Electrical Lens Driver 4

Lens Driver Manual

Optotune Switzerland AG | Bernstrasse 388 | CH-8953 Dietikon | Switzerland
Phone +41 58 856 3000 | www.optotune.com | info@optotune.com

Table of Contents

1. Overview

The Electrical Lens Driver offers a simple yet precise way to control Optotune's electrical lenses. There are two types of lens drivers available. The Electrical Lens Driver 4 is used to drive Optotune's OEM-type lenses with 6-pin FPC cables and comes in a plastic housing. The second Lens Driver 4i is suited to drive the industrial versions with 6-pin Hirose connectors and is contained in a steel housing. The main features are:

- Current control from -290 to +290 mA with 12 bit precision
- Drive frequencies from 0.25 to 2000 Hz (rectangular, triangular or sinusoidal)
- I2C sensor read-out e.g. for temperature compensation
- USB powered
- Driver software in Windows 7, 8 and 10
- Available with or without housing

Mechanical specifications	Lens Driver 4	Lens Driver 4i	
Dimensions (L x W x H)	77 x 19 x 13	99.05 x 19 x 13.5	mm
Weight	11	41	g
Interface to lens	0.5mm pitch 6 way FPC connector	6-pin Hirose connector	
Interface to PC	USB Type A		

Electrical specifications	Lens Driver 4	Lens Driver 4i	
Maximum output current	From -290 to 290 mA depending on resistance (see Figure 6)		
Maximum output update frequency	>100		kHz
USB input voltage	5		V
Power consumption	50-1100		mW
Digital to analog converter	12 bit Analog Devices ADN8810		
Microcontroller	8-bit, 16 MHz with 32 KB Flash (Atmel ATmega32U4)		
Connector	6-way FPC (Molex 503480-0600)	6-way Hirose HR 10 G	
Maximum cable length	3 meters [1]		

Thermal specifications	Lens Driver 4	Lens Driver 4i	
Operating temperature	-20 to +65		°C
Storage temperature	-40 to +85		°C

1.1 Integration Into OEM Systems

Both Lens Drivers are easily integrated into OEM systems. The microcontroller that is used offers serial interfaces over USB and 5V UART (which can be translated to RS-232 using a MAX232 or MAX3232 chip). Also, analog input from 0-5V is available on the driver PCB. Schematics and part list of the Lens Drivers are available on request. Documentation of the firmware is presented at the end of this document. For more information, please contact sales@optotune.com.

---

[1] 5 m have been successfully tested, however the capacitance is 550 pF (I2C specification 400 pF max).

1.2   System requirements

- Windows 7 or Windows 8.1, 32/64 bit or Windows 10
- Lens driver: USB 2.0 port
- uEye camera: USB 2.0 port (preferably 3.0)

2. Hardware Operation

2.1   Connecting to a lens with FPC cable

The Molex flex cable 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.

⚠️   **Do not hot plug/unplug FPC cable from driver connector as permanent damage may result!**
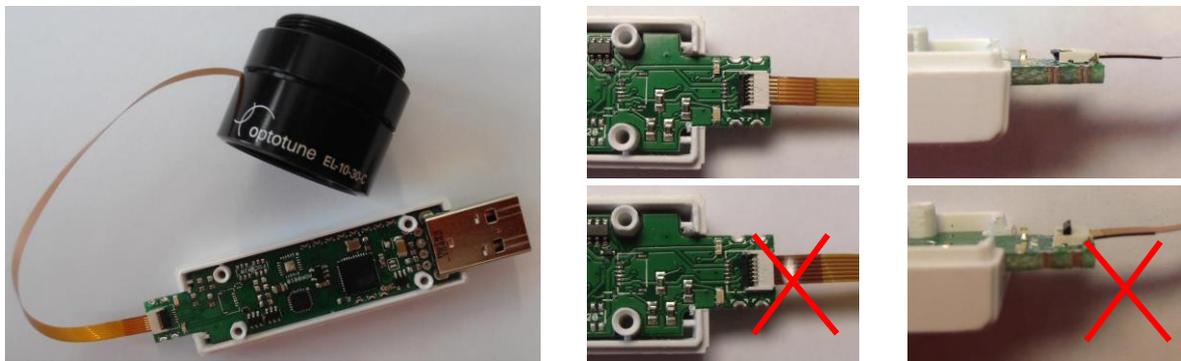

Figure 1: Connecting the EL-10-30-C directly to the lens driver.

## 2.2   Connecting industrial-style lenses to the Lens Driver 4i

The connection of the industrial EL-10-30-Ci to the Lens Driver 4i is straight forward. Simply connect the cable to the plug, the position of the pins is unambiguous.

2.3   Open and Close Housing of Lens Driver 4i

To open the housing use the gaps on the side to spread the cover side-walls with an appropriate flat head screw driver. Hold the enclosure base and lift the cover to open the casing:

