Electrical Lens Driver 4

Software Manual

System requirements

- Windows 7, 32/64 bit
- Lens driver: USB 2.0 port
- uEye camera: USB 2.0 port (preferably 3.0)

How to connect the lens to the driver

Connecting the EL-10-30-C:

The Molex flex cable of the EL-10-30-C lens can be plugged directly into the connector of the lens driver. The copper side of the cable has to be upwards and the black clamp has to be closed.

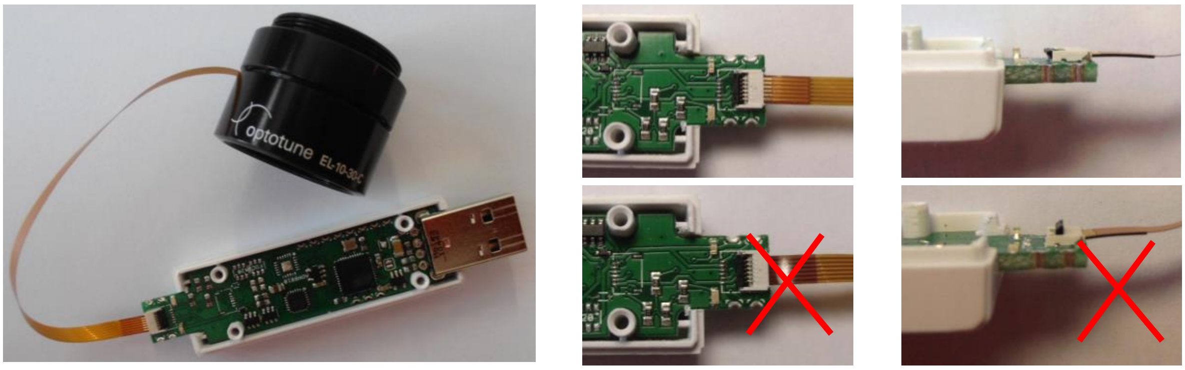


Figure 1: Connecting the EL-10-30-C directly to the lens driver

Connecting the EL-10-30-C to the lens driver with an extension cable:

If a larger distance is required, the easiest way is to use an USB extension cable for the lens driver. If a larger distance in between the lens and the driver is required, an extension as described in Figure 2 can be built. For large distances, shielded cables are recommended to ensure interference-free performance of the I^2C bus. The butterfly connector and the 5cm long transition cable are provided with the lens driver.

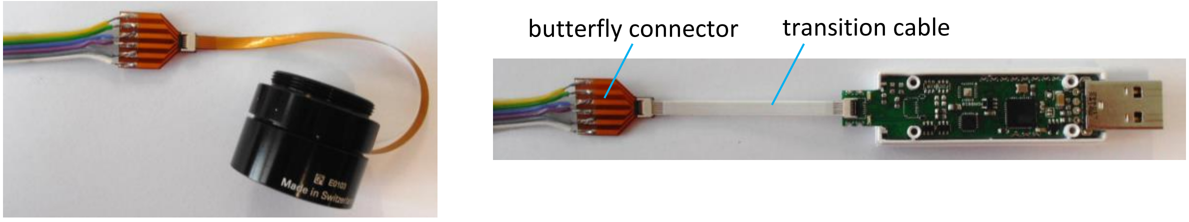


Figure 2: Connecting the EL-10-30-C with an extension cable to the lens driver

Connecting the EL-10-30 compact to the lens driver:

The 30cm long cables of the EL-10-30 compact can be directly soldered to the lens driver. The plus and minus poles are indicated in the figure below.

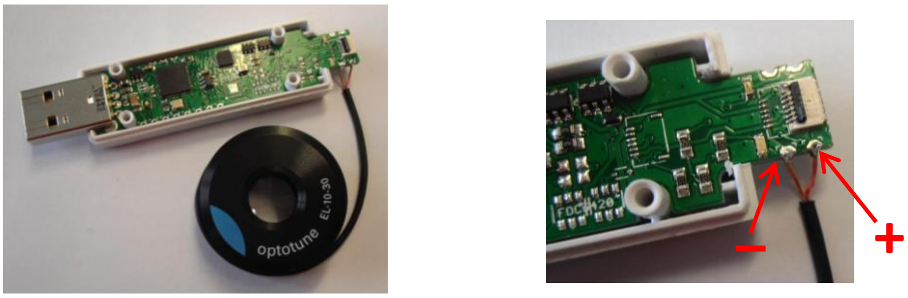
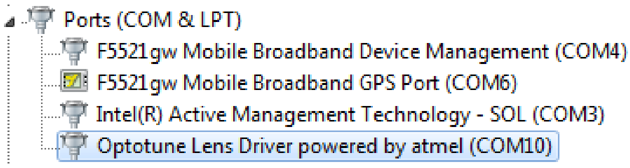


