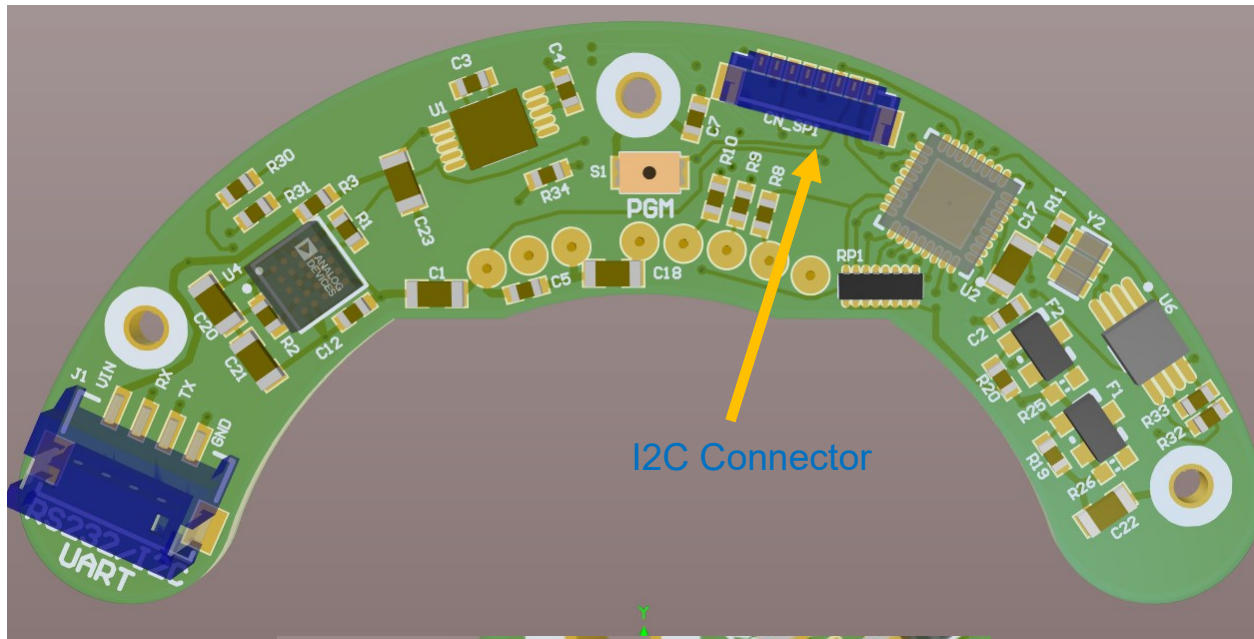
Variable: `bool` PortConnected
Values: true - port open ; false - port not opened
Description: Reports if the com port is open

Variable: `bool` AutoFocusMode
Values: true - lens switch is set to AF; false - lens switch is set to MF
Description: Reports lens switch status.



CLC Hardware communication interfaces

The CLC connections for embedded systems are available on two connectors. The UART, RS232 and USB are available on J1. RS232 Power can be 7-40V or 5.0V depending on build configuration. UART and USB must be 5.0V (USB adapter provides USB power which is 5.0V).



UART
RS232
And
USB adapter
cable
Connector



Manufacturing Drawing: Camera Host to CLC Connection

This diagram shows the CLC PCB Components and Connectors
A illunis camera is used to communicate to the CLC.

I2C communication is 100Khz

The illunis camera must have the expansion connector and fuse populated.
The firmware must support the CLC.

Note that the USB UART can be used simultaneously with the I2C. The 5V connection to the USB UART must be disconnected or share a common ground path with the camera

The pins 1,2,3,8 must be connected. All I2C signals must be 3.3V to 5.0V.

The Camera connection is a 1-1 cable design... i.e. a pin 1 to pin 1 cable is used.