In order to close the housing, place the cover on top of the enclosure base (a) and move it towards the USB stick (b) until the front side clips fit in the openings of the cover. Then push the cover at the back downwards (c) until the back side clips snap into the square holes (d). If the clips do not snap in, gently apply pressure from the top/bottom/side to move the cover forward/backward or up/down relative to the base unit.
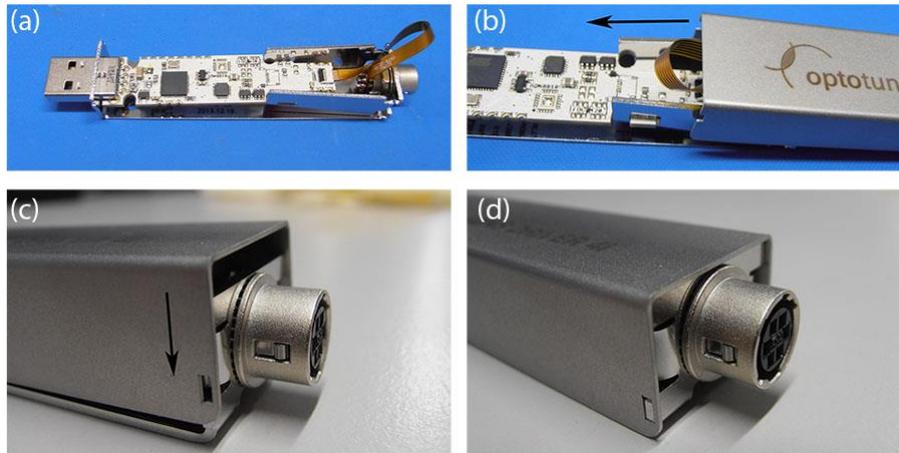
2.4 Connector Pinout
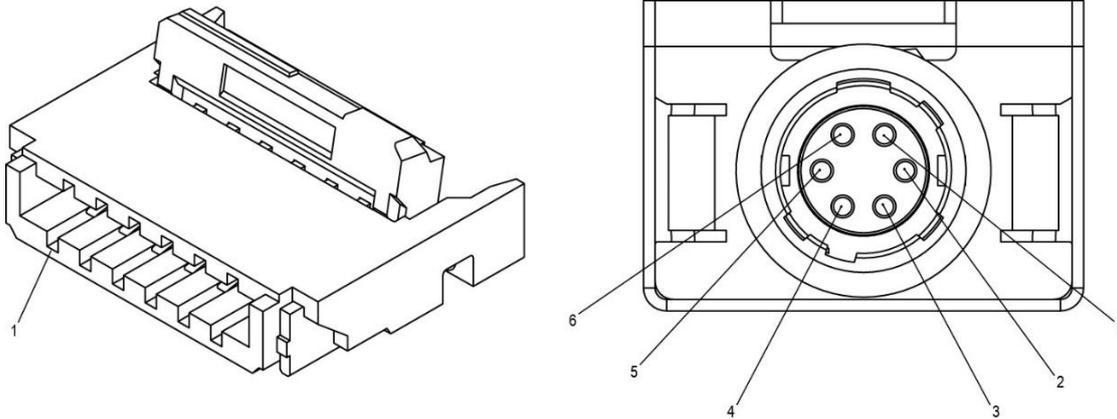
Figure 2: Connector pinout of the Lens Driver 4 (left) and Lens Driver 4i (right).

Lens Driver 4			Lens Driver 4i	
Position	Function		Position	Function
1	I2C Gnd		1	Lens (+ pole)
2	Lens (- pole)		2	Lens (- pole)
3	Lens (+ pole)		3	I2C Gnd
4	I2C SDA		4	I2C Vcc 3.3V
5	I2C SCL		5	I2C SCL
6	I2C Vcc 3.3V		6	I2C SDA

3. Lens Driver Controller Software

3.1   Software Installation

- Download the **Lens Driver 4 Software Package** from Optotune's website:
  https://www.optotune.com/registration-for-software-download
- Run Setup.exe
- Follow the installation wizard

3.2   Installation of the Windows driver in Windows 7, Windows 8.1 and Windows 10

Please follow the instructions below to install the Windows driver for the Optotune Lens Driver

- IMPORTANT: Please disconnect any lenses from the Lens Driver hardware!
- Connect Lens Driver to a free USB port on your PC.
- Windows will now try to install a driver. This process won't succeed because of a missing driver software. This can be ignored.
- Open Windows "Device Manager".
- **Windows 7, 8, 8.1**
  In "Device Manager" navigate to the category "Other devices", right click on the device "Optotune LD" and select "Update Driver Software...".

  Windows 10
  In "Device Manager" navigate to the category "Ports (COM & LPT)", right click on the device "USB Serial Device (COMX)" and select "Update Driver Software...".

- Select "Browse my computer for driver software".
- Click on "Browse" and navigate to the directory where you installed the Lens Driver Controller application (usually *C:\Program Files (x86)\Optotune AG\Lens Driver Controller*). Confirm the folder choice by clicking "OK".
- Make sure "Include subfolders" is checked and click "Next".
- Click "Next" in the "Update Driver Software" window.
- The installation proceeds and installs the Windows driver.
- The Windows driver is successfully installed if you see a device called "Optotune Lens Driver powered by atmel" in the section "Ports (COM & LPT)" within the "Device Manager".

4.   Preparing Lens Driver 4

In order for the Lens Driver 4 to operate correctly, a product specific firmware needs to flashed. Please refer to the manual of the Lens Driver Firmware Flash Utility (included in the **Lens Driver 4 Software Package** available on our website) on how to flash the firmware. The following list gives an overview of the different firmware types.

Compatible Lens	Firmware Type
EL-10-30-C EL-10-30-Ci	A
EL-10-30-TC EL-16-40 series	F
EL-12-30-TC EL-3-10	A or F (focal power mode not supported)

5.   Operating Lens Driver Controller

Launch Optotune Lens Driver Controller and click on **Connect.** This will establish the hardware connection and open the main window with the current control and the temperature readout.

