MANUAL

CONFIGURATION PROGRAM

FOR "CUA-USB" AND "CUA-ETH" CONTROL UNITS

NEWSON NV

Table of Contents

1 SETUP COMMUNICATION

1.1 ETHERNET COMMUNICATION WITH "CUA-ETH" DEVICE

By default, CUA-ETH control units are configured with a static IP address 172.16.224.20 and netmask 255.255.0.0. In order to address the control unit, the IP addresses of the PC must be on the same Local Area Network and subnet. Therefor when you try to access the control unit for the first time, the IP address has to be in the 172.16.x.x domain. A valid configuration on the PC would be a static IP 172.16.224.100 with netmask 255.255.0.0.

It is also possible to change the IP address of the control unit using CuaEthSetup.exe. This software is part of the rhothor™ installation package. The device MAC-address is labelled on the CUA-ETH controller. Using the last 2 numbers of this MAC address the CuaEthSetup.exe software can broadcast a special UDP packet on the Local Area Network to modify the IP of a CUA-ETH controller.

Note: The subnet mask indicates that the first 2 octets in each IP address on the network must be the same number. This allows network devices to communicate on the Local Area Network (LAN). For more information on configuring a static IP address in Windows, see the Windows help files. You can access Windows help files by clicking the Windows Start button and selecting Help. Search in Windows help for static IP address.

1.2 AUTOMATIC "CUA-USB" DRIVER INSTALLATION ON WINDOWS8 / WINDOWS10

CUA-USB controllers use a standard windows device driver (Winusb.sys) and can be installed automatically on Windows 8 and later Windows OS versions. When you connect a CUA-USB device, the system reads device information and loads Winusb.sys automatically.

1.3 MANUAL INSTALLATION OF "CUA-USB" DRIVER ON WINDOWS7 / VISTA / XP

For older operating systems, WinUSB installation is part of a driver package which can be downloaded from newson's website. This driver package is an unsigned driver installation package. This means you might need to disable driver signing enforcement when installing the CUA-USB controller on windows x64 versions.

