Motorized Beam Expander

User Manual

2021 version v.2.5

Altechna

Contents

Motorized Beam Expander (MBE)

January 2021

Product warranty or service will not be extended if:

1. The product is repaired, modified or altered. Unless such repair, modification of alteration is authorized in writing by Altechna.
2. The LOT number of the product is defaced or missing.

Altechna provides this manual “as is” without warranty of any kind, either express or implied, including but not limited to the implied warranties or conditions of merchantability for a particular purpose. In no event shall Altechna, its directors, employees or agents be liable for any indirect, special incidental, or consequential damages (including damages for loss of profits, loss of business, loss of use or data, interruption of busines and the like), even if Altechna has been advised of the possibility of such damages arising from any defect or error in this manual or product.

Specifications and information contained in this manual are furnished for informational use only and are subject to change at any time without notice and should not be construed as a commitment by Altechna. Altechna assumes no responsibility or liability for any errors or inaccuracies that may appear in this manual, including the products and software described in it.

Products and corporate names appearing in this manual may or may not be registered trademarks or copyrights of their respective companies and are used only for identification or explanation and to the owner’s benefit, without intent to infringe.

1. Introduction

This user manual is designed to help to install and operate MBE. Before installing and operating MBE, please read installation and operation instructions carefully. If there are any questions about contents of this manual, please contact sales@altechna.com. Altechna reserves the right to update contents of this manual without any notification.

Disclaimer:

MBE is designed as beam expander (not reducer), thus its operation is limited to beam expansion. In case MBE is used as beam reducer (i. e. in reverse direction), Altechna bears no responsibility for any consequential damage on MBE itself or other components used in the optical scheme.

2. Application

- Quick laser beam parameter (diameter/divergence) change
- Autocollimation
- Combining several laser processes into one manufacturing stage

3. Technical Information

3.1. Main features

- Diffraction limited optical design
- Suitable for usage with ultrafast picosecond and femtosecond lasers
- No internal foci
- Uploaded magnification presets for quick set-up
- Magnification/divergence fine tuning

3.2. Specifications

Continuously variable magnification	1x-5.5x; 1x-2x
Wavelengths	1030 & 1064 nm 515 & 532 nm 343 & 355 nm
Control interfaces	USB 1.1/2.0/3.0 “Virtual serial (COM) port”, RS232 or Ethernet
Software platform	Control software works in Windows OS: XP/2003/Vista/7/8/10. Linux and MAC users can control device with low level commands (drivers are not necessary), sent through USB virtual serial (COM) port or RS232 or Ethernet
Input aperture	13 mm
Exit aperture	48 mm (1x-5.5x model); 19 mm (1x-2x model)
Overall transmission	>97.5%
Pointing stability	<0.1 mrad
Expansion repeatability	<1%

Table 1. MBE specifications.

MBE is designed to comply with diffraction limited condition, meaning that if laser beam had good optical quality before MBE (e.g. OPD <λ/4) after propagation through MBE laser beam experiences negligible OPD degradation, and thus can be focused into minimum focal spot.

Magnification / Wavelength	x1	x1.5	x2
1030-1064 nm	Ø9.5 mm	Ø8.0 mm	Ø6.5 mm
515-532 nm	Ø8.0 mm	Ø6.5 mm	Ø5.0 mm
343-355 nm	Ø7.5 mm	Ø5.5 mm	Ø4.5 mm

Table 2. Max input beam diameter (at $1/e^2$) ensuring diffraction limited performance for MBE 1x-2x models.

Magnification / Wavelength	x1(x1.1)	x2	x3	x4	x5.5
1030-1064 nm	Ø6.0 mm	Ø9.0 mm	Ø8.0 mm	Ø7.0 mm	Ø5.5 mm
515-532 nm	Ø5.0 mm	Ø7.0 mm	Ø6.5 mm	Ø5.5 mm	Ø4.5 mm
343-355 nm	Ø5.0 mm	Ø6.0 mm	Ø5.5 mm	Ø4.5 mm	Ø4.0 mm

Table 3. Max input beam diameter (at $1/e^2$) ensuring diffraction limited performance for MBE 1x(1.1x)-5.5x models. Magnification range for 1030-1064 nm and 515-532 nm wavelengths are limited to 1.1x-5.5x.

3.3. Operation principle

MBE incorporates a 5-lens system with 2 moving groups of lenses. The lenses can move independently of each other, for maximum system flexibility.

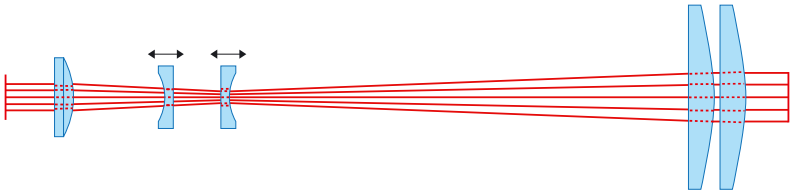
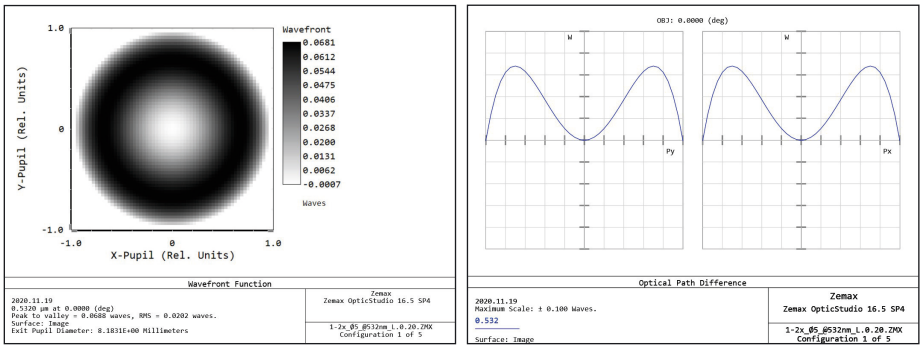


Figure 1. The MBE optics layout.

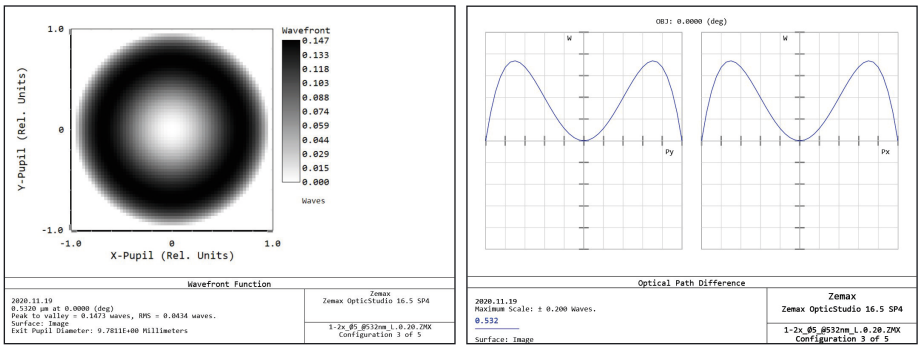
3.4. Optical performance

MBE is designed to operate under diffraction limited regimes, which means the optical path difference (OPD) introduced to the beam will not be higher than λ/4 (PV) and λ/14 (RMS) in a real system. In the graphs below, the nominal OPDs are provided to illustrate the performance of MBE at the maximum recommended apertures before taking any manufacturing errors into account.