Controls

Figure 3 shows the main window. 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.
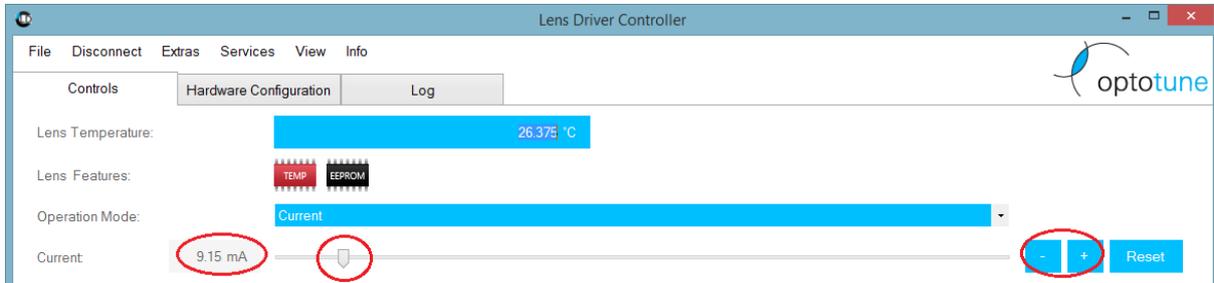
Figure 3: Screenshot of the main window of the Lens Driver software with the current control.

The drive signal can be chosen to be a DC, sinusoidal, rectangular or triangular[2] signal with the possibility to set the upper and the lower signal level as well as the driving frequency. This is indicated in Figure 4.
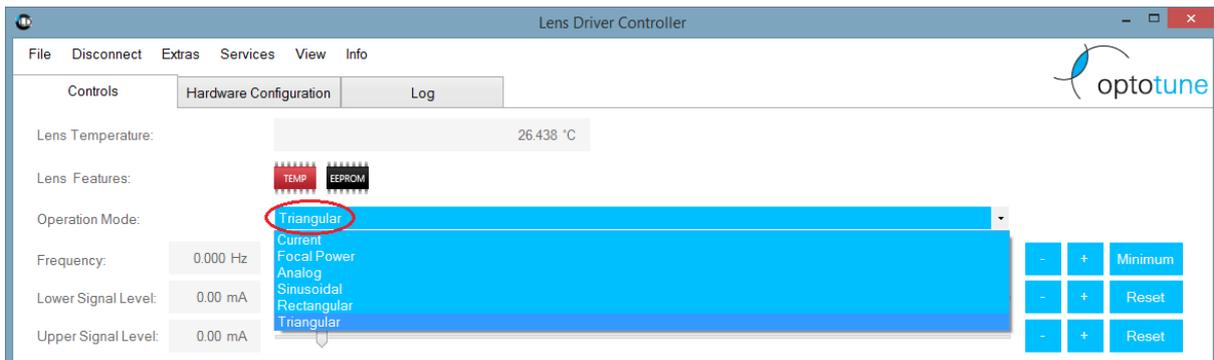
Figure 4: The drive signal can be chosen to be a DC, sinusoidal, rectangular or triangular[2] signal.

5.1   Limiting the Maximum Current

The limits for the current are set in the **Hardware** tab, shown in Figure 5.
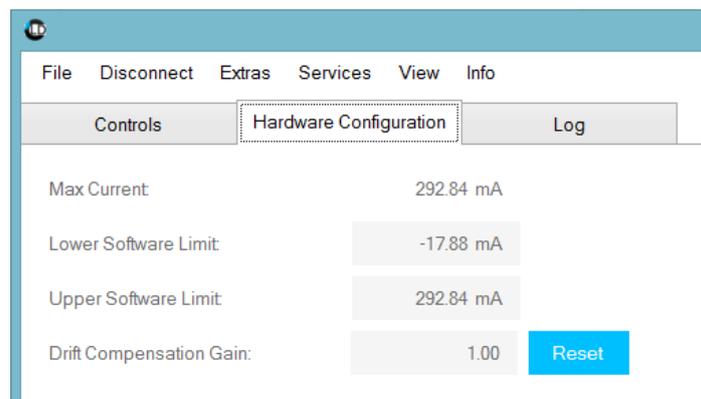
Figure 5: Setting the current limits.

[2] FW Type A only

The maximum current of the driver is limited to either 290mA, see Figure 6, or 2.8V divided by the resistance applied.
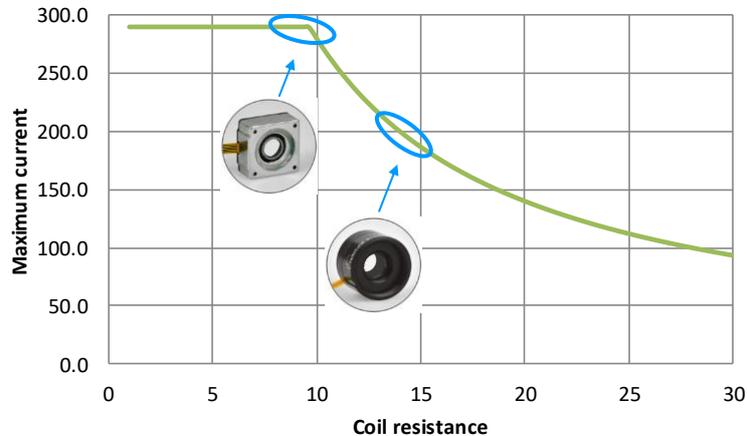

Figure 6: Dependence of maximum current on coil resistance.

5.2   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 0.2 to 0.7 diopters per 10°C temperature increase, depending on the lens model. This temperature effect is systematic and reproducible and is therefore accurately compensated via a temperature sensor STTS2004 with an I^2C sensor read-out, integrated in the lens (except in the EL-3-10). This allows controlling the focal power directly.

In **Focal Power mode**, see drop down menu in Figure 4, temperature independent lens operation is ensured. Depending on the present temperature, the current applied to the lens is adjusted for compensation of the temperature drift. A look-up table with the calibration data for the temperature compensation are stored directly on the EEPROM of each individual lens. With the temperature compensation enabled, the absolute reproducibility achieved over an operating temperature range of 10 to 50°C amounts to typically 0.05 or 0.1 diopters for the EL-16-40-TC and EL-10-30-TC series, respectively.