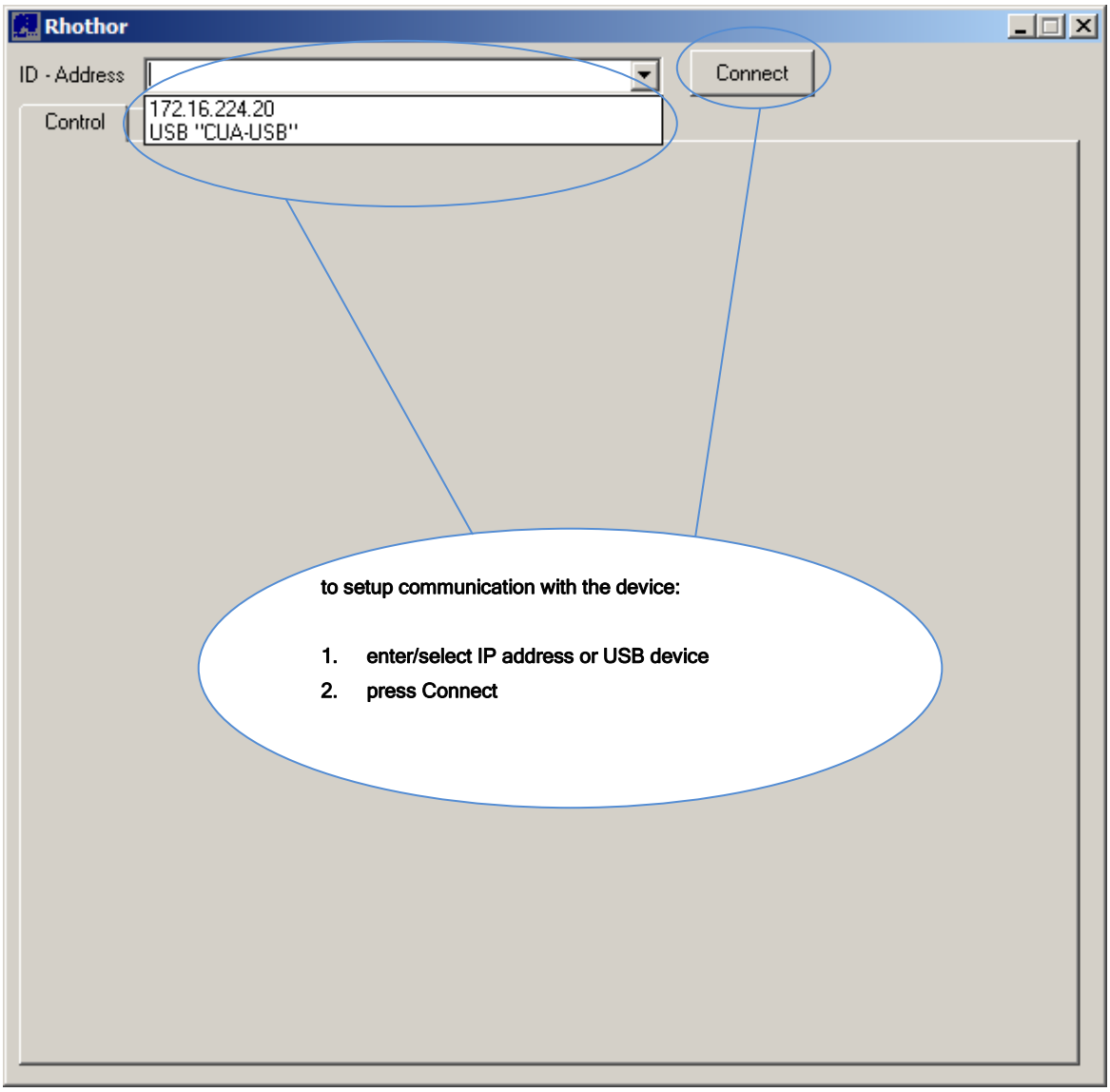
For windows 7/Vista

1. Connect the CUA-USB controller to the computer with an USB cable.
2. Go to start and right-click Computers and select **Manage.** This will bring the Computer Management Window
3. Open Device Manager, right-click the new device and click **Update Driver Software**
4. You have two options here, select **Browse my computer for driver software**
5. Click **Browse** and point to the rhothor.inf file of the driver folder (part of the driver package), and then click **Open**
6. Click **Install this driver software anyway,** click **Close**

For Windows XP

1. On the **Welcome to the Found New Hardware Wizard** screen verify that **Install from a list or specific location (Advanced)** is selected, and then click **Next**.
2. When the **Please choose your search and installation options** appears, select **Search for the best driver** in these locations. Verify that the item **Include this location in the search** is checked, click **Browse** and point to the rhothor.inf file of the driver folder (part of the driver package), and then click **Next**
3. When the **Windows Logo testing** dialog appears, click **Continue Anyway**.
4. On the **Completing the Found New Hardware Wizard** screen, click **Finish**.