Figure 3: Connecting the EL-10-30-C with an extension cable to the lens driver

Installation of the driver

The steps for the installation of the Windows driver for the LensDriver V4 hardware are described in the following. **Attention: the lens should not be connected to the LensDriver during installation!**

- Plug in the LensDriver (only the LensDriver, **without** a lens connected to it) to a USB port of your PC
- In the popup window that appears, click on "Skip obtaining driver software from Windows Update"
- In the Windows Start menu right click on "Computer" and select "Properties" (or hit Windows+Break)
- In the window that appears, click on "Device Manager"
- Go to the category "Other devices", right click on the "Unknown device" and select "Update Driver Software"
- Select "Browse my computer for driver software"
- Click on "Browse" and navigate to the provided "Optotune LensDriver Windows Drivers" folder and click "OK"
- Click "Next" in the "Update Driver Software" window
- The installation proceeds and installs the driver software
- The driver software is successfully installed if you see a "Optotune LensDriver powered by atmel" in the section "Ports (COM & LTP) " within the "Device Manager"



Ports (COM & LPT)
F5521gw Mobile Broadband Device Management (COM4)
F5521gw Mobile Broadband GPS Port (COM6)
Intel(R) Active Management Technology - SOL (COM3)
Optotune Lens Driver powered by atmel (COM10)

Figure 4: Screenshot of the Device Manager showing correctly installed driver. If the Lens Driver shows an exclamation mark, then the Windows driver has not been loaded correctly. In that case, try connecting to different USB ports (avoid using USB hubs if possible) and restarting the PC.

Installation of the software

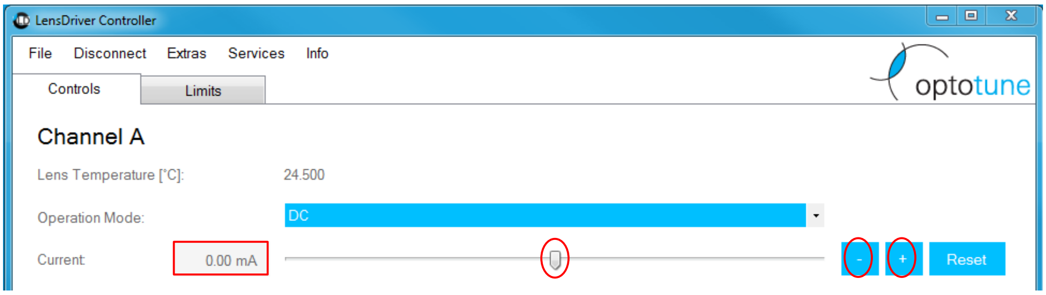
- Run Setup.exe
- Follow the installation wizard

Operating the lens driver

Clicking on **Connect** will connect to the hardware and open the main window with the current control and the temperature readout.

Controls

The output current can be changed, either by shifting the arrow or by using the +/- buttons. Using the +/- buttons together with the Shift key increases the step size. Alternatively, the desired value can be written in the gray box.



Channel A

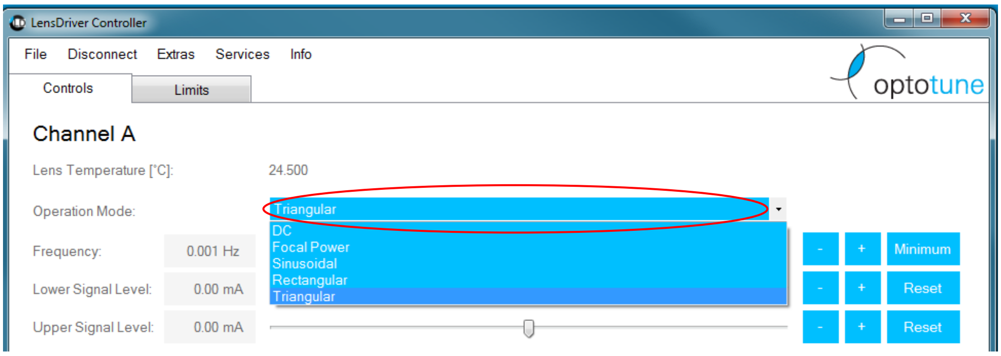
Lens Temperature [°C]: 24.500

Operation Mode: DC

Current 0.00 mA

Figure 5: Screenshot of the main window of the Lens driver software with the current control