1.0



x1.5



x2.0

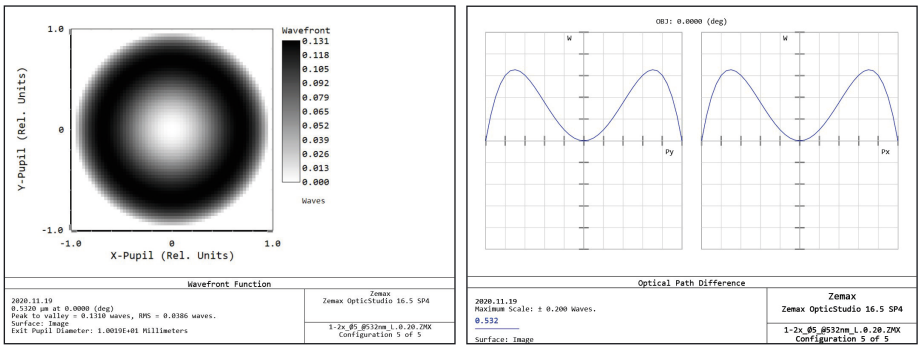
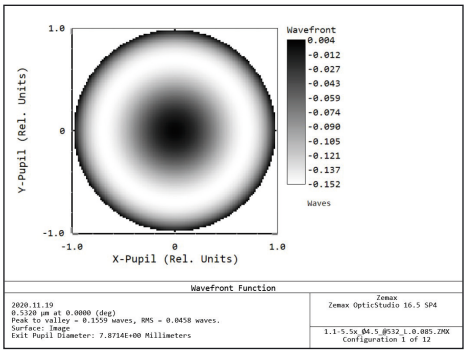
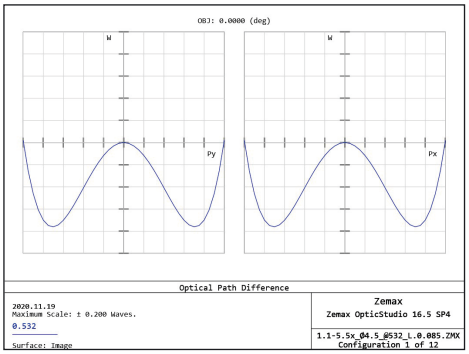


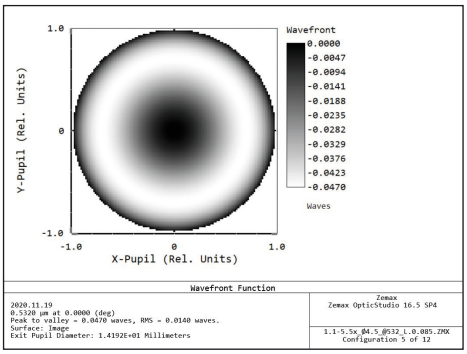
Figure 2. The nominal OPD graphs of MBE 1x-2x @ 532 nm at magnifications x1.0; x1.5 and x2.0.

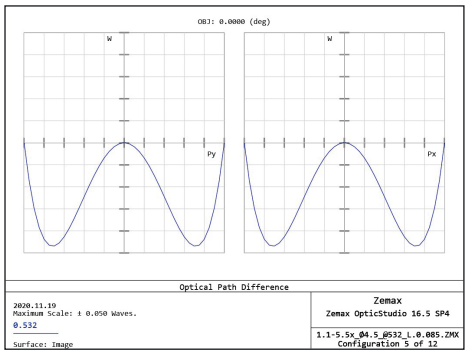
x1.1



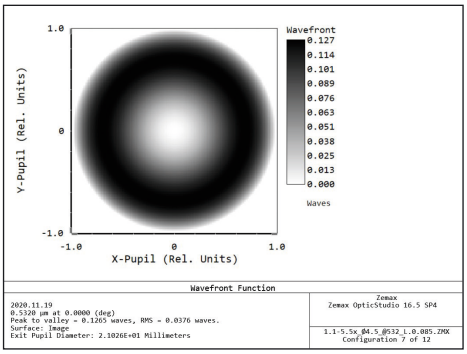


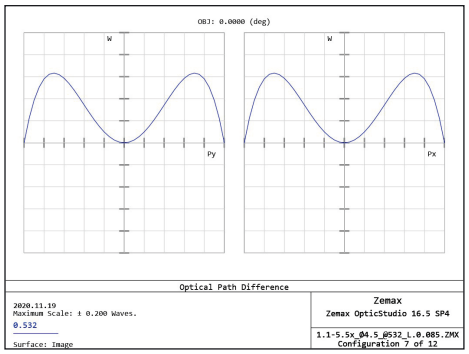
x2.0



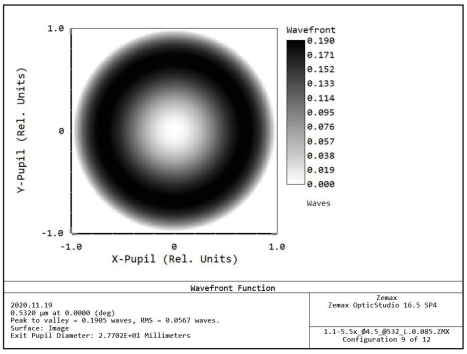


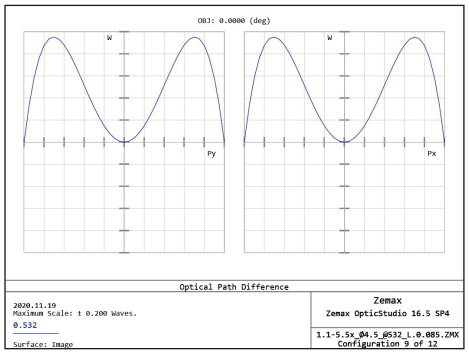
x3.0



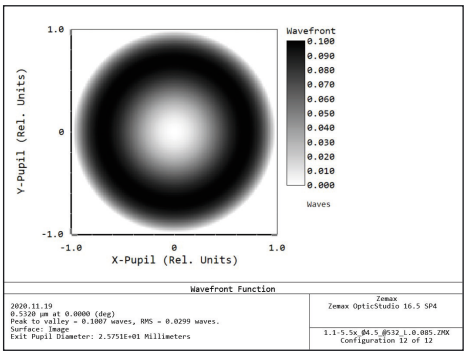


x4.0





x5.5



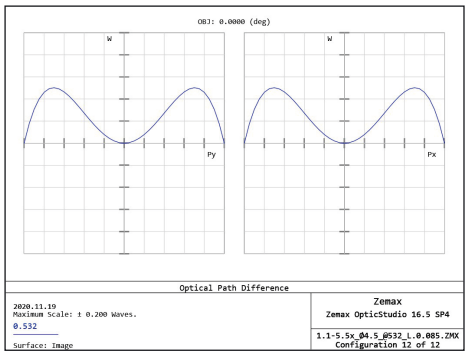


Figure 3. Nominal OPD graphs of MBE 1x-5.5x @ 532 nm at magnifications x1.1; x2.0; x3.0; x4.0 and x5.5.

NOTE: The provided OPD values are nominal, meaning that manufacturing errors are not considered; thus, the OPD values in the graphs are considerably below the diffraction limit requirements. In real systems the OPD values may be higher; nevertheless, real systems are designed to meet the diffraction limited criteria for the maximum recommended input beam parameters.

4. MBE Usage with Windows PC

4.1. Computer requirements

- Free USB port or RS232 port. MBE is compatible with USB 1.1, USB 2.0 and USB 3.0 Computer administrator rights (only for installation)
- Operating systems supported: Windows XP sp3, Windows Server 2003 sp2, Windows Vista sp1, Windows Server 2008, Windows 7, Windows 8 and Windows 10
- Microsoft.Net framework 4.0 redistributable (installs automatically)
- Microsoft Visual C++ Redistributable 2019 or later (installs automatically)
- Adobe Acrobat Reader 7.0 or above

4.2. “AltOne” software installation

“AltOne” software is a dedicated software for all Altechna products. It has been designed to automatically detect different Altechna products by the product’s serial number.

The software can operate only when connected to a “USB” or “RS232” port. When communicating through an “Ethernet” port, you should establish the connection yourself by taking your system’s configuration into account.