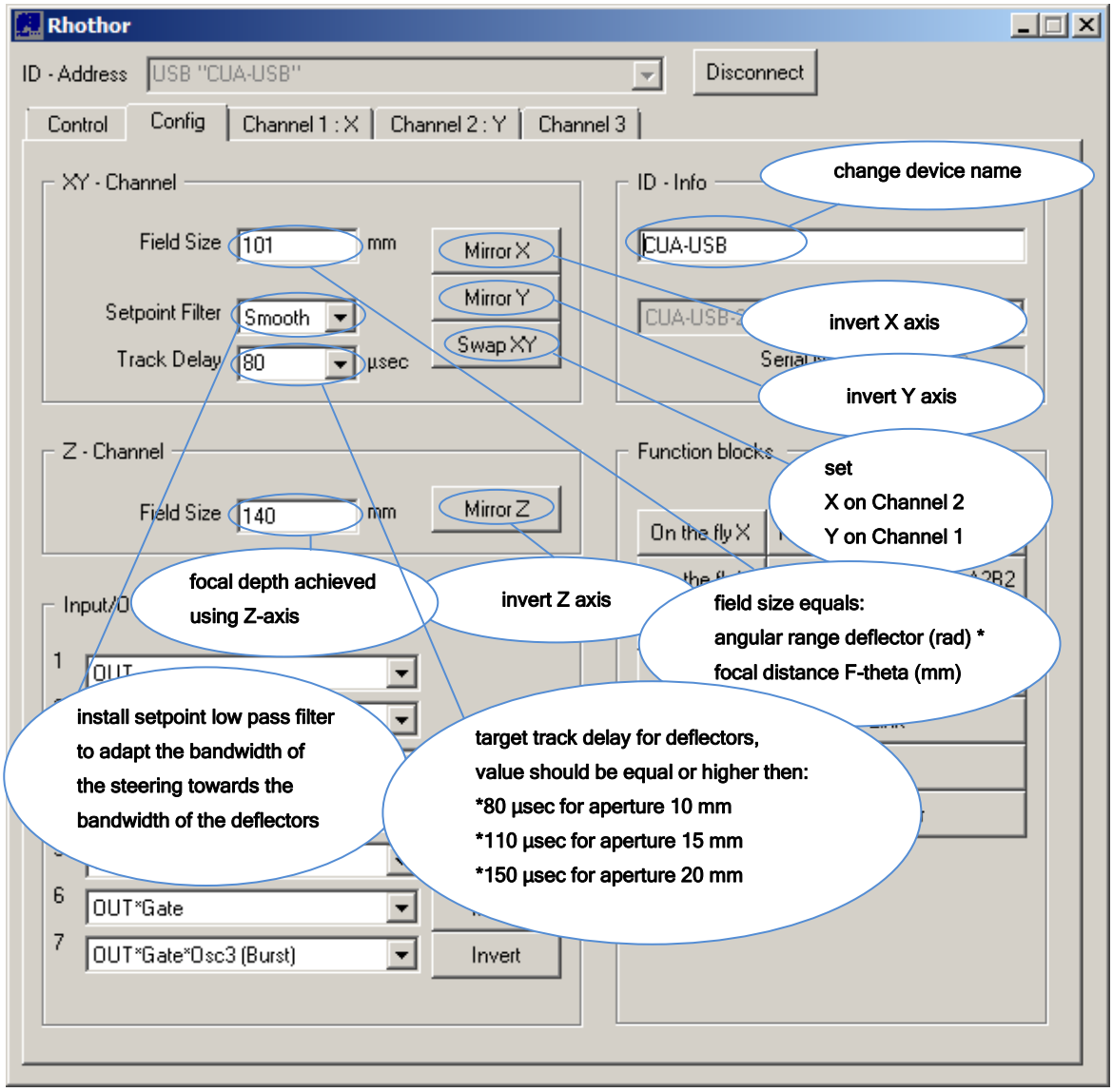
2 STARTING RHOTHOR.EXE



3 CONFIG TAB

Before using the deflection system, its properties should be set. Press the “Config” tab to change the configuration settings.

3.1 GLOBAL SETTINGS



3.1.1 Mirror/Swap

It is possible to define the coordinate system of deflection system by clicking the “MirrorX”, “MirrorY”, “MirrorZ” and “Swap XY” buttons

3.1.2 Device Name

It is possible enter a device name in the ID-Info text field. The device name is usefull for addressing when multiple CUA-USB devices are connected to the same computer. An unique device name allows to connect to the correct control board.

3.1.3 Fieldsize

Fieldsize is the size of the marking area in mm. This is the area that the system can cover. The range depends on the angular range of the installed rhothor™ deflectors and the focal length of the mounted optics.

The value to be entered can be calculated:

FIELDSIZE = FOCAL_LENGTH * ANGULAR_RANGE/1000

"focal_length" is the focal length of the used lens.
"angular_range" is the optical scanning range of the installed motors in milli radians.

Example:
If an f-theta 163 mm flat field lens is used on a system fitted with RTA-AR800 motors the fieldsize that has to entered equals:

fieldsize = focal_length * angular_range/1000
= 163*800/1000 = 130 mm => fill in 131

Trial and error is another way to determine the value that has to be entered. Mark a square and measure its size. The relation between measured size and required size of the square can be used to change the fieldsize setting. The square needs to be positioned at centre of the marking area and must be smaller than the fieldsize.

Always add 1mm to the calculated fieldsize in order to avoid overflow results when moving to the boundaries of the deflection system.

3.1.4 Track Delay

A rhothor™ deflection system uses digital regulator technology to control the motors in a closed loop fashion. Track delay is the time delay between setpoint value and actual value of a closed loop system. In an ideal system this track delay is zero, all steering commands are executed immediately. However zero track delay is not possible. The deflectors need time to accelerate. Track delay allows those motors to execute the steering commands. The track delay value defines the dynamics of the connected deflector. Choosing a low value will reduce rounding errors at corners. There are limits to take into consideration:

Aperture (mm)	Track delay (µsec)
10	>=80
15	>=110
20	>=150

Settings values lower than those limits will not damage the system, but give it a tendency to go into oscillation. Furthermore there is a relation between drag error and desired maximum marking speed. If a high maximum marking speed value is needed, the track delay should be increased giving more time to the motors to accelerate.