The drive signal can be chosen to be a DC, sinusoidal, rectangular or triangular signal with the possibility to set the upper and the lower signal level as well as the driving frequency:



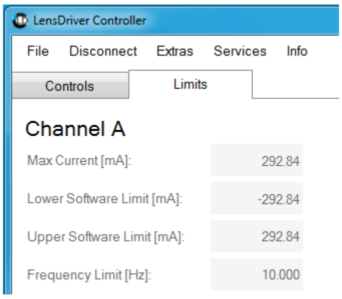
Channel A

Lens Temperature [°C]: 24.500
Operation Mode: Triangular
Frequency: 0.001 Hz
Lower Signal Level: 0.00 mA
Upper Signal Level: 0.00 mA

Figure 6: The drive signal can be chosen to be a DC, sinusoidal, rectangular or triangular signal

Limits

The firmware limits for current and frequency can be set in the **Limits** tab.



Channel A

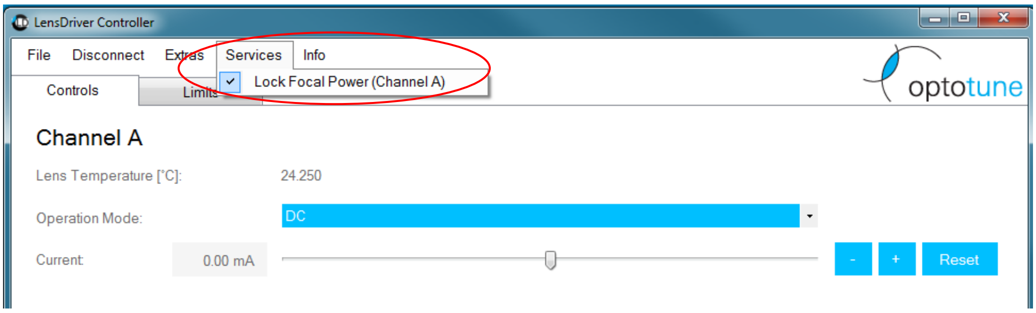
Max Current [mA]: 292.84
Lower Software Limit [mA]: -292.84
Upper Software Limit [mA]: 292.84
Frequency Limit [Hz]: 10.000

Figure 7: Setting firmware limits

Temperature compensation

When heating up the lens, the fluid expands in volume and therefore, the focal length of the lens decreases. The focal length decreases linearly by approximately 0.67 diopters per 10°C temperature increase. This temperature effect is systematic and reproducible and therefore, the focal length can be controlled if the temperature is known. The EL-10-30-C has a built-in temperature sensor (SE97B) with an I^{2}C sensor read-out which can be used for temperature compensation.

In the DC output mode, temperature independent lens operation can be ensured by activating the **Lock Focal Power** button in *Services* which will keep the focal length of the lens constant. Depending on the present temperature, the current applied to the lens is adjusted for compensation of the temperature drift.

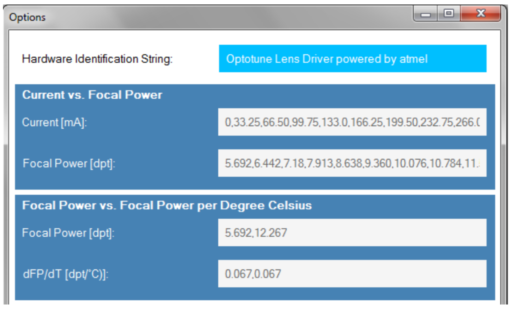


Channel A

Lens Temperature [°C]: 24.250
Operation Mode: DC
Current: 0.00 mA

Figure 8: Lock FV keeps the focal length of the lens constant, independent of the temperature of the lens

In *File → Options*, the relevant parameters for the temperature compensation can be adjusted. Those are the relation focal power [dpt] vs. current [mA] of the lens at room temperature and the temperature change of the focal length [dpt/°C] as a function of the focal power [dpt]. The values for the focal power as a function of the applied current at room temperature for a specific lens can be found in the datasheet of the lens. The datasheet is provided upon request directly from Optotune (sales@optotune.com). With the temperature compensation enabled, a constant focal length value within around ±0.07 dpt for a temperature change of 10°C is achieved.