Software can be downloaded from the product page:
https://www.altechna.com/products/mbe-motorized-beam-expander/

Installation:

1. Run the “AltOne-Setup.exe” installation file. In a case where the software is being installed on an operating system that does not meet the requirements, only the USB drivers will be installed. Click “Yes” to continue.
2. An installation window will appear, click “Next” to continue:

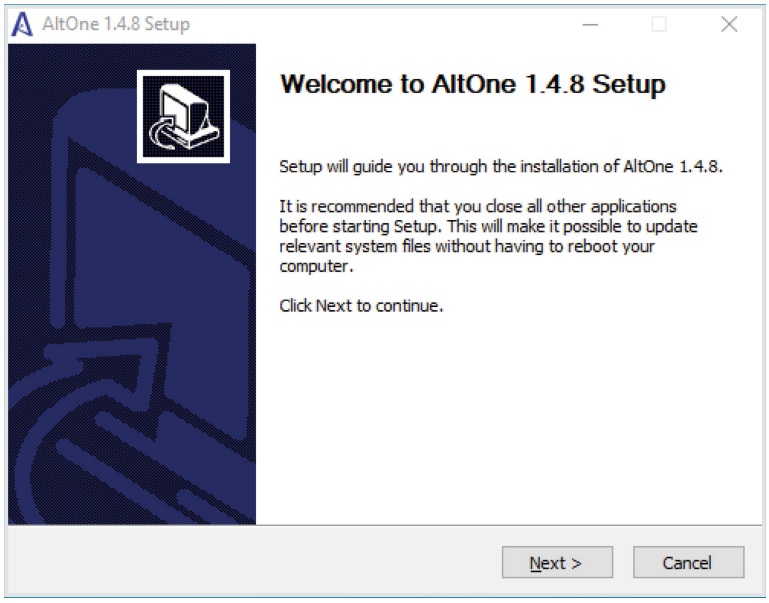


Figure 4. “AltOne” installation.

3. Click “Next”:

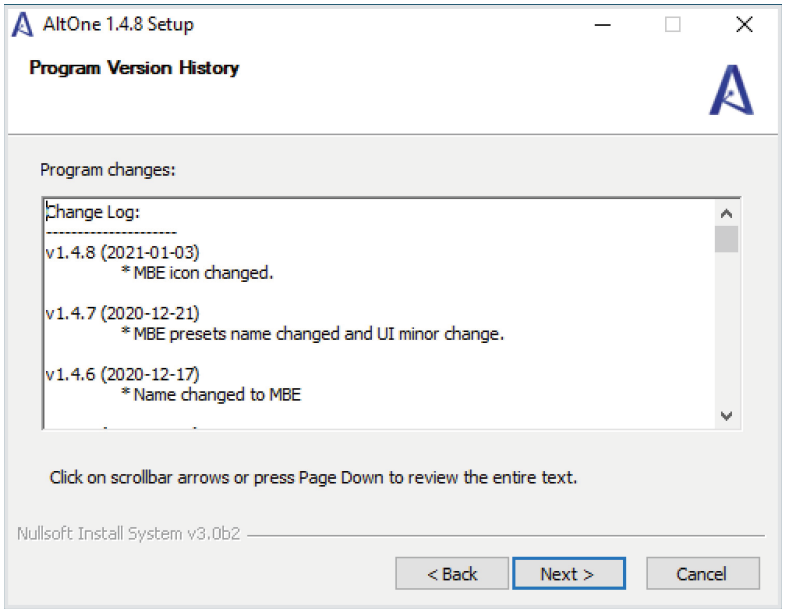


Figure 5. “AltOne” installation.

4. Select installation directory and click “Next” to begin installation:

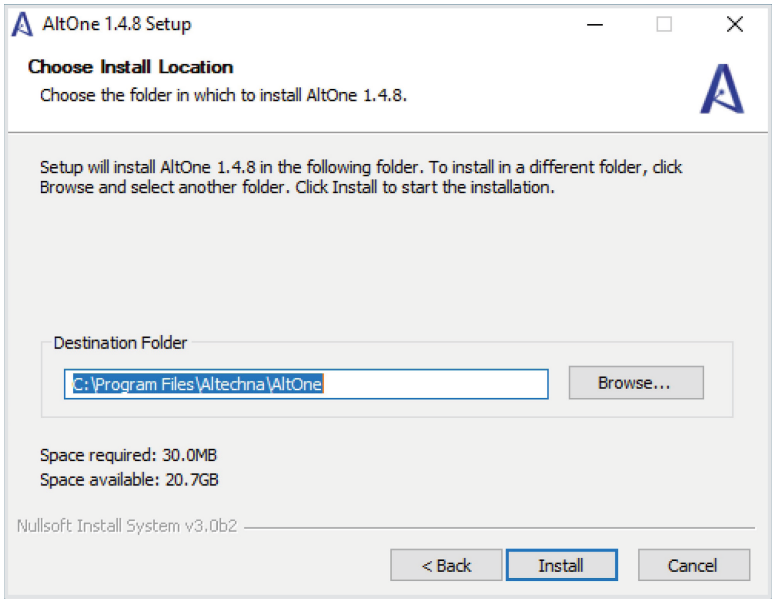


Figure 6. “AltOne” installation.

5. “AltOne” software requires “Microsoft.NET Framework 4 Client Profile” or a higher version to be installed. Setup will offer to download it, so choose “Yes” if an active internet connection is available. Choose “No” if you prefer to download the framework from www.microsoft.com/en-us/download/details.aspx?id=17113 and install it manually. You should also choose “No” if a newer version of Microsoft.NET framework is unwanted, or a working internet connection is not available. This dialog will not appear if the framework is already installed. The download size is 42 MB.
6. Setup will download “Microsoft .NET Framework 4 Client Profile”.
7. After the download is finished, “Microsoft .NET Framework 4 Client Profile” will be installed, so wait for the installation to be complete. This can take more than 10 minutes on a slower machine.

8. Setup will finish by installing the drivers. Click “Next” to continue.

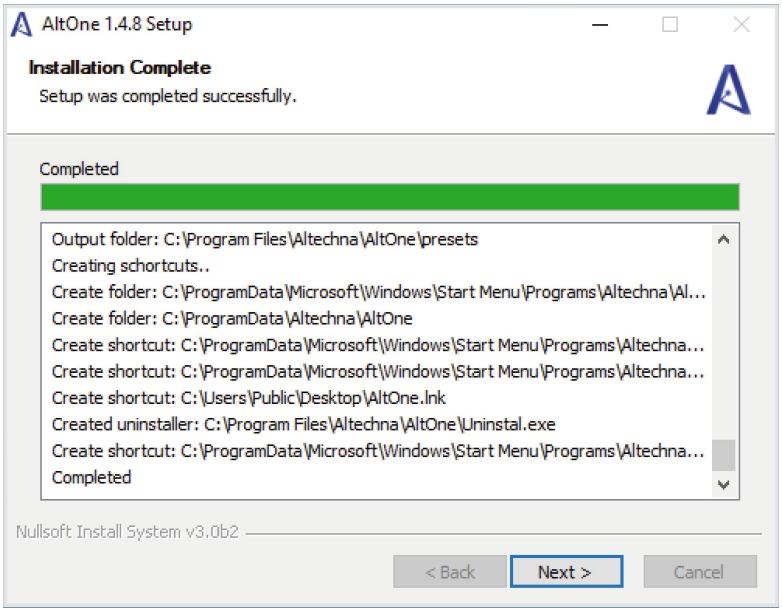


Figure 7. “AltOne” installation.