5.3   Achievable Range of Focal Power

If the focal power is outside the guaranteed range, a warning message will appear in yellow "warning focal power outside guaranteed range" and the panel has a yellow background, see Figure 7. The user can then change the focal power until the displayed focal power no longer has a yellow color and is within the guaranteed range.
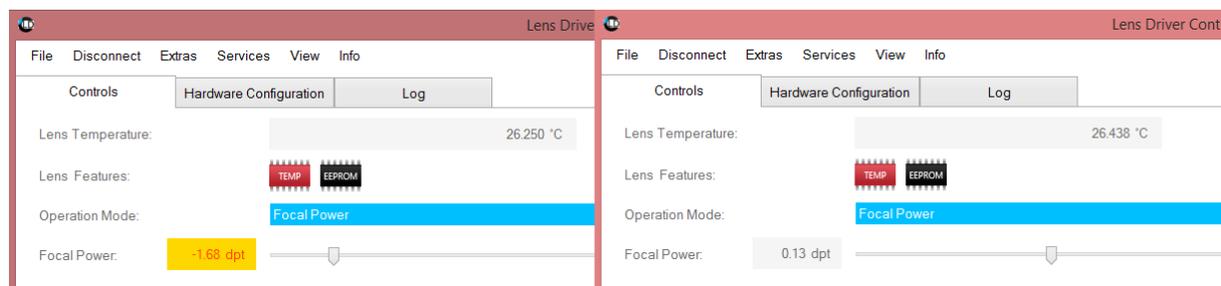

Figure 7: On the left, the focal power is outside the guaranteed range and the panel is yellow. On the right, the focal power is within the range.

The limits for the different ranges of focal power are determined by the temperature limits and the maximum and minimum current. The maximum focal power value limit is located at highest encoded temperature limit

and maximum current and the minimum focal power limit is located at the lowest encoded temperature and the minimum current. This is explained in Figure 8. The highest and lowest temperature define a sector for the linear relation of focal power versus current, indicated by the light and dark gray line. The values are adjustable in **File → Options** under **Temperature Settings**. The colored line represents the focal power range at the current temperature between the highest and lowest temperature, with the colors representing the following states:

- **Green:** Focal power can successfully be maintained at predefined temperatures limits in this range.
- **Yellow:** Focal power can be maintained at the current temperature and/or current drift state.
- **Red:** Focal power cannot be maintained at the current temperature and/or drift state.

The minimum green/yellow limit is determined by the focal power value at the highest temperature and minimum current. The maximum green/yellow limit is determined by the focal power value at the lowest temperature and maximum current. Both limits are indicated by the dashed horizontal lines in Figure 8. The closer together the minimum and maximum temperature limits are, the larger the guaranteed range (green) will be. In addition, the yellow/red limits are determined by the focal power values at min/max current at the current temperature.
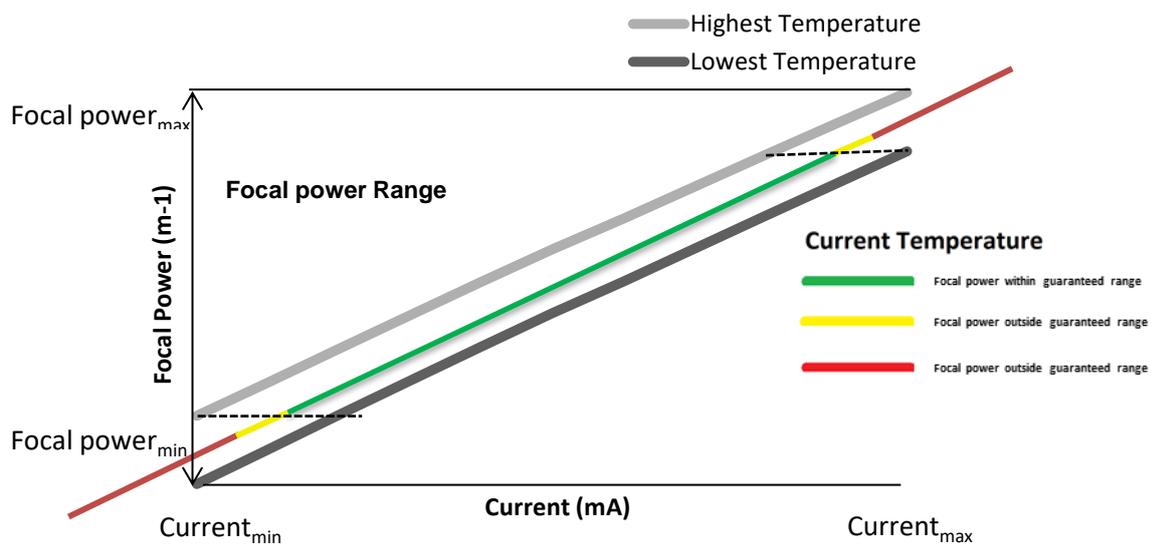

Figure 8: Illustration of the achievable range of focal power (vertical axis) versus current (horizontal axis). The guaranteed range (green line) also depends on the highest and lowest temperature, indicated by the gray lines.

5.4   Drift Compensation

When set to a certain focal power the lens shows a tiny drift over timescales of about 5min. If a highly stable focus position over long times is required, using the implemented drift compensation is recommended. To do so the *Drift Compensation Gain*, found under **Hardware** (see Figure 5) has to be set by the user to a finite value, typically 1, otherwise no compensation is taking place. Higher values increase the gain up to a maximum value of 5. Internally, two parameters $Drift_{max}$ and $Drift_{current}$ are affected by the *Drift Compensation Gain*, as discussed in the following.