Hardware Identification String: Optotune Lens Driver powered by atmel

Current vs. Focal Power
Current [mA]: 0,33.25,66.50,99.75,133.0,166.25,199.50,232.75,266.
Focal Power [dpt]: 5.692,6.442,7.18,7.913,8.638,9.360,10.076,10.784,11.

Focal Power vs. Focal Power per Degree Celsius
Focal Power [dpt]: 5.692,12.267
dFP/dT [dpt/°C)]: 0.067,0.067

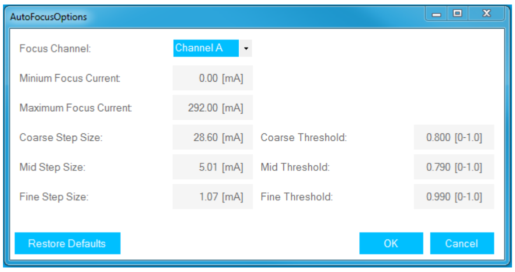
Figure 9: Settings for the temperature compensation

Autofocus

When using the Electrical Lens Driver 4 in combination with a uEye camera, it also offers an autofocus feature. Please note that this feature is for testing purposes only and Optotune does not offer support for this part of the software. The autofocus software is only tested yet with the camera model *IDS UI-3580CP-C-HQ*.

Settings:

In *Extras → uEye Viewer → Lens driver → Auto Focus Settings,* the settings have to be set according to:



Focus Channel: Channel A
Minium Focus Current: 0.00 [mA]
Maximum Focus Current: 292.00 [mA]
Coarse Step Size: 28.60 [mA]
Coarse Threshold: 0.800 [0-1.0]
Mid Step Size: 5.01 [mA]
Mid Threshold: 0.790 [0-1.0]
Fine Step Size: 1.07 [mA]
Fine Threshold: 0.990 [0-1.0]

Figure 10: Settings for the Auto Focus option

The camera settings can be changed in *Camera → Settings.*

Open the **uEye Viewer** and press **Play**. For autofocus, click on the image and the current applied to the lens is adjusted for focusing of the image.

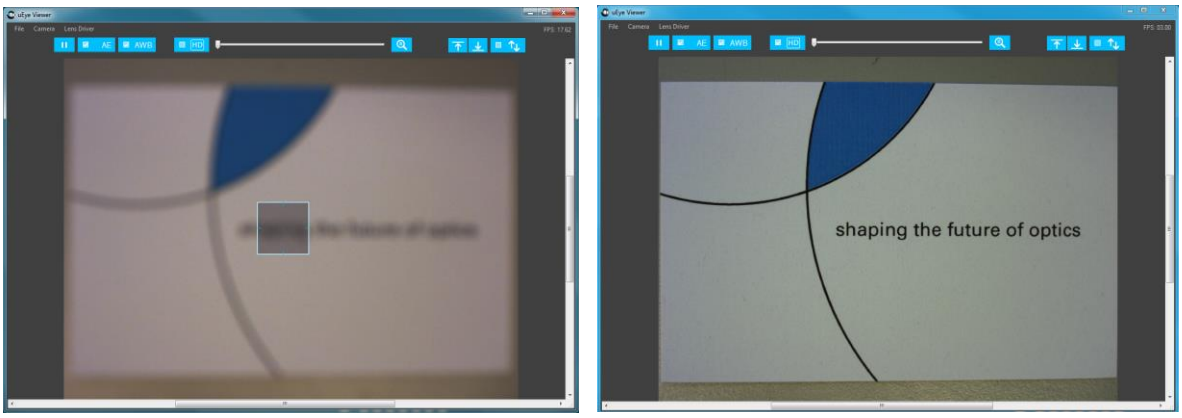
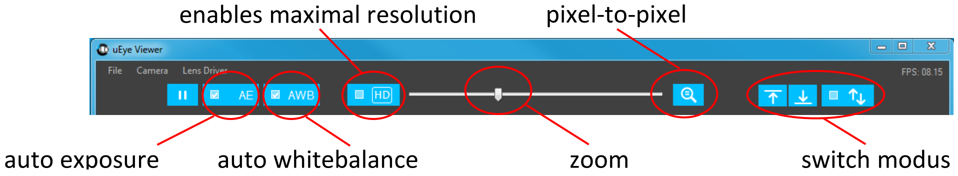