9. Click “Finish” to end the installation. The program cannot be opened if only the drivers were installed.

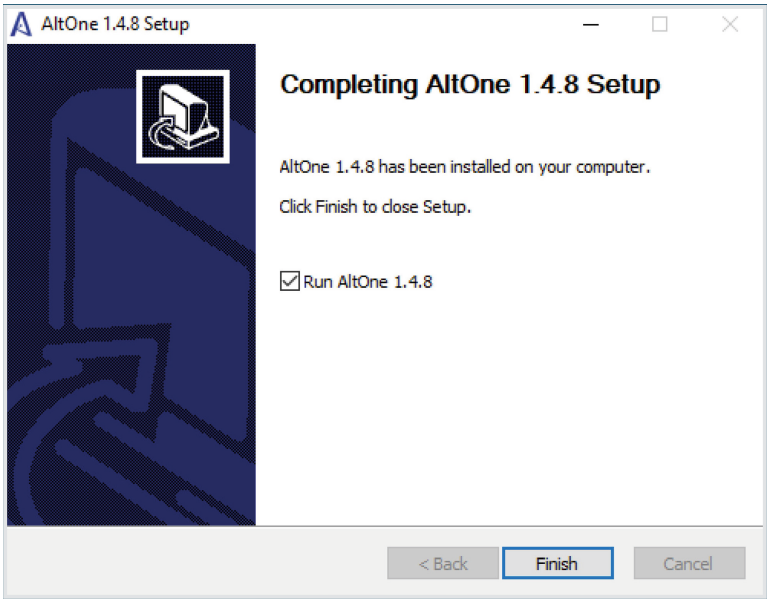


Figure 8. “AltOne” installation.

10. The “AltOne” software icon will appear on the “All Users” desktop and “All Users” start menu.

When using MBE with USB:

1. Connect MBE to its controller.
2. Connect MBE controller and PC via USB cable.
3. Plug in MBE power supply jack.
4. Windows will detect new hardware. Wait until windows configures new device.

4.3. Program first run

Launch “AltOne” program using “AltOne” icon on desktop or from “Start Menu → All Programs → Altechna → AltOne → AltOne”. AltOne “Select Device” window will appear. At least one device must be displayed on the list. If the list is empty, please check USB/RS232 cable, power connection. “Status” LED must be active if power is OK. Click refresh button to rescan all ports.

4.4. The “Select Device” tab page

The “Select Device” tab contains a list of the currently connected and powered Altechna supported devices. This window is used to choose the device you want to work with if there are several controllers connected to a single computer. The functions of this window are described in the picture below.

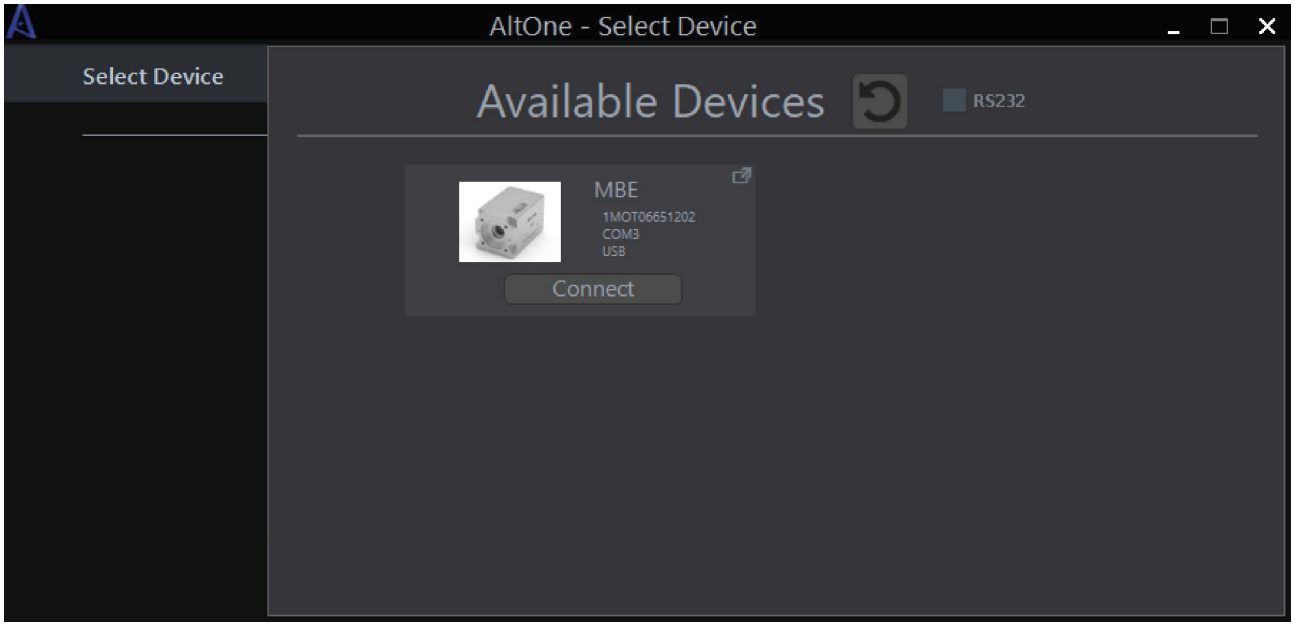


Figure 9. AltOne “Select Device” tab is displayed every time program is started. There is one device (MBE) attached to PC in shown screenshot. Click “Connect” to load control tabs for it.

Every device in the “Select Device” tab list has information about the connected device. This information contains:

- Device picture
- Device type
- Device serial
- Port name
- Connection type (“USB” or “RS232”)

Every device in the list can be connected by using the “Connect” button. After a successful connection to your device, you should be automatically redirected to the main control tab (Fig 11).

In the case of multiple device connections at the same time, use the pop-up icon “” in the top right corner to open the selected device in a separate window.

RS232 connection

In the case of connecting a device via RS232, you should add the devices manually. To do this, check the “RS232” check-box located near the “Refresh” button and a dropdown box with the “Add” button will then appear. When you click on the dropdown box, the software will populate it with all the ports available to the PC. Then, you can just select a specific port and push the “Add” button to add a new RS232 device to the “Select Device” window.

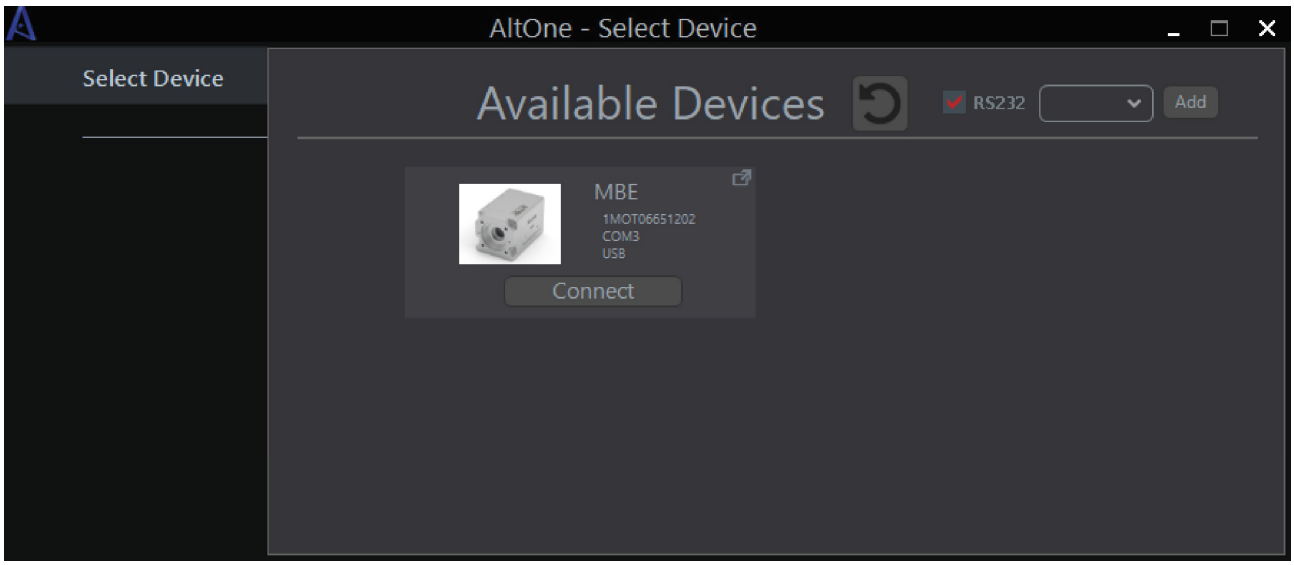


Figure 10. “Select Device” tab with “RS232” checkbox checked.