When finite drift compensation is set it also scales the achievable focal power range. Figure 9 illustrates the effect that drift compensation will have on the achievable focal power range. First, compared to Figure 8, the yellow/red boundaries in Figure 9 are shifted upwards due to a finite current drift ($Drift_{current}$). As a consequence, the achievable focal power range is also shifted upwards by a small amount. Second, the maximum drift $Drift_{max}$ set by the drift compensation is added to the minimum guaranteed focal power limit, which is represented by the lower yellow/green boundary, and therefore slightly reduces the guaranteed focal power range. The maximum drift is determined by the gain variable and a larger gain variable will result in a larger maximum drift.

For lenses with a small focal power range the effect of increasing the gain variable to high values (e.g. 3 or higher) can be that the achievable focal power range is too small such that focal power cannot anymore be changed by the user. In that case the gain variable must be reduced.

A visual representation of the overall achievable focal power range as shown on the focal power scroll bar in the software can be found in Figure 10.
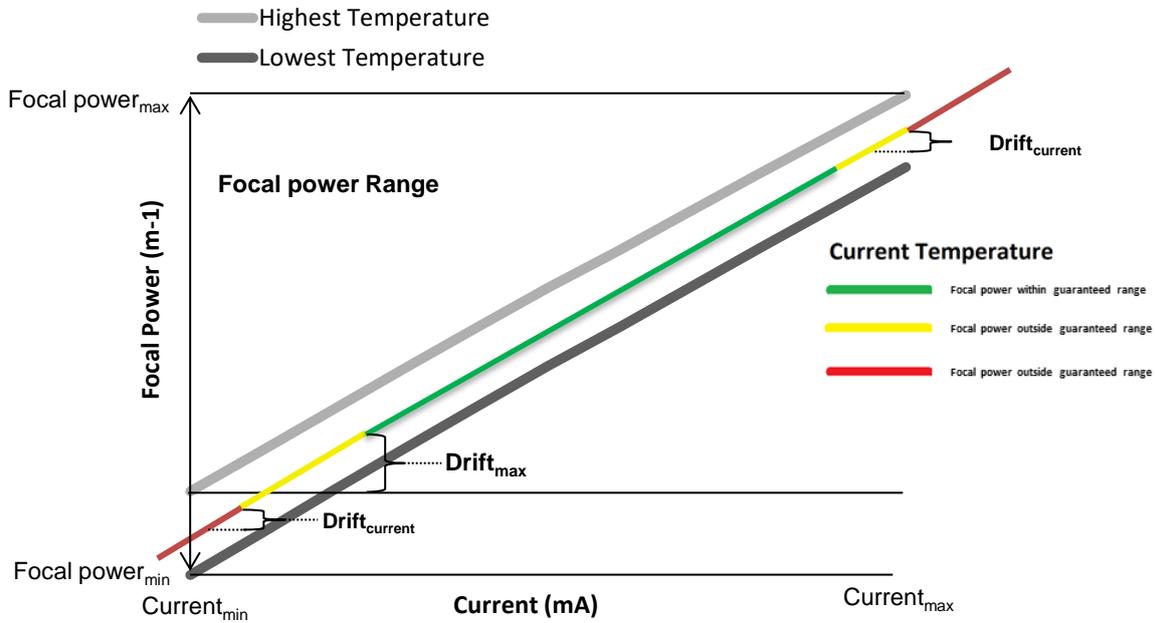

Figure 9: Illustration how drift compensation affects the achievable range of focal power.
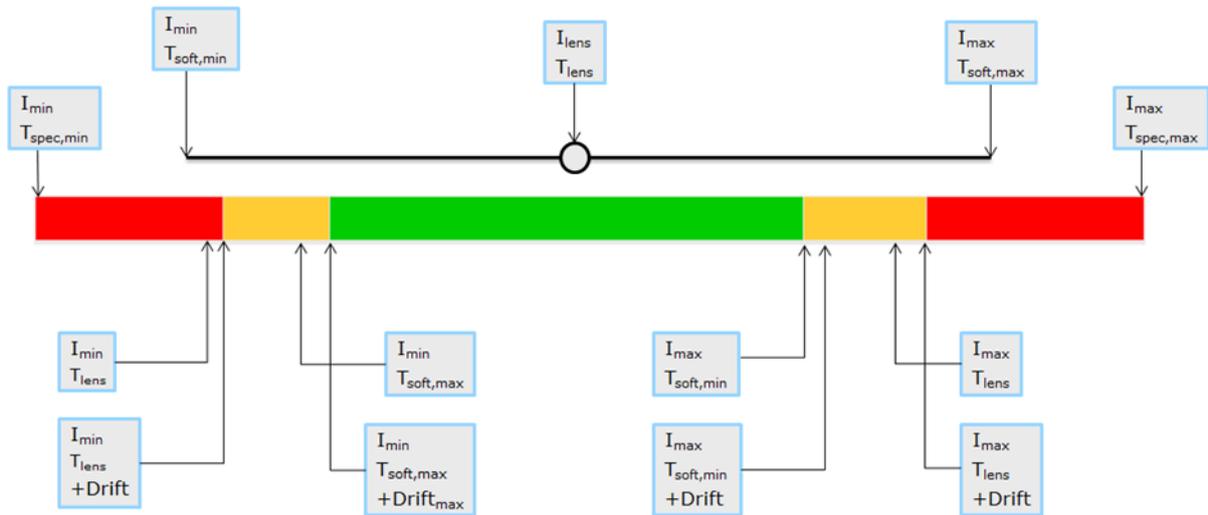

Figure 10: Overall achievable focal power range as shown on the focal power scroll bar.