Figure 11: Autofocus is achieved by clicking on the image

Options:



Pixel-to-pixel modus: in the 'pixel-to-pixel' modus, the resolution is chosen in the way that one pixel of the image corresponds to one pixel of the screen.

Switch modus: In this modus, the driver switches between two current values with a rectangular driving signal. The lower and upper current level can be determined in the following way. First perform autofocusing for the lower level (click on the part of the image which should be in focus) and afterwards click on the lower limit button (arrow with an underline). Repeat the steps for the upper limit (arrow with an overline). If you than enable the button with the two arrows, the current will switch between the two values. The frequency can be changed in the *Controls* window.

Software protocol for serial-port

This section describes the protocol used to control the lens driver in case customized software is written. The protocol can be implemented using any programming language. An example for the CRC calculation is only given in C.

The documentation shows commands sent out by user software (i.e. PC) and the answer to be expected as a response from the driver as well as the action that is taken by the driver as a result of the command. Commands in quotes (") have to be sent as a string (or char array) with **each letter representing one byte** except for "\r" meaning "carriage return", "\n" meaning "new line" and are represented by a single byte (ASCII code). Letters in red are sent as ASCII characters and letters in blue are coded as binary bytes.

Connection:

The Microcontroller used runs a virtual com port driver provided by Atmel. The connection settings are:

- Baudrate: 115200 (others may also work since the port is virtual)
- Parity: None
- Stop Bits: One

Handshake:

The handshake command is used to check if the hardware is ready and running. It can also be used as a reset function as it will reset the current to zero. Other commands are also valid without sending this command after initialization, it is optional.

PC sends	Driver answer	Driver action	Comment
"Start"	"Ready\r\n"	reset current to zero, flush input buffer	optional command

Current set commands:

A current set command sets a new output current for a channel. A current set command is constructed as AxxLH. It always starts with the channel identifier "A" for channel A[1]. The current sent out (i_o) is coded as a 12bit value (x_i) that is mapped to the maximum hardware current range (i_c). The maximum hardware current range (calibration value i_c) can be read and written from the driver, see section about calibration commands. By default, i_c = 293mA. The formula to calculate the value x_i for a certain current is:

$$\mathbf{x_i = i_o / i_c * 4096}$$

Example for a current of 50mA for a calibration value of 293mA:
50mA / 293mA * 4096 = 699

---

[1] Currently, the LensDriver only hosts one channel. In future, a second channel might be added to control two lenses simultaneously.

Commands to set current are sent as one char selecting the channel, followed by two bytes containing the 16bit signed current value x_i [-4096 to 4096], followed by two bytes for the CRC16 checksum which is calculated over all the preceding data bytes (see CRC16 code section on how to implement it) with the low byte of the CRC sent first: "LH" means '**L**ow Byte', '**H**igh Byte'. If a value is out of boundaries (i.e. 5000) it is limited to 4096 by the firmware, no overflow will occur. If the software current limit is set lower than 4096 (i.e. 3000, see calibration commands), current requests bigger than the software limit will be reduced down to the limit (i.e. 3000). The driver does not reply to correctly received current commands to make the possible update rate as fast as possible. Invalid commands (i.e. incorrect CRC) will be answered with "N\r\n".

Char coding for the current set command:

"A"	Channel A

PC sends	Driver answer	Driver action	Comment
"AxxLH"	none	set channel A current to xx	xx is a signed 16bit integer with a value between -4096 and 4096, high byte sent first
"AxxLf"	"N\r\n"	none	f = error in CRC byte, this line applies for all detected CRC errors.