4.5. The “MBE” main tab

The “MBE” main tab is used to control the beam expansion and adjust the divergence (Fig 11). When the expansion is changed, the divergence lens position is changed automatically (for a collimated beam).

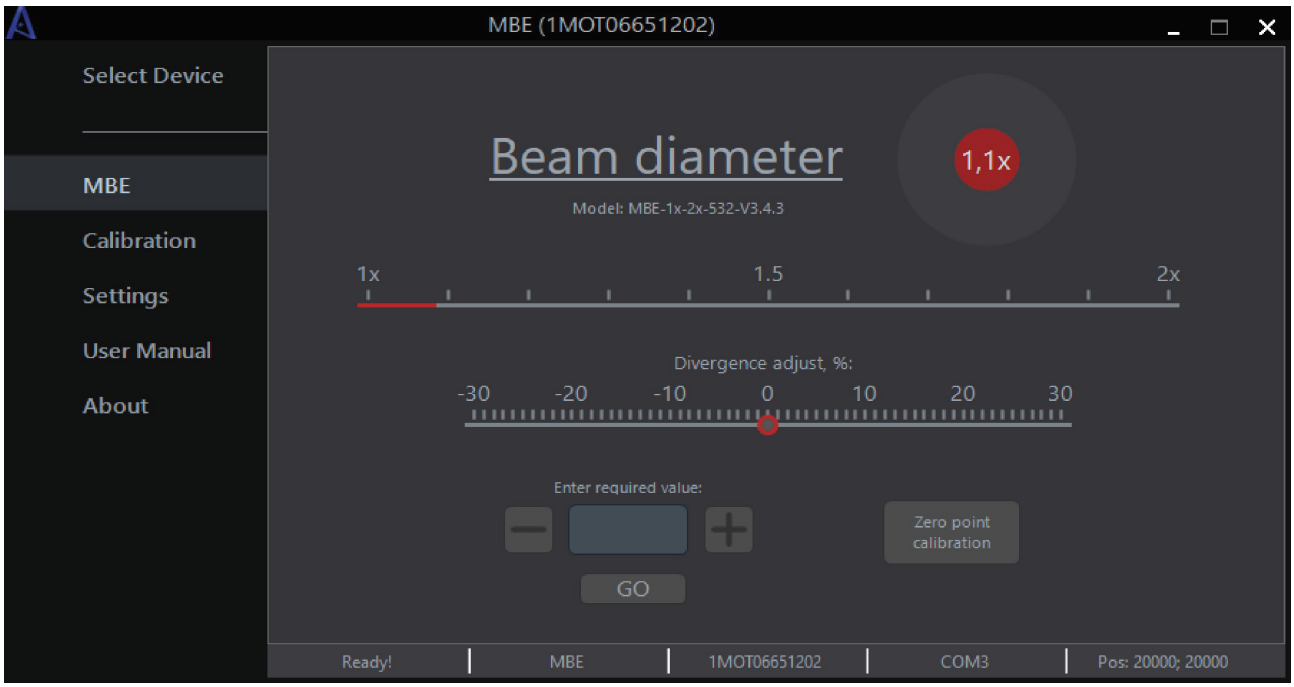


Figure 11. “MBE” main tab.

In the main “MBE” window (Fig. 11) you can change the beam diameter with a slider or by entering value manually.

When you change the beam diameter value, the software automatically adjusts the divergence lens to achieve a collimated beam in the output of the device. If needed, you can use the “Divergence adjust” slider to adjust the divergence (from -30 to +30%) or use custom beam expansion and divergence lens positions to achieve your specific work parameters.

Also, in the main window you can calibrate the zero point by pushing the “Zero-point calibration” button. See the Calibration section.

4.6. The “Calibration” tab

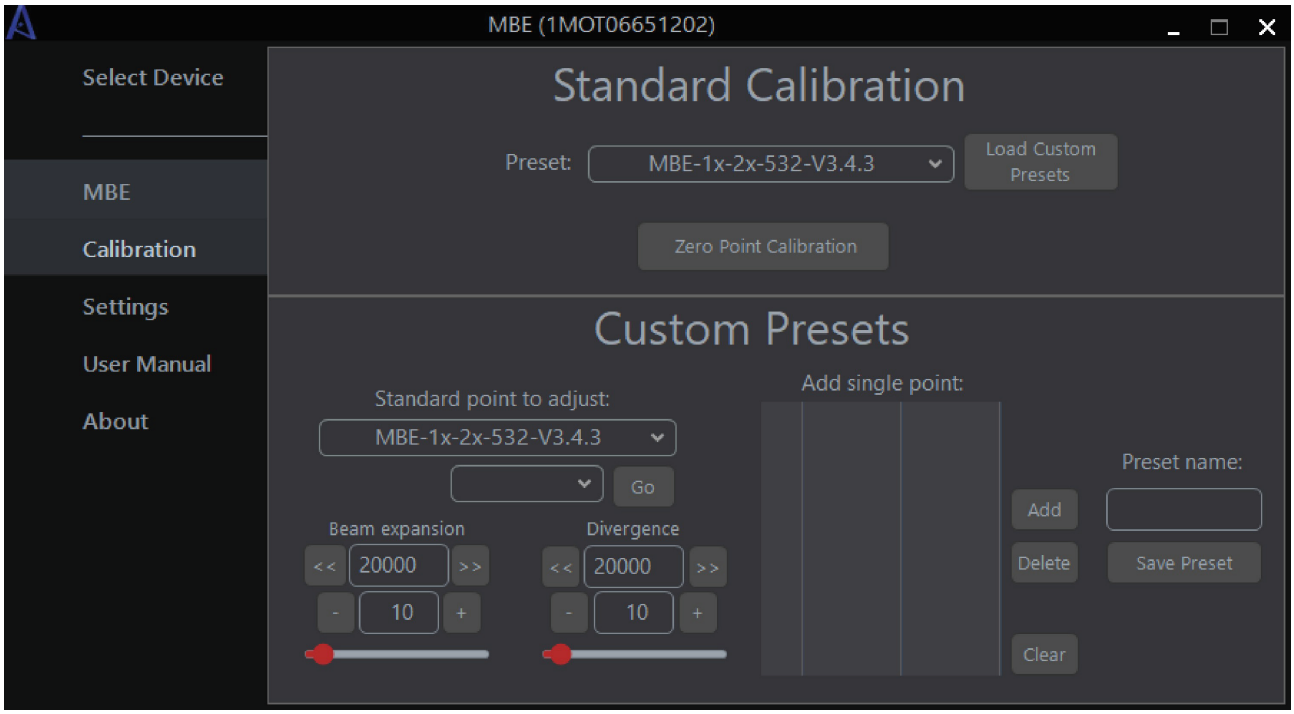


Figure 12. The “MBE” calibration tab.

If your device is not calibrated, all the controls will be disabled. To calibrate your device, go to the “Calibration” tab.

Calibration procedure:
In the “Calibration” window (Fig. 12), you can use the standard or custom presets to calibrate the zero point for precise movement.