3.1.5 Setpoint filter

When there are significant changes in the setpoint stream send to the scanner it can generate overshoot and oscillations in the scanner. This is the result of the PID controller reacting to the large target change. The integrator part of the PID regulation will cause an overshoot.

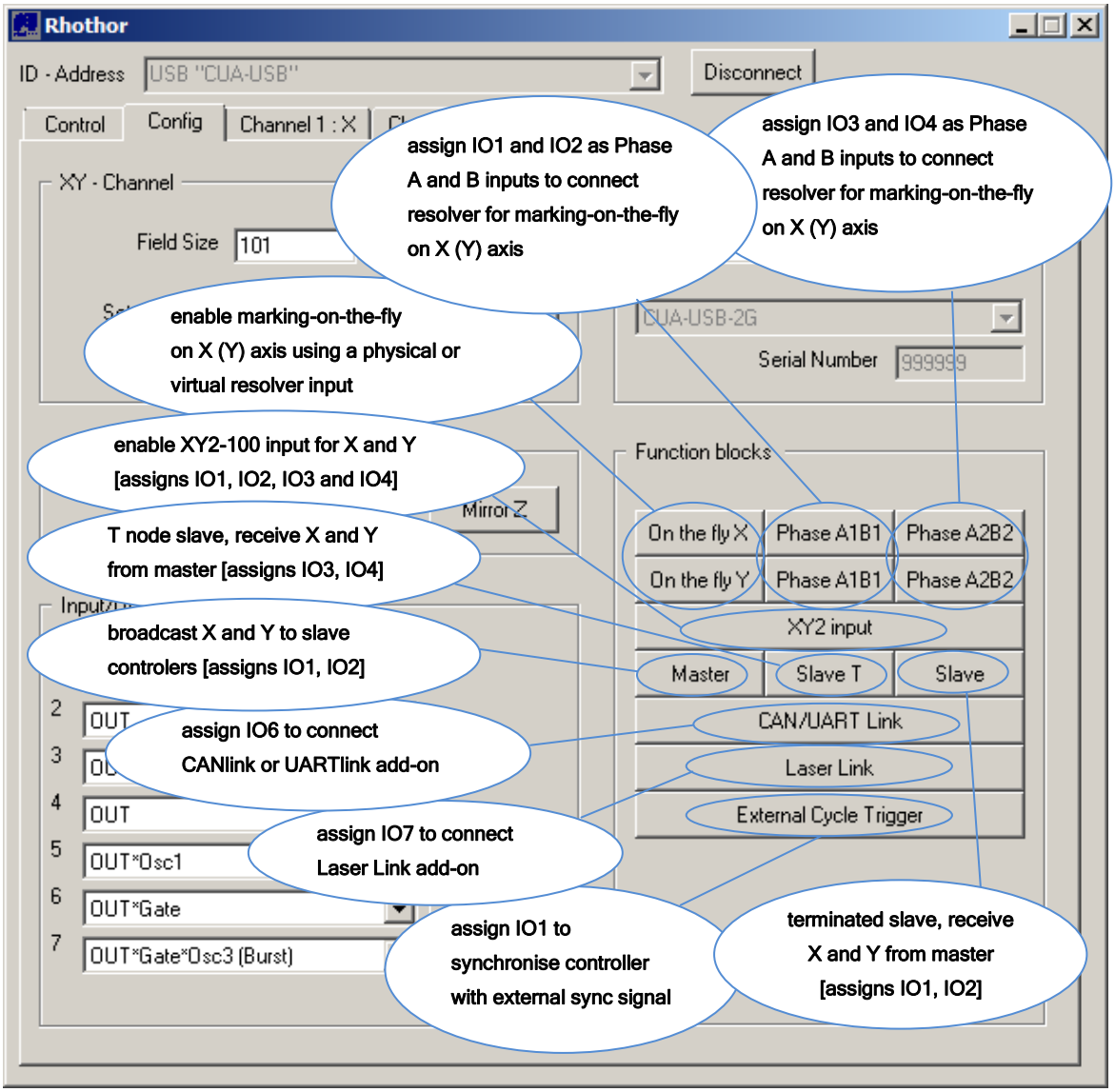
A setpoint filter allows achieving the desired change in setpoint but avoids overshoots by passing the setpoint stream through a low-pass filter and this way improving the PID's ability to regulate control and maintain performance. The bandwidth of the steering is adapted to the bandwidth of the scanner.