Exemplary current set command

Command type: Set current
Channel: A
Current value: 1202

Resulting command:
Byte 0: 0x41 (corresponds to ASCII "A")
Byte 1: 0x04 (high current byte) / 0b00000100
Byte 2: 0xb2 (low current byte) / 0b10110010 -> with byte 1 the total current bits are 0b0000010010110010 which equals to 1202 dec
Byte 3: 0xd2 (low CRC byte)
Byte 4: 0xa1 (high CRC byte)

For verification of the command, the CRC checksum (see details below) over the whole 5 bytes of the command can be calculated which should be zero.

Mode commands

Mode commands allow to access frequency modes (see table). A mode change command always starts with "M", followed by a "w" to write the command. Reading commands back is by sending an "r" instead of a "w" is possible but not tested and recommended. The "w" is followed by a char identifier that selects the signal type ("S", "Q", "T" or "D", see table) and finally a char selecting the channel ("A"). The command is finished by a 16bit CRC calculated from the four command bytes. The low byte of the CRC is sent first.

To set the property of the selected mode (i.e. frequency and currents) a signal property change command is used. This command starts with a "P" followed by a "w" to write the command. Again, reading commands back is by sending an "r" instead of a "w" is possible but not tested and recommended. The "w" is followed by a char identifier that selects the property to be changed ("U", "L" or "F", see table). Next a char selecting the channel ("A") is sent. The next four bytes sent represent the data for the command. It is a 32-bit unsigned integer for the frequency or a 16bit signed integer followed by two dummy bytes for the current. The frequency needs to be multiplied by 1000 (fixed comma representation). Example: For a frequency of 12Hz the integer sent out needs to be 12000 (0x00002EE0 as a 32bit hex). The bytes to be sent out (in this order) are: 0x00, 0x00, 0x2E, 0xE0. The command is finished by a 16bit CRC calculated from the eight command bytes. The low byte of the CRC is sent first.

Char coding for the mode change command:

"A"	Channel A
"S"	Sinusoidal signal
"Q"	Square signal
"D"	DC signal
"T"	Triangular signal

Char coding for the signal property change command:

"A"	Channel A
"U"	upper swing current [-4095 to 4095]
"L"	lower swing current [-4095 to 4095]
"F"	Frequency (in 'value in [mHz] = value in [Hz] *1000')

Mode command examples:

PC sends	**Driver answer**	**Driver action**
"MwSALH"	"MSALH\r\n"	set channel A to sinusoidal waveform
„PwFAyyyyLH"	nothing	set channel A frequency to yyyy (in mHz, 32bit value)
„PwUAyyddLH"	nothing	set channel A upper signal current to yy (12 bit value) (dd are two dummy stuffing bytes and can be 0)

Calibration commands

Calibration commands are used to set and read calibration values and software current limits. These values are stored in the EEPROM, a non-volatile memory, and are kept there for years.

Software current limits can be set to protect a lens from overcurrent or to fix the maximum focal power. Limits also apply in analog input mode.

> **Note**: in analog input mode the input voltage 0-5V is mapped to [-4095 to 4095] in a 10bit resolution. Software current limits apply.

A calibration command always starts with "C" followed by "r" to road or "w" to write the command value. The next byte determines the type of value. It can be "M" for the maximum current calibration value (in [mA*100]), "U" for the upper software current limit or "L" for the lower software current limit. Next a char selecting the channel ("A") is sent followed by two data bytes (16-bit integer). The maximum current calibration value is the output current measured for the x_i = 4095 value and does normally not need to be changed as design guarantees 1% accuracy to the default value. The default value is 29284 (292.84mA).The software current limits are sent as a 12bit current value and can also be negative like a normal current command. As usual the command is finished by a 16bit CRC calculated over the seven command bytes.

> **Note:** the upper and lower limits are always saved to EEPROM that allows "only" 100'000 write cycles before it starts wearing out. There is no limit on reading.

Char coding for the Calibration command:

"A"	Channel A
"M"	Maximum hardware current [mA]*100 (calibration value)
"U"	Upper software current limit [-4095 to 4095]
"L"	Lower software current limit [-4095 to 4095]

Calibration command examples:

PC sends	**Driver answer**	**Driver action**	**Comment**
"CrMAddLH"	"CMAyyLH\r\n"	none	read channel A calibration, "dd" are two dummy bytes and can be 0

"CwLAxxLH"	"CLAxxLH\r\n"	write xx as new lower limit of channel A	limit is immediately active, answer is for double checking
"CrUAddLH"	"CUAyyLH\r\n"	none	read channel A upper current limit, "dd" are two dummy bytes and can be 0

Temperature reading

If the lens connected supports temperature reading, this command will read back the temperature of the integrated sensor (NXP SE97B). To convert the value to a temperature the following formula can be used:

Temperature [°C] = data * 0.0625 [°C]

For more details about the conversion please refer to the datasheet of the temperature sensor.