Default presets are included in the “AltOne” software. You should choose the preset according to your MBE model. If you are going to work with a different preset than your MBE model, it could potentially damage the device.

If you do not have custom presets, you can create them (in the “Custom Presets” section) based on the standard positions of the lens by trimming those positions or by creating entirely different positions for your specific application. You can do this by changing the “Beam Expansion” and “Divergence” values with a slider, by moving the lens with the buttons or by entering the values manually. After finding the desired “Beam Expansion” and “Divergence” values, you can save them into the list by pushing the “Add” button. You can add up to 10 points to this list with the “Beam Expansion” and “Divergence” values. To save the entire list and use it in the main MBE window, enter a selected preset name in the “Preset name” text box and click “Save preset”. After saving the preset it will automatically load and will be used when changing the expansion values in the main window.

4.7. The “Settings” tab

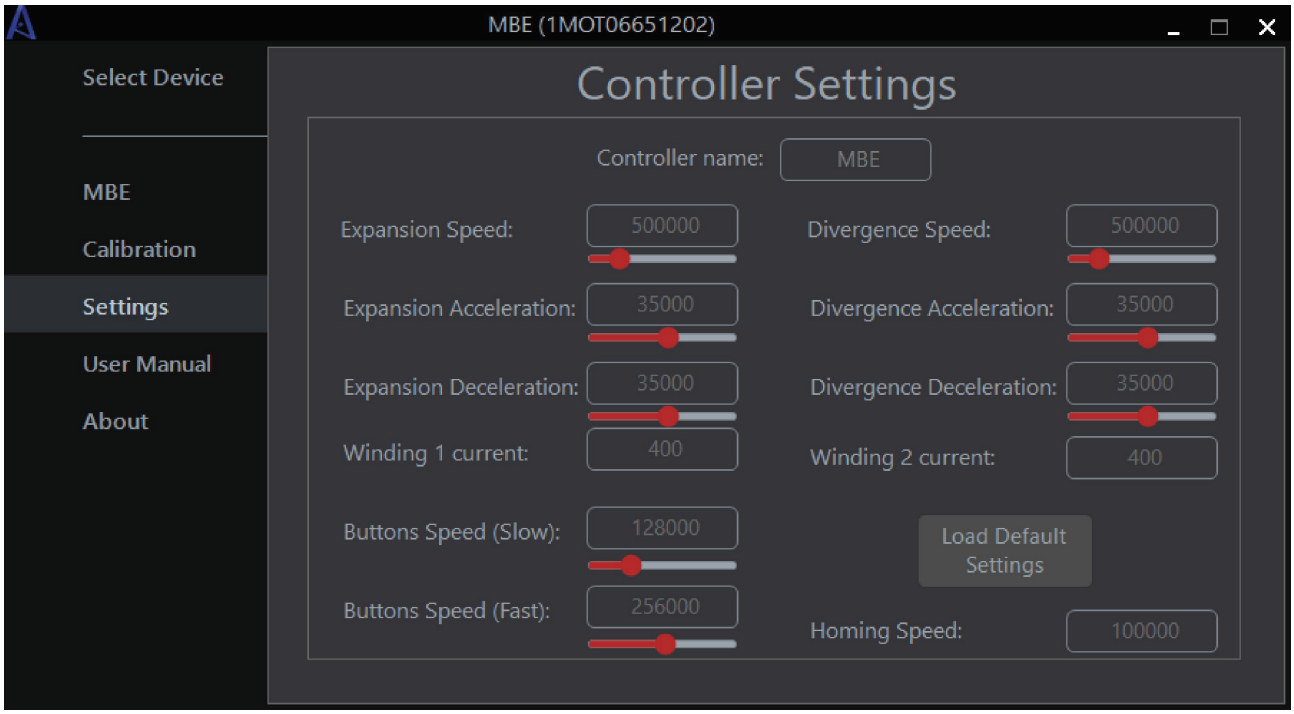


Figure 13. The “MBE” Settings tab.

In the “Settings” window (Fig. 13) you can change the parameters for separate lens movement, change the controller name and load the default settings.

At the top, by changing the “Controller name” text box, you can name your device to keep it separate from other devices you might have.

By changing “Expansion Speed” or “Divergence Speed” with the slider or manually, you can change how fast a specific lens will move in the device. In the same manner you can adjust the acceleration and deceleration of these lenses. The optimal speed values for fast and accurate positioning are the default values. Higher values of the speed and acceleration of the positioning could decrease the repeatability and accuracy of the positioning. Lower values of the speed and acceleration could increase the repeatability and accuracy of the positioning but may decrease the overall speed of the system.
In the “Winding … current” text boxes, you can change the working current of the motors. All values are indicated in “mA”. We suggest not changing these values without consulting with our Technical Support team.

the “Buttons Speed (Slow)” and “Buttons Speed (Fast)” text boxes are used for adjusting the speed that just the expansion lens moves when you press the physical buttons on your controller. When you first press these physical buttons, the expansion lens will move at the “Slow” speed and after 1 second it will start to move at the “Fast” speed.

In the “Homing Speed” text box, you can change the speed at which the zero-point calibration is done for both lenses. A lower homing speed will increase the reference point accuracy.

4.8. The “User Manual” tab

In this tab you can comfortably read user manual of your MBE.

Figure 14. The “User manual” tab.

When you enter this tab, you can maximize this window for greater readability. Also, you can zoom, print, change page and save document as if you were using “Adobe Acrobat Reader” software.

This tab only works if you have “Adobe Acrobat Reader” software for reading PDF files.

4.9. The “About” tab

In this tab you can see your product name, software version and controller’s firmware version.

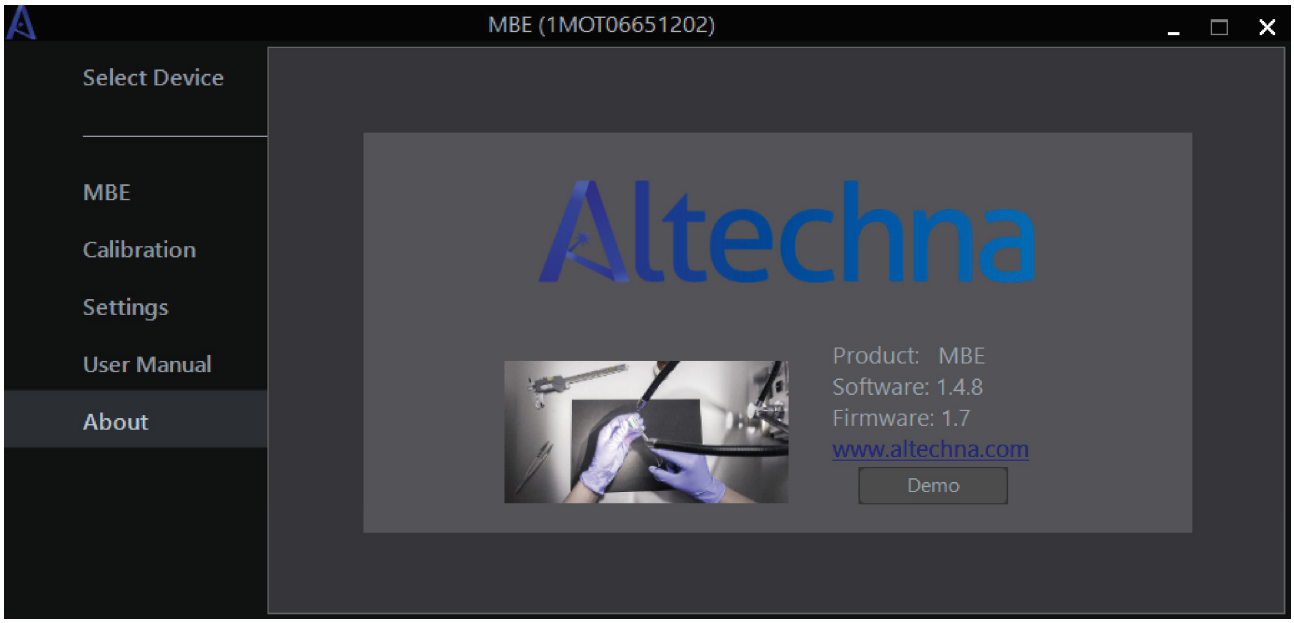


Figure 15. The “About” tab.