5.5 Trigger Output Signal

For sinusoidal, triangular and rectangular drive signals, hardware pin A (see Figure 11) outputs a trigger signal which can be used to synchronize the lens driver to another hardware component. The synchronization output signal toggles between 0 V and 5 V and provides a maximum of 15 mA. Please note that the pin "*Analog In A*" has been re-configured from an analog input to a digital output (trigger output).
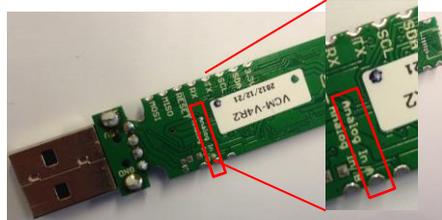

Figure 11: Hardware pin A for trigger output signal.

The relation between the trigger output signal and the corresponding output signal of the Lens Driver for different drive signals are shown in Figure 12:
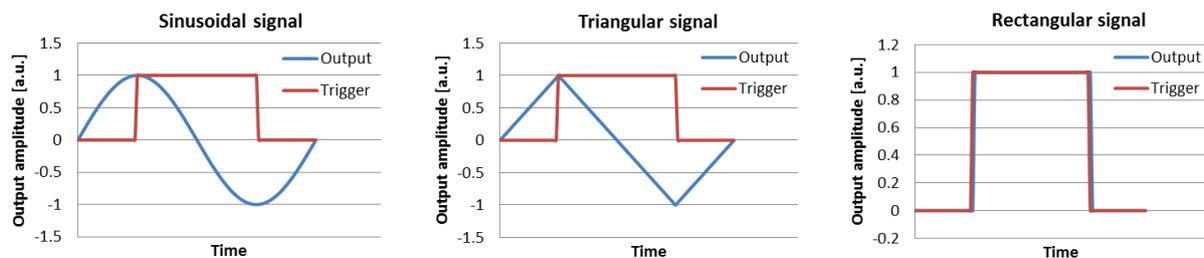
Figure 12: Relation between trigger and output signal for different drive signals.

5.6 Lens Control by Analog Signal

The lens current can be controlled by the analog signal reading feature, which is built into the firmware. The analog voltage applied on hardware pin B (see Figure 13) must be between 0 V and 5 V. The ADC has a resolution of 10 bit and therefore the digital signal value lies between 0 and 1023. The applied voltage and therefore the 10 bit value is then linearly mapped in firmware to the range defined by the lower and upper software limits in the **Hardware Configuration** (see Figure 14) tab of **Lens Driver Controller**. These limits are persisted in the firmware configuration and are reloaded when the lens driver is powered up (Lens Driver Controller is not required to run for this). In analog mode (see Figure 15), the Lens Driver autonomously and continuously maps the analog value to the corresponding current value applied to the lens.

Note that in this stand-alone analog mode the analog value is only mapped to current and thus no automatic temperature compensation is possible. The next section explains how the analog value can be mapped to optical power by going through the Lens Controller Software.
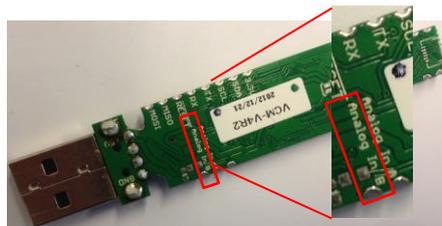
Figure 13: Hardware pin B for analog input signal.
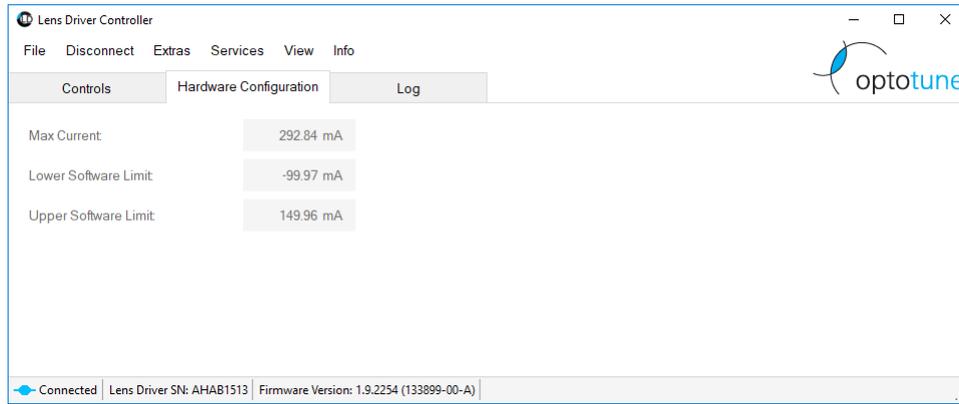
Figure 14: Hardware Configuration (example shows mapping between -100mA and 150mA)
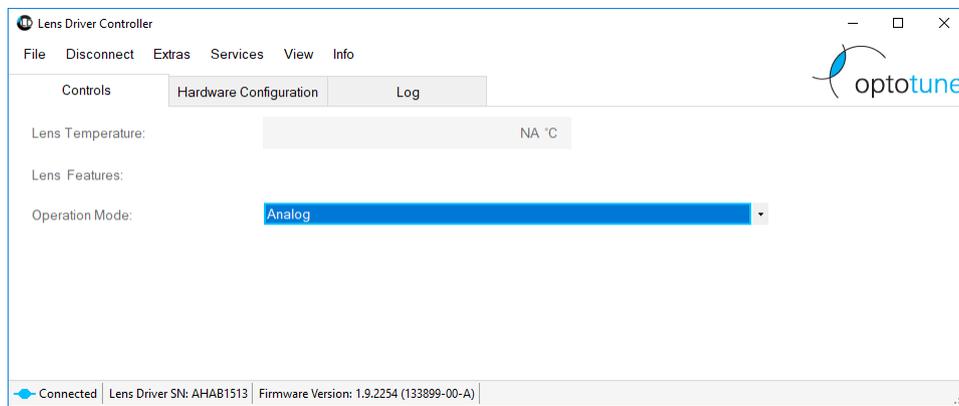
Figure 15: Analog mode enabled in Lens Driver Controller

5.7   Sensor Control

Through an analog signal that is read by the lens driver, an external voltage from an analog sensor can be used to control the focal power of the lens. Sensor control can be used in either **Current** or **Focal Power** mode, and is enabled under **Services -> Sensor Control**.