The temperature read command starts with a “T” followed by the channel selection byte (“A”). The command is completed with the 16bit CRC calculated over the two command bytes. Low CRC byte is sent first.

Possible temperature commands:

PC sends	Driver answer	Driver action	Comment
“TALH”	“TAEddLH\r\n”	Get temperature reading of lens on channel A	E is 0x00 if read is successful, 0xff otherwise. “dd” is the temperature data.

CRC calculation

A 16-bit CRC code (CRC-16-IBM standard) is used to check for communication errors. The code examples below use the reverse polynomial implementation.

- Normal polynomial: 0x8005
- Reverse polynomial: 0xA001
- Initial value to be used: 0x0000

The following example shows the implementation in the Lens Driver firmware, which is written in C:

```
uint16_t crc16_update(uint16_t crc, uint8_t a)
{
        int i;
        crc ^= a;
        for (i = 0; i < 8; ++i)
        {
                if (crc & 1)
                crc = (crc >> 1) ^ 0xA001;
                else
                crc = (crc >> 1);
        }
        return crc;
 }

/* Example usage of the CRC function */
uint8_t data[] = { 0x41, 0x04, 0xb2 };  // Example: Set current level to 1202
uint16_t checkcrc(void)
{
        uint8_t crc = 0, i;
        for (i = 0; i < sizeof(data) / sizeof(data[0]); i++)
        {
                crc = crc16_update(crc, data[i]);
        }
        return crc; //returns checksum over all elements
 }
```

To check a data array for valid data, attach the CRC as the last two bytes in the order of low byte first, high byte second. If all is checked in CRC function, it must return 0 if no error is contained in the message.

Example : `uint8_t test[5] = {0x02, 0x1c, 0xb8, crc&0xFF, crc>>8}`

The following example shows the implementation in the Lens Driver software, which is written in C#:

```
private byte[] AddCRC(byte[] command)
{
        UInt16 CRC = 0;

        byte[] commandWithCRC = new byte[command.Length + 2];

        CRC16IBM CRC16IBMCalculator = new CRC16IBM();
        CRC = CRC16IBMCalculator.ComputeChecksum(command);

        Array.Copy(command, 0, commandWithCRC, 0, command.Length);

        commandWithCRC[commandWithCRC.Length - 2] = (byte)(CRC&0xFF);
        commandWithCRC[commandWithCRC.Length - 1] = (byte)(CRC >> 8);

        //Check if calculation is correct (should equal 0)
        check = CRC16IBMCalculator.ComputeChecksum(commandWithCRC);

        return commandWithCRC;
}

public class CRC16IBM
{
        const ushort polynomial = 0xA001;
        ushort[] table = new ushort[256];

        public ushort ComputeChecksum(byte[] bytes)
        {
                ushort crc = 0; // initial CRC value
                for (int i = 0; i < bytes.Length; ++i)
                {
                        byte index = (byte)(crc ^ bytes[i]);
                        crc = (ushort)((crc >> 8) ^ table[index]);
                }
                return crc;
        }

        public byte[] ComputeChecksumBytes(byte[] bytes)
        {
                ushort crc = ComputeChecksum(bytes);
                return BitConverter.GetBytes(crc);
        }

        public CRC16IBM()
        {
                ushort value;
                ushort temp;
                for(ushort i = 0; i < table.Length; ++i) {
                        value = 0;
                        temp = i;
                        for(byte j = 0; j < 8; ++j) {
                                if(((value ^ temp) & 0x0001) != 0) {
                                        value = (ushort)((value >> 1) ^ polynomial);
                                }else {
                                        value >>= 1;
                                }
                                temp >>= 1;
                        }
                        table[i] = value;
                }
        }
}
```

Errata

In all read commands the CRC code that is sent from the driver is always “00”. This also applies to the temperature read command.