5. MBE Controller Hardware

5.1. Controller specifications

MBE controller is two stepper motor driver with specifications listed in Table 4 below.

Characteristic	Rating
Max output voltage	+12 V
Max output current	2 A
Current regulation type	Pulse Width Modulation
Microstepping capability	Steps 1/256
Position feedback	Open loop operation (no external position feedback encoder)
Controller protection	Driver has overheating and over current (2A) protection
Device can be operated by	USB, RS232 and Ethernet
Limit switch	Three limit switches can be connected and used for homing and stopping the motors
Device connector	26 pin D-SUB connector

Table 4. Controller specifications.

5.2. Controller usage

FRONT VIEW

The following connections are situated in the front of MBE controller: power jack (12 V), 26 pin D-SUB for motor, Ethernet and RS232 connectors for MBE (Fig. 16).

Figure 16. Front view of MBE Controller.

TOP VIEW

There are two buttons in the top of the MBE controller, "+" and "-".

- **"+"** – while holding moves expansion lens to a maximum position
- **"-"** – while holding moves expansion lens to a minimum position
- **HOME** - holding both buttons **for 3 sec.** moves both lenses to home position

There are 4 LED indicators:

- **Power –** indicates power from USB connection with PC.
- **Status:**
  - If blinking slowly (1 blink in 1 sec.) – need homing
  - If blinking fast (3 blinks in 1 sec.) – the lens position is changing
  - If illuminating continuously – MBE is homed, the movement is stopped.
- **MIN –** the MBE expansion lens is at its minimum position
- **MAX –** the MBE expansion lens is at its maximum position

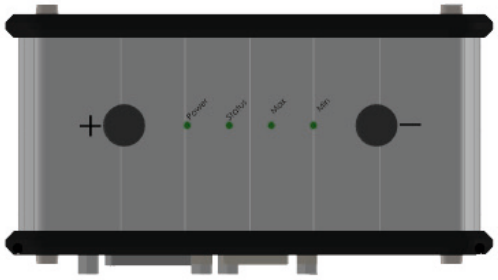


Figure 17. Top view of MBE Controller.

6. Serial and Ethernet Communication

6.1. Serial protocol description

When using “RS232” (or “USB”*) port all communication between controller and PC (or other device) is established by using RS232 communication protocol.

COM-port parameters are fixed on the controller-side:

Speed:	115200 baud
Frame size:	8 bits
Stop-bits:	1 bits
Parity:	none
Flow control:	none
Byte receive timeout:	400 ms

*USB port simulates COM port on a PC.

All the commands are the same for all communication protocols (see “6.4 Controller Commands”). For more information, please see “RS232 communication example” software and C# code (Visual Studio project).

6.2. Ethernet protocol description

When using an “Ethernet” port, all communication between the controller and the PC (or another device) is established by using the TCP/IP communication protocol. When the controller is connected to an “Ethernet” port, it constantly tries to connect to the device (server) with the IP address that is specified in the controller’s memory. To connect and communicate with the controller, you should create a dedicated TCP server that will send and receive data to and from controller.

To achieve a connection between the server and the controller, you must configure the specific addresses and ports in the controller and the server (they must match in both devices). You can change the controller’s IP address and port using the “TCP sample” software.

When creating a server on your device, don’t use dynamic IP (DHCP), because your server IP address should be static for the controller to maintain the connection.

The commands are the same for all communication protocols (see “6.4 Controller Commands”). For more information, please see the “TCP sample” - software and C# code (Visual Studio project).

6.3. Command execution

All data transfers are initiated by the PC, meaning that the controller waits for incoming commands and replies accordingly. Each command is followed by a controller response, with rare exceptions in the case of some service commands. One should not send another command without waiting for the answer of the previous command.

Command processing does not affect the real-time engine control (PWM).

Both the controller and the PC have an IO buffer. Received commands and the command data are processed once and then removed from the buffer. Each command consists of a 6-byte identifier and optionally a data section followed by its 2-byte CRC. Data can be transmitted in both directions, from the PC to the controller and vice versa. A command is scheduled for execution if it is a legitimate command and (in the case of data) if its CRC matches. After processing a correct command, the controller replies with a 1-byte response – AA (for "OK") or 01 (for "Not OK"), followed by the data and its 2-byte CRC if the command is supposed to return data.

The device is a slave – it can't initiate any data transfer. The PC is a master. Each data transfer session looks like this:

	Description	**Note**
PC	Send command	
Device	Confirm correct command receive (0xAA or 0x01)	If 0x01 (not correct command), PC should send command again
Device	Send additional data to PC	If needed to complete command

NOTE: OK - 0xAA, NOT OK - 0x01.

In case of an error in data from device to PC, PC will initiate new session starting from sending a command.
Each frame consists of **start symbol (@)**, data string **length (16 bits)**, ASCII **command (always 3 bytes)**, **data** (if need to transmit some data for command execution), **CRC (16 bits)** which is calculated for ASCII command and DATA fields:

Start	**Length0, Length1**		**ASCII command**			**DATA**		**CRC0, CRC1**	
@	0x00	0x00	A	B	C	……	……..	0x00	0x00

Note for RS232 interface: device recognizes end of the data frame if the pause after last transferred byte is > 50 ms.

For each correctly received command device answers - **0xAA**. If command is not correct it will answer - **0x01**.

Example command for homing:

Start ("@")	**Length0, Length1 (3 bytes)**		**ASCII command ("hom")**			**DATA (none)**		**CRC0, CRC1**	
0x40	0x03	0x00	0x68	0x6f	0x6d	……	…….	0xd5	0x94

Example command for moving to "123456" absolute position:

Start ("@")	**Length0, Length1 (3 bytes)**		**ASCII command ("rad")**			**DATA ("123456")**				**CRC0, CRC1**	
0x40	0x07	0x00	0x72	0x61	0x64	0x40	0xE2	0x01	0x00	0x1C	0xFD

Device calculate CRC, using this procedure (C):

```
uint16_t myCRC16_CCITT_CRC(uint16_t *crc, uint8_t c)
{
            uint8_t  i;
            uint16_t  cc;

            cc = ( (uint16_t ) c ) << 8;

            for ( i=0; i<8; i++ )
            {

            If (( (*crc) ^ cc ) & 0x8000)

            (*crc) = ((*crc) << 1) ^ MTT;


            else (*crc) <<= 1;

            cc <<= 1;
            }
            return *crc;
}
```

Procedure for PC (C#):

```
//CRC16 XMODEM implementation
private static short calcCrc(byte[] data)
{
             unchecked
        {
           short crc = 0;

           for (int a = 0; a < data.Length; a++)
           {
              crc ^= (short)(data[a] << 8);
              for (int i = 0; i < 8; i++)
              {
                 if ((crc & 0x8000) != 0)
                    crc = (short)((crc << 1) ^ 0x1021);
                 else
                    crc = (short)(crc << 1);
              }
           }
           return crc;
        }
}
```