A setpoint filter is installed by choosing between 4 possibilities: “None”, “Sharp”, “Normal” and “Smooth”. This setting defines the time constant of setpoint low pass filter. The default value is “Normal”.

Filter	Filter Time Constant
None	-
Sharp	0.5*Track delay
Normal	Track delay
Smooth	1.5*Track delay

Note: When installing a setpoint filter, an extra time delay will be added between the steering setpoint and the actual track.(the filter time contant) This delay can be compensated in applications by modifying the laser timings using rhothor library function “rtSetLaserTimes”.

3.2 FUNCTION BLOCKS



When activating function blocks, certain I/O’s will be assigned for certain functions. Some function blocks use internally the same logic and cannot be combined together.

IO ASSIGNMENT	IO1	IO2	IO3	IO4	IO5	IO6	IO7
On the fly X	-	-	-	-	-	-	-
On the fly Y	-	-	-	-	-	-	-
On the fly X + Phase A1B1	PHASE A1	PHASE B1	-	-	-	-	-
On the fly X + Phase A2B2	-	-	PHASE A2	PHASE B2	-	-	-
On the fly Y + Phase A1B1	PHASE A1	PHASE B1	-	-	-	-	-
On the fly Y + Phase A2B2	-	-	PHASE A2	PHASE B2	-	-	-
XY2-100 input	SENDCK	SYNC	CHANNEL X	CHANNEL Y	-	-	-
Master	MASTER	MASTER	-	-	-	-	-
Slave T	-	-	-	-	SLAVE	SLAVE	-
Slave	SLAVE	SLAVE	-	-	-	-	-
CAN/UART Link	-	-	-	-	-	CAN/UARTK	-
Laser Link	-	-	-	-	-	-	LASER LINK
External Cycle Trigger	CYCLE SYNC	-	-	-	-	-	-

3.2.1 On the Fly with Constant Speed

When only function block “On the fly X” or function block “On the fly Y” is activated, the CUA controller will use a virtual encoder to simulate constant speed. The step size of the virtual encoder will be added each 5µs to the encoder position to simulate a constant speed. The properties of the virtual encoder can be changed through rhothor™ library command “rtSetResolver”.

3.2.2 On the Fly with Encoder Input

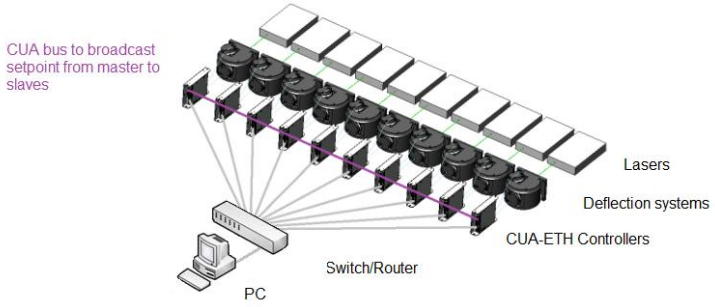
When function blocks “On the fly X + PhaseA1B1” or “On the fly X + PhaseA2B2” are activated, the CUA controller will process the encoder signals in real-time and compensate X axis for any encoder position changes. It is also possible to support dual axis On-The-Fly by combining for instance “On the fly X + PhaseA1B1” with “On the fly Y + PhaseA2B2” The encoder itself can be enabled through rhothor™ library command “rtSetResolver”.

3.2.3 XY2-100 Input

When function block “XY2-100” activated the CUA controller acts as a XY2-100 standard slave controller for XY deflection.

3.2.4 Master/Slave

Master/Slave is a communication architecture where one device (the master) has unidirectional control over one or more devices (slaves). CUA controller boards can be configured to implement this kind of architecture. A master will broadcast the target positions to all the slaves who will simultaneously follow this master track.