To enable this functionality, a set of calibration points must first be inputted. A calibration point consists of an analog input value set by a sensor and a desired current/focal power value. If sensor control is enabled, the software will linearly interpolate between the configured points, and set the current/focal power according to the analog input. It is important to note that sensor control will only be enabled if at least two calibration points have been defined. Calibration points can be stored using the sensor control configuration form found under **Extras -> Sensor Control Configuration**. The points are stored in a modifiable table, and are saved in the software for future use. The screenshot below represents the sensor configuration form in **Current** mode.
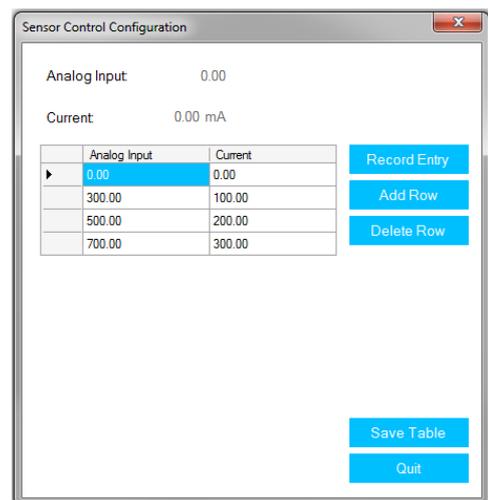
Button Functions:

- **Record Entry:** Store the latest analog input and current/focal power value as a calibration point.
- **Add Row**: Create an empty row in the configuration table to manually input a calibration point.
- **Delete Row:** Every row with a highlighted cell will be removed from the table.

- **Save Table:** Save the configuration table. Note that sensor control will only work if at least two calibration points have been saved.
- **Quit:** Exit the sensor control configuration form. Users will be prompted to save any unsaved data.

6. Communication Protocol

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.

While this section describes the protocol for the main commands used to control the lens driver, detailed information on other commands can be found in the command protocol file which is available on request**.**

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
- Data Bits: 8

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

6.1 Current Set Commands

A current set command sets a new output current for a channel. A current set command is constructed as AwxxLH. It always starts with the channel identifier "A" for channel A[3], and the write identifier char "w". 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, $i_c = 293$mA. The formula to calculate the value x_i for a certain current is:

$x_i = i_o / i_c * 4096$

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

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 'Low Byte', 'High 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 cali-

---

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

bration 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
"w"	Write Identifier

PC sends	Driver answer	Driver action	Comment
"AwxxLH"	none	set channel A current to xx	xx is a signed 16bit integer with a value between -4096 and 4096, high byte sent first
"AwxxLf"	"E1"<crc>"\r\n"	none	Here we simulate an error, passing a wrong CRC value. f = error in CRC byte, this line applies for all detected CRC errors. For all the error codes refer to the structure protocol.

Exemplary current set command

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

> **Resulting command**:
> Byte 0: 0x41 (corresponds to ASCII "A")
> Byte 1: 0x77 (corresponds to ASCII "w")
> Byte 2: 0x04 (high current byte) / 0b00000100
> Byte 3: 0xb2 (low current byte) / 0b10110010 -> with byte 2 the total current bits are 0b0000010010110010 which equals to 1202 dec
> Byte 4: 0x26 (low CRC byte)
> Byte 5: 0x93 (high CRC byte)

For verification of the command, the CRC checksum (see details below) over the whole 6 bytes of the command can be calculated which should be zero.

6.2 Focal Power Set Command

A focal power set command sets the focal power of the lens. The command starts with the prefix "P", followed by the write identifier "w", the coding char "D", and the channel identifier "A". The command then includes 2 bytes for the desired focal power value and 2 dummy bytes. The focal power value has a firmware type specific encoding.

Firmware type A (EL-10-30):

$x_i = (fp + 5) * 200$

Firmware type F (EL-16-40):

$x_i = fp * 200$

Example for setting a focal power of 5 diopters in type A firmware:
(5 + 5) * 200 = 2000

Similar to current set commands, the command is finished by a 16bit CRC calculated from the eight command bytes. The low byte of the CRC is sent first. It is important to note that this command will only work in controlled mode (see Mode Commands below).

Char coding for the focal power set command:

"P"	Prefix
"w"	Write Identifier
"D"	Coding
"A"	Channel A

PC sends	Driver answer	Driver action	Comment
"PwDAxxYYLH"	none	set focal power of lens through channel A	xx is a signed 16bit integer, high byte sent first. YY are 16 dummy bits. For the usable limits of the signed 16bit integer please refer to reply command that is sent when setting Focal Power Mode
" PwDAxxYYLf "	"E1"<crc>"\r\n"	none	Here we simulate an error, passing a wrong CRC value. f = error in CRC byte, this line applies for all detected CRC errors. For all the error codes refer to the structure protocol.