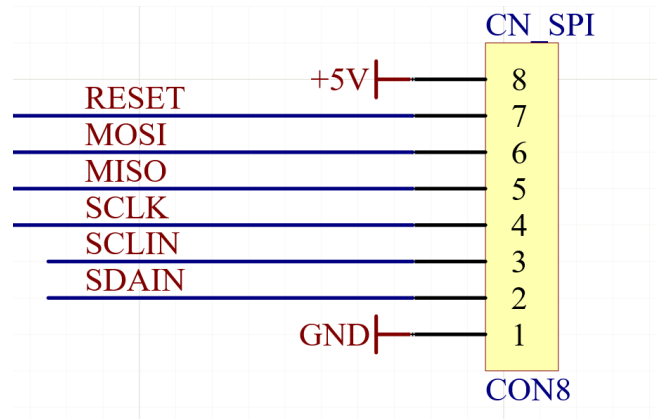
An example (pre made) cable is from digikey:

Part Number : 455-4058

JST

9 Position Cable Assembly

Do not drive the RESET pin



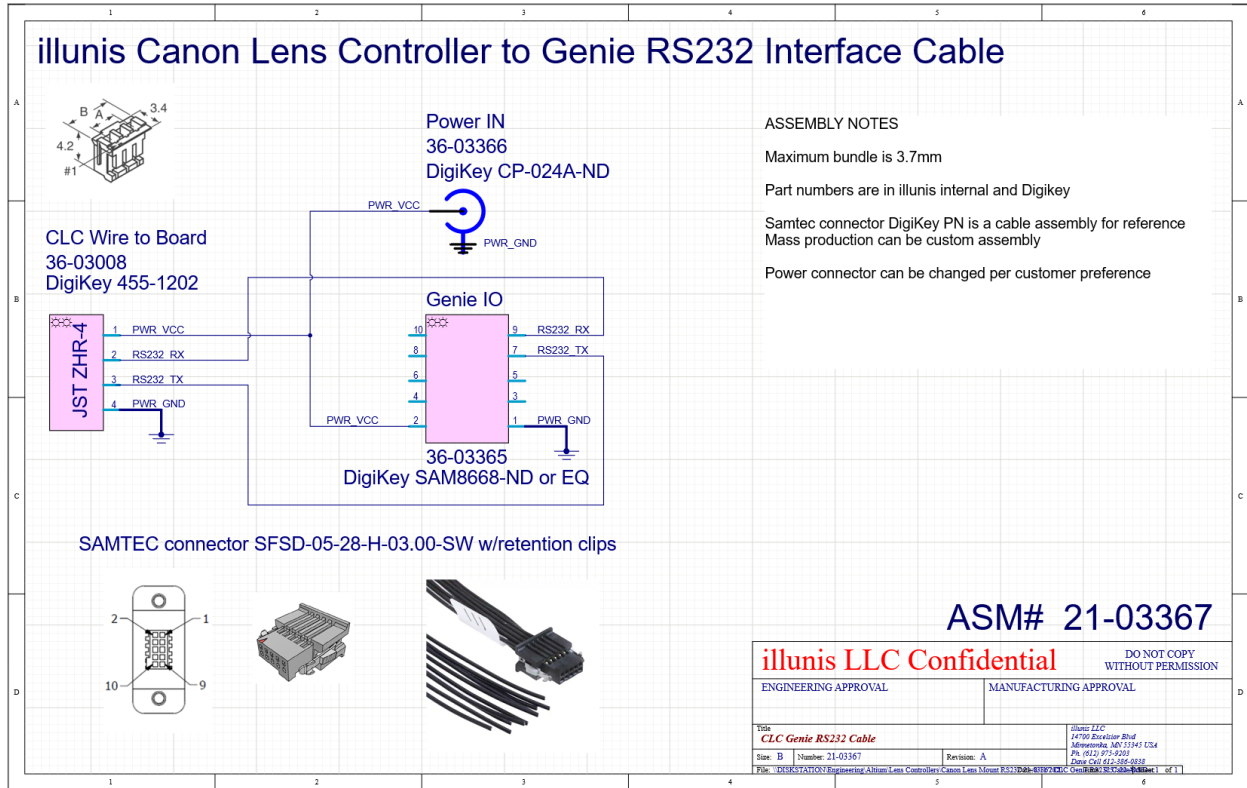
CLC I2C Connection



CLC-RS232 for Teledyne Luminaria Genie Cameras

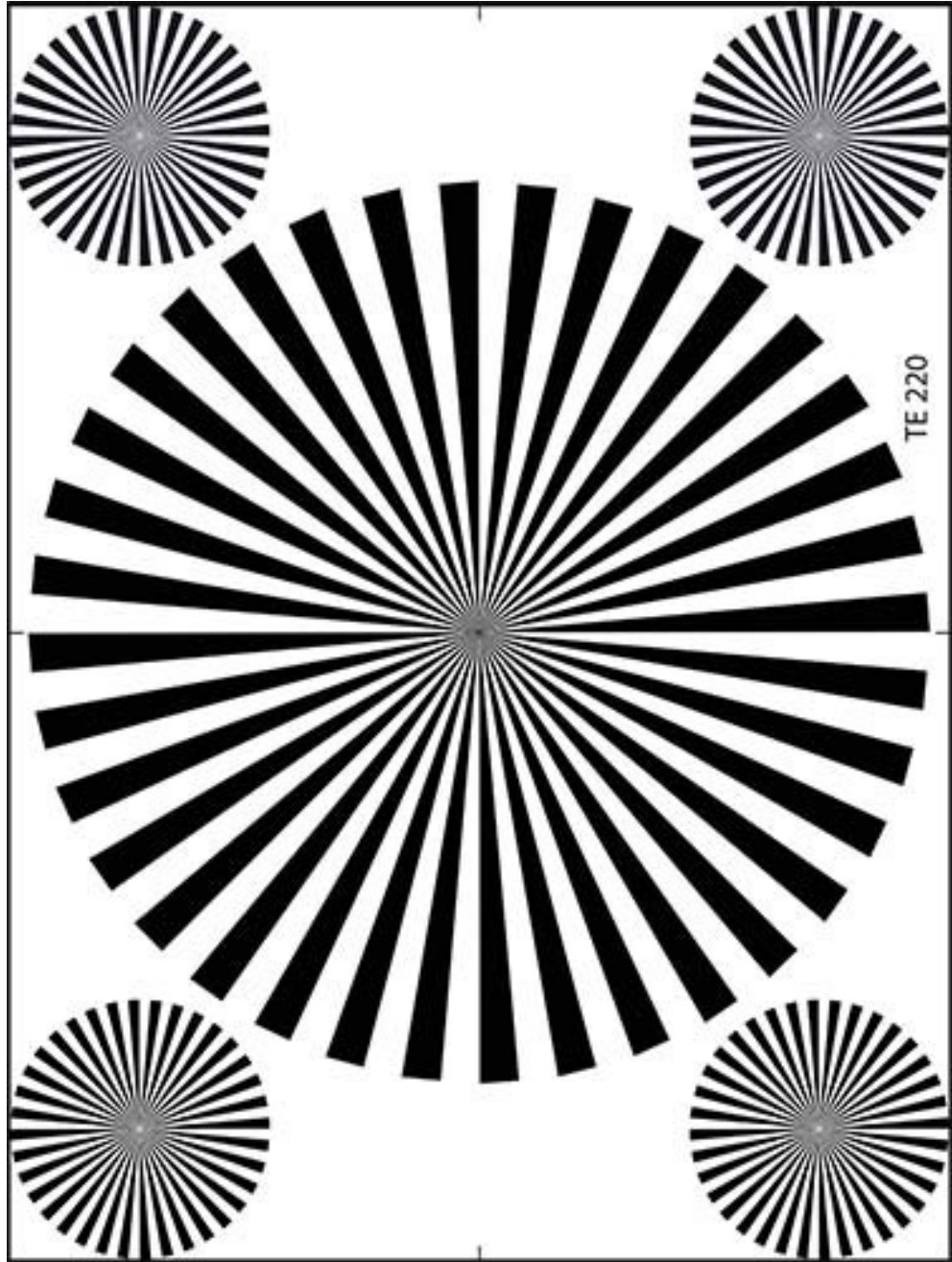
The CLC-RS232 can be used with Genie cameras configured for RS232 control. Power must be configured for 7-40V operation.

A source of cables can be found at:





illunis Canon Lens Controller



For more information on any illunis product, including detailed specifications and options, please visit our website at www.illunis.com, email info@illunis.com, or call illunis at the phone number listed below.

illunis LLC
Headquarters
14700 Excelsior Blvd
Minnetonka, MN 55345 USA

Phone: 952.975.9203
FAX: 952.294.8308

www.illunis.com