In order to create such a CUA bus network:

- configure 1 CUA controller with function block “Master” enabled
- configure 0, 1 or n controllers with function block “Slave T” enabled
- configure 1 CUA controller with function block “Slave” enabled
- The physical bus consists of two twisted pairs.

  Twisted pair1: $IO1_{MASTER} \rightarrow IO5_{SLAVE\_T,0} \rightarrow IO5_{SLAVE\_T,1} \rightarrow \dots \rightarrow IO5_{SLAVE\_T,n} \rightarrow IO1_{SLAVE}$

  Twisted pair2: $IO2_{MASTER} \rightarrow IO6_{SLAVE\_T,0} \rightarrow IO6_{SLAVE\_T,1} \rightarrow \dots \rightarrow IO6_{SLAVE\_T,n} \rightarrow IO2_{SLAVE}$

3.2.5 CAN/UART Link

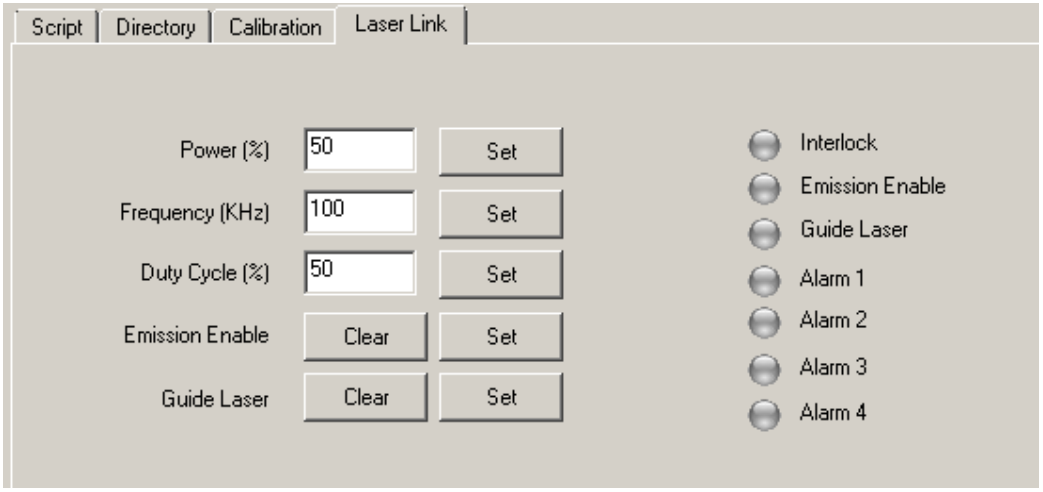
In order to work with a CANlink or a UARTlink hardware add-on, the extension board needs to be activated on the CUA controller by enabling the “CAN/UART Link” function block. Once enabled CANlink and UARTlink can be accessed through dedicated rhothor™ library functions.

3.2.6 LaserLink

In order to work with a LaserLink module, the link needs to be activated on the CUA controller by enabling the “Laser Link” function block. Once enabled and the LLink is connected to IO7 “LASER LINK” the laser control module can be accessed through dedicated rhothor™ library functions.

Also when the LLink is recognised by the rhothor™ software an extra tab page appears on the Control tab. This tab is called “Laser Link” and on this page it is possible to control the attached laser through the LaserLink module.

As an example: When the CUA controller discovers the LLink-08 then following page is available.



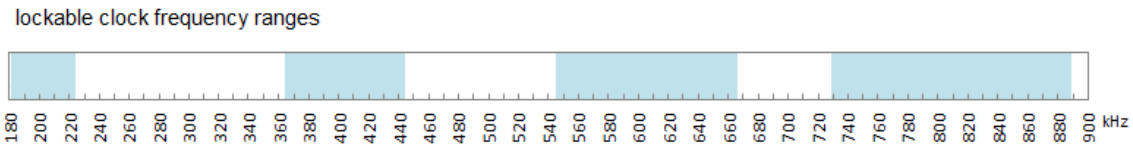
(The LLink-08 is a CUA add-on to control lasers from IPG, YLP series directly)

3.2.7 External Cycle Trigger

When the function block “External Cycle Trigger” is activated, the cycle time of the CUA controller is synchronised with an external running clock. The external clock signal should be connected to IO1 “CYCLE SYNC”. The synchronisation is done by altering the DSP main clock which has standard a cycle time of 5 µsec. The cycle processing locks with an external trigger deviating the cycle time from the standard period of 5 µsec.