Exemplary focal power set command

    Command type: Set focal power
    Channel: A
    Focal power value: 2000 (5 diopters)

    Resulting command:
    Byte 0: 0x50 (corresponds to ASCII "P")
    Byte 1: 0x77 (corresponds to ASCII "w")
    Byte 2: 0x44 (corresponds to ASCII "D")
    Byte 3: 0x41 (corresponds to ASCII "A")
    Byte 4: 0x07 (high focal power byte) / 0b00000111
    Byte 5: 0xd0 (low focal power byte) / 0b11010000 -> with byte 1 the total current bits are 0b 0000011111010000 which equals to 2000 dec
    Byte 6: 0x00 (required dummy byte)
    Byte 7: 0x00 (required dummy byte)
    Byte 8: 0x31 (low CRC byte)
    Byte 9: 0xfd (high CRC byte)

6.3   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 mode type ("S", "Q", "T","C" 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.

The mode "C", or Controlled Mode, allows the driver to maintain the focal power of a connected lens. This mode must be enabled in order to use focal power set commands.

Char coding for the mode change command:

"A"	Channel A
"w"	Write Identifier
"S"	Sinusoidal signal
"Q"	Square signal
"D"	DC signal
"T"	Triangular signal
"C"	Controlled Mode

Char coding for the signal property change command:

"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
"MwCALH"	"MCAsBBCCLH\r\n"	set channel A to focal power controlled mode. The byte s is a status byte #1 (refer to "Communication Protocol" Excel file), BB is a 16bit signed integer max focal power, and CC is a 16bit signed integer min focal power. The focal power values **fp** can be calculated from the sent integers x_i by using the following formula (depending on firmware type): **Firmware type A (EL-10-30):** $fp = x_i/200 - 5$ **Firmware type F (EL-16-40):** $fp = x_i/200$
„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)

Setting focal power mode sends a command back containing the usable limits for the focal power values.

6.4   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

6.5 Temperature Reading

If the lens connected supports temperature reading, this command will read back the temperature of the integrated sensor (STTS2004). 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
"TCALH"	"TCAddLH\r\n"	Get temperature reading of lens on channel A	"dd" is the temperature data as 16 bit signed integer.

6.6 CRC Algorithm

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

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

6.7 Checking for Communication Error

Checking for a communication error is done by calculating the CRC checksum over the whole command data array which includes two CRC bytes at the end that were added from the sender. CRC checksum calculation over the whole array results in a CRC checksum equal to zero if no data corruption is present.

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

```
/* Example usage of the CRC function */
// Example:    Set current level to 1202 command sent from PC (includes CRC bytes)
//             With this data array the returned uint16_t crc checksum will be zero
//             (0x00) since no communication error has occurred.
//             uint8_t data[] = { 0x41, 0x77, 0x04, 0xb2, 0x26, 0x93 };

uint16_t calculatecrc(void)
{
        uint16_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
    }


    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;
    }
```

6.8   Adding a CRC Checksum to a Data Array

In order to add a CRC checksum to a data array

1. Calculate CRC checksum over data array
2. Append the calculated CRC checksum at the end of the data array

Example:
```
uint8_t dataarray[3] = {data byte 0, data byte 1, data byte 2}
uint8_t dataarraywithcrc[5] = { data byte 0, data byte 1, data byte 2, crc&0xFF,
crc>>8}
```

If this is done correctly, a consecutive CRC checksum calculation over the whole array needs to result in a CRC checksum equal to zero (see method *AddAndCheckCRC*).

The following example written in c# shows the implementation in the Lens Driver Controller software:

```
    private byte[] AddAndCheckCRC(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]);
```

Optotune Switzerland AG | Bernstrasse 388 | CH-8953 Dietikon | Switzerland
Phone +41 58 856 3000 | www.optotune.com | info@optotune.com

```
                    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;
        }
}
}
```

7. UART Communication (only Applicable for EL-10-30-C Firmware Type A)

The Lens Driver V4 consists of an ATMEGA 32U4 Atmel chip that can utilize an UART interface for serial communication. Communication can be established by connecting to the RX and TX pins on the microcontroller. These pins are also accessible at the edge of the Lens Driver V4 for easy connection.
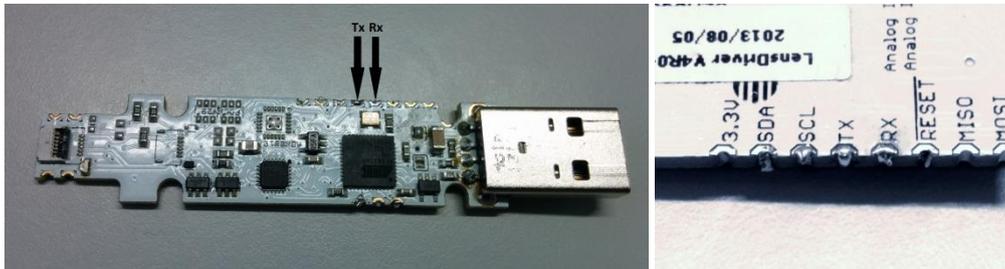

Figure 16: The above image shows the location of the Rx and Tx pins on the Lens Driver V4. The name of each of the pins is also written on PCB of the Lens Driver V4, as shown in the diagram on the right.

The Lens Driver V4 operates at 5V. The serial communication in the Lens Driver V4 is TTL serial (transistor-transistor logic) and therefore will always remain between 0V and Vcc (5V). The standard RS232 communication differs from the TTL serial at a hardware level. By RS232 a logic HIGH is represented by a negative voltage (anything between -3V to -25V) while a logic LOW is represented by a positive voltage (anything between +3 to+ 25).

In order to communicate with the Lens Driver V4 using RS232, the signals must be inverted and the voltages need to be regulated to a voltage that is not harmful for the microcontroller. A solution to this problem can be to use a MAX-232 between the two devices, which solves the problem of inverting and converting the voltage. It is important to also be aware of ground loops that may occur from different power supplies.

The same commands as specified for the USB connection can be used for the UART interface.

The UART port communication port specification is as follows:

Baud rate: 38400

Data bits: 8

Stop bits: 1

Parity: none