6.4. Controller commands

Movement commands				
Command	**Data**	**Return values**	**Description**	**Notes**
“hom” (0x686F6D)	none	0xAA or 0x01 (OK or NOTOK)	Start the homing procedure for **expansion** motor.	
“ho2” (0x686F32)	none	0xAA or 0x01 (OK or NOTOK)	Start the homing procedure for **divergence** motor.	
“hob” (0x686F62)	none	0xAA or 0x01 (OK or NOTOK)	Start the homing procedure for **both** motors.	
“rad” (0x726164)	4 bytes	0xAA or 0x01	Move **expansion** lens to absolute micro step position x.	Woks only when device **is “Homed”.**
“ra2” (0x726132)	4 bytes	0xAA or 0x01	Move **divergence** lens to absolute micro step position x.	Woks only when device **is “Homed”.**
“rab” (0x726162)	4 bytes	0xAA or 0x01	Move **both** lenses to absolute micro step position x.	Woks only when device **is “Homed”.**
“rgd” (0x726764)	4 bytes	0xAA or 0x01	Move **expansion** lens x micro steps relative to the current position.	Woks only when device **is “Homed”.**
“rg2” (0x726732)	4 bytes	0xAA or 0x01	Move **divergence** lens x micro steps relative to the current position.	Woks only when device **is “Homed”.**
“rgs” (0x726773)	4 bytes	0xAA or 0x01	Move **expansion** lens x micro steps relative to the current position.	Works even if device **is not “Homed”.**
“rs2” (0x727332)	4 bytes	0xAA or 0x01	Move **divergence** lens x micro steps relative to the current position.	Works even if device **is not “Homed”.**
“stp” (0x737470)	none	0xAA or 0x01	Stop **expansion** motor smoothly if it is currently running.	
“st2” (0x737432)	none	0xAA or 0x01	Stop **divergence** motor smoothly if it is currently running.	
“stb” (0x737462)	none	0xAA or 0x01	Stop **both** motors smoothly if it is currently running.	

Status and information commands				
Command	**Data**	**Return values***	**Description**	**Notes**
“ost” (0x6F7374)	none	8 bytes (**debug**) + 4 bytes (**flags**) + 4 bytes (**position**) + 8 bytes (**debug**)	Get controller run state, position, limit switch states and etc. (**For expansion motor**)	**“debug”** bytes are just for debugging purposes. **“flags”** (logic 1 – active): bit0 – device is running bit1 – homing in progress bit2 – device is not homed bit3 – hardware error, can’t move bit4 – calibration corrupted bit5, bit6, bit7 – unused bit8 – motor driver reset detected bit9 – driver hi temp warning bit10 – left LS is pressed bit11 – error in load (motor) bit12 – motor driver error bit13 – stallguard is active bit14 – motor is standstill bit15 – target velocity reached bit16 – driver over temperature bit17 – target position reached bit18 – under voltage error bit19 – right LS is pressed bit20 – device is homed bit21 – calibration is done bit22 – load warning (open load detected on motor phase A or B) bit23 – device FRAM error

“os2” (0x6F7332)	none	8 bytes (**debug**) + 4 bytes (**flags**) + 4 bytes (**position**) + 8 bytes (**debug**)	Get controller run state, position, limit switch states and etc. (**For divergence motor)**	Same structure as in “ost” command.
“osb” (0x6F7362)	none	4 bytes (**expansion flags**) + 4 bytes (**expansion position**) + 4 bytes (**divergence flags**) + 4 bytes (**divergence position**)	Get controller run state, position, limit switch states and etc. (**For both motors)**	Same “flags” structure as in “ost” command.
“cd ” (0x6F7320)	none	4 bytes + 4 bytes + 4 bytes + 4 bytes + 4 bytes + 1 byte + 4 bytes + 4 bytes + 4 bytes + 4 bytes + 4 bytes + 10 bytes + 8 bytes + 8 bytes + 8 bytes + 8 bytes + 1 byte + 1 byte	Gets all parameters for **expansion** motor.	**Data received:** microsteps_per_degree (float) speed (Int32, usteps/sec) acceleration (int32, usteps/sec) deceleration (int32, usteps/sec)) winding current (int32), mA limit_flags (uint8) timeout_speed_ms (int32) speed_btn_slow (int32) speed_btn_fast (int32) speed_home(int32) offset_microsteps (int32) min_power (float) max_power (float) measurement units (string 5 char) preset_but0 (double) preset_but1 (double) preset_but2 (double) preset_but3 (double) preset_but4 (double) flags_gui (uint8) flags_user (uint8)
“cd2” (0x6F7332)	none	4 bytes + 4 bytes + 4 bytes + 4 bytes + 4 bytes + 1 byte + 4 bytes + 4 bytes + 4 bytes + 4 bytes + 4 bytes + 10 bytes + 8 bytes + 8 bytes + 8 bytes + 8 bytes + 1 byte + 1 byte	Gets all parameters for **divergence** motor.	**Same received data structure as in “cd ” command.**
“pw ” (0x707720)	none	16 bytes (char)	Show serial number of a device.	
“n ” (0x6E2020)	none	17 bytes (char)	Show name of a device.	
“v ” (0x762020)	none	5 bytes (char)	Show firmware version of a device.	
“p ” (0x702020)	none	5 bytes (char)	Returns “pUSB:” if device is connected.	This command can be used to “ping” controller (to check if controller is attached to particular COM port).

Configuration commands				
Command	**Data**	**Return values**	**Description**	**Notes**
“spd” (0x737064)	1+4 bytes	0xAA or 0x01 (OK or NOTOK)	Set **expansion** motor speed. Value should be from 0 to 8000000. **First byte is for motor number.**	microsteps/t** **(Default value: 500000)**
“acl” (0x61636C)	1+4 bytes	0xAA or 0x01	Set motor acceleration. Value should be from 0 to 65535. **First byte is for motor number.**	microsteps/ta*** **(Default value: 30096)**
“dcl” (0x64636C)	1+4 bytes	0xAA or 0x01	Set motor deceleration. Value should be from 0 to 65535. **First byte is for motor number.**	microsteps/ta*** **(Default value: 30096)**
“wcr” (0x776372)	1+4 bytes	0xAA or 0x01	Set motor drive current. Value should be from 50 to 600 mA. **First byte is for motor number.**	Higher current will generate more heat. **(Default value: 400 mA)**
“hcr” (0x686372)	1+4 bytes	0xAA or 0x01	Set motor hold in position current when motor is idle. Value should be from 50 to 600 mA. **First byte is for motor number.**	Higher current will generate more heat. **(Default value: 100 mA)**
“sav” (0x736176)	4 bytes + 4 bytes + 4 bytes + 4 bytes + 4 bytes + 1 byte + 4 bytes + 4 bytes + 4 bytes + 4 bytes + 4 bytes + 10 bytes + 8 bytes + 8 bytes + 8 bytes + 8 bytes + 1 byte + 1 byte	0xAA or 0x01	Saves all parameters for **expansion** motor.	For sending - s**ame data structure as in „cd „ command.**
“sa2” (0x736132)	4 bytes + 4 bytes + 4 bytes + 4 bytes + 4 bytes + 1 byte + 4 bytes + 4 bytes + 4 bytes + 4 bytes + 4 bytes + 10 bytes + 8 bytes + 8 bytes + 8 bytes + 8 bytes + 1 byte + 1 byte	0xAA or 0x01	Saves all parameters for **divergence** motor.	For sending - s**ame data structure as in „cd „ command.**
“sn ” (0x736E20)	<= 17 bytes (char)	0xAA or 0x01	Write maximum of 17 character long name to a device.	
“clg” (0x636C62)	none	0xAA or 0x01	Clear device calibration info and set it to default values.	

Table 5. Controller commands.

***** If command returns data, in front of data will always be “0xAA (OK)” and data length (2 bytes). At the end it should have 2 byte CRC.

**** “t” –** time equivalent for speed. t = 1.39810.

***** “ta” –** time equivalent for acceleration and deceleration. ta = 0.01527.

Tel. +370 527 25 738
Fax +370 527 23 704
info@altechna.com

Mokslininku st. 6A
Vilnius, Lithuania

www.altechna.com