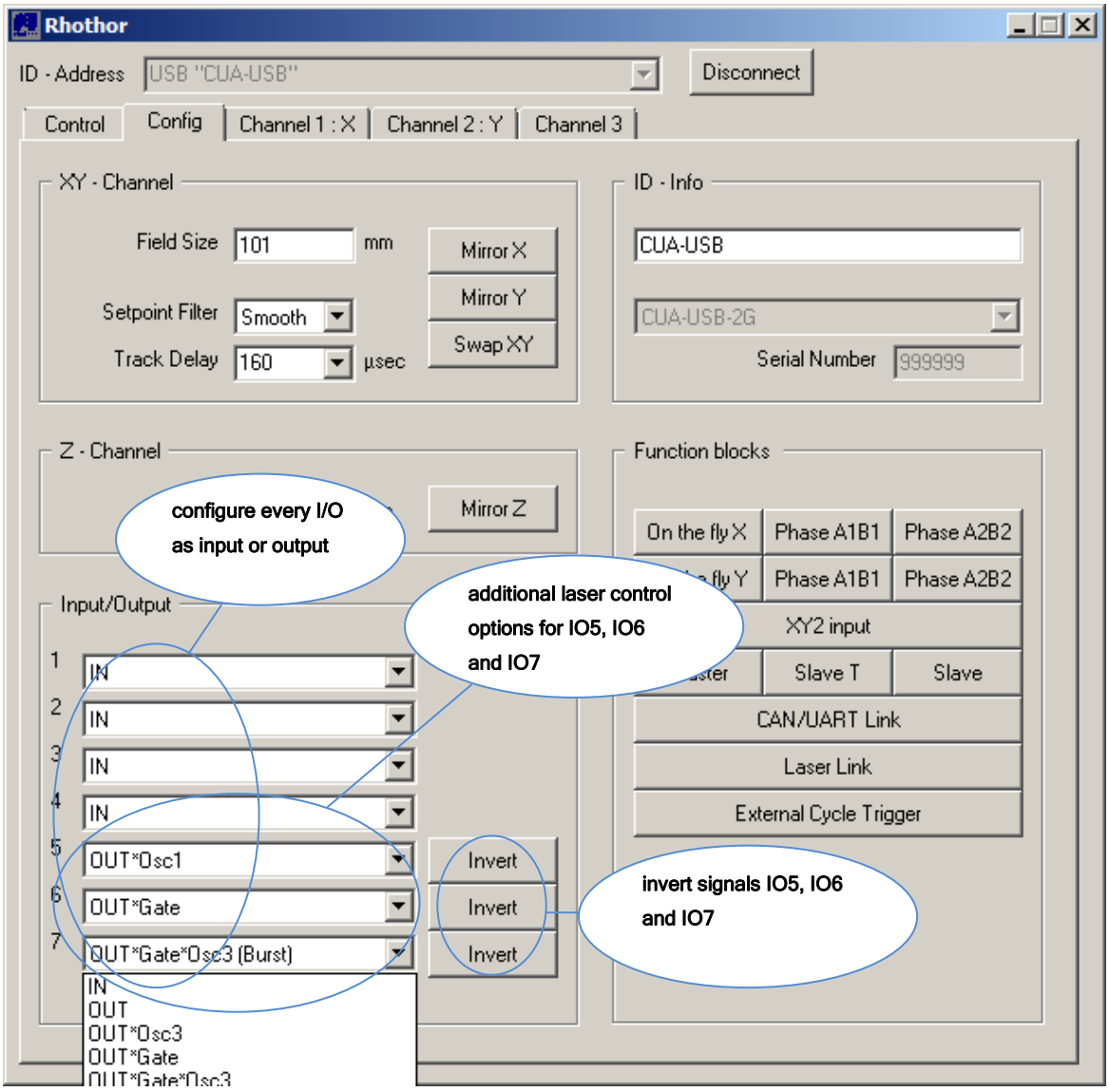
The locking will lie between 4.5 µsec to 5.5 µsec (10%). This means that de external clock must have a period so that a n-times of that period lies within said range:

In case of 650 kHz clock the system locks successfully at 3 cycles, resulting in 4.6 µsec cycle time.
In case of 700 kHz clock the system does not lock.
In case of 750 kHz clock the system locks successfully at 4 cycles, resulting in 5.3 µsec cycle time.



3.3 INPUT/OUTPUT

CUA controllers have 7 differential input/outputs which can be configured for different purposes. Setup the configuration by selecting the desired function from the dropdown list.



When an activated function block uses some of the IO's, they become allocated and will be shown reserved (disabled). All other ones are free to be set as an input or an output. All IO pins use RS485 compatible drivers. Connection to different signal types should be done through interface electronics.

IO channels 5, 6 and 7 are intended to serve as an interface to the laser. When the system is processing vectors not only the motors but also the laser has to be controlled in a synchronised fashion. A deflection system with integrated control can be connected to almost any laser type. Three programmable oscillators, each linked with one of these IO's, can be interconnected to generate laser signals. The oscillators can be set to generate the laser's trigger frequency or provide a pulse width modulated output for generating analog values.

To enable the laser signal on the output, the "OUT" state of the output has to be set High. This can be done by calling the rhorhor library function "rtSetIO" or setting it manually on the "Control" tab.

Selections IO5

1. IN: use as an input
2. OUT: use as an output
3. OUT*Osc1: binary AND-function of out and oscillator 1. (*)
4. OUT*Gate: binary AND-function of out and Gate.
5. OUT*Gate*Osc1: binary AND-function of out and oscillator 1 and Gate. (**)
6. OUT*(Gate*Osc1+!Gate*Osc2): when out is set, gate controlled switch between oscillator 1 and 2. (***)

(*) On a laser with analog power control this mode can be used to generate an analog voltage using an external lowpass filter or an AppLink-03 add-on.

(**) On a CO_2 laser this pin can be used to pulse width modulate the laser. Set the duty cycle of oscillator 1 to desired value. Whenever GATE is active the CO2 laser sees oscillator 1 signal.

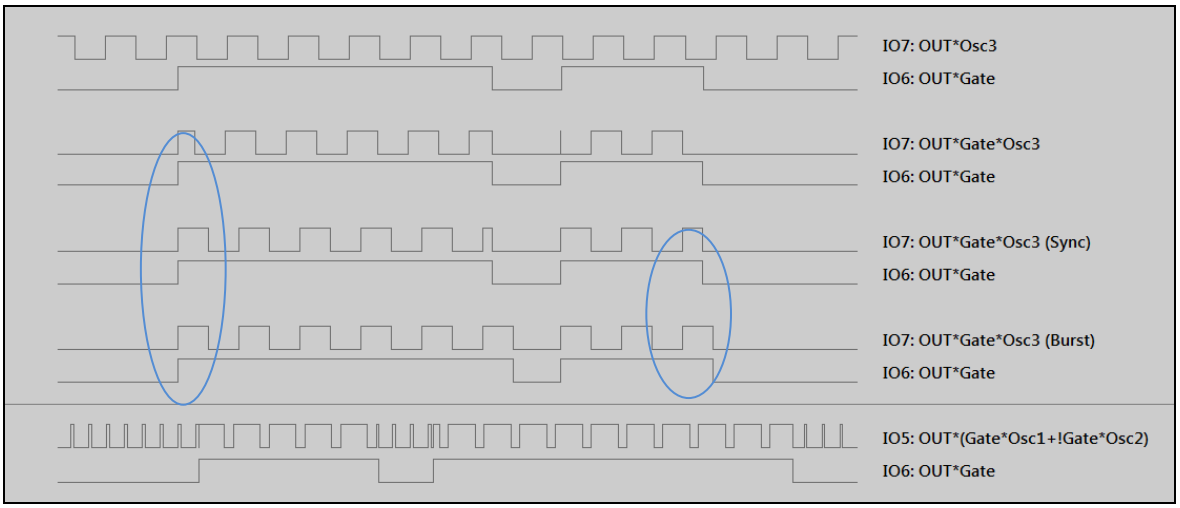
(***) On a CO_2 laser this setting can be used to switch between two pulse width modulation levels of the laser.

Selections IO6

1. IN: use as an input
2. OUT: use as an output
3. OUT*Osc2: binary AND-function of out and free running oscillator 2.
4. OUT*Gate: binary AND-function of out and GATE.
5. OUT*Gate*Osc2: binary AND-function of out and oscillator 2 and Gate.

Selections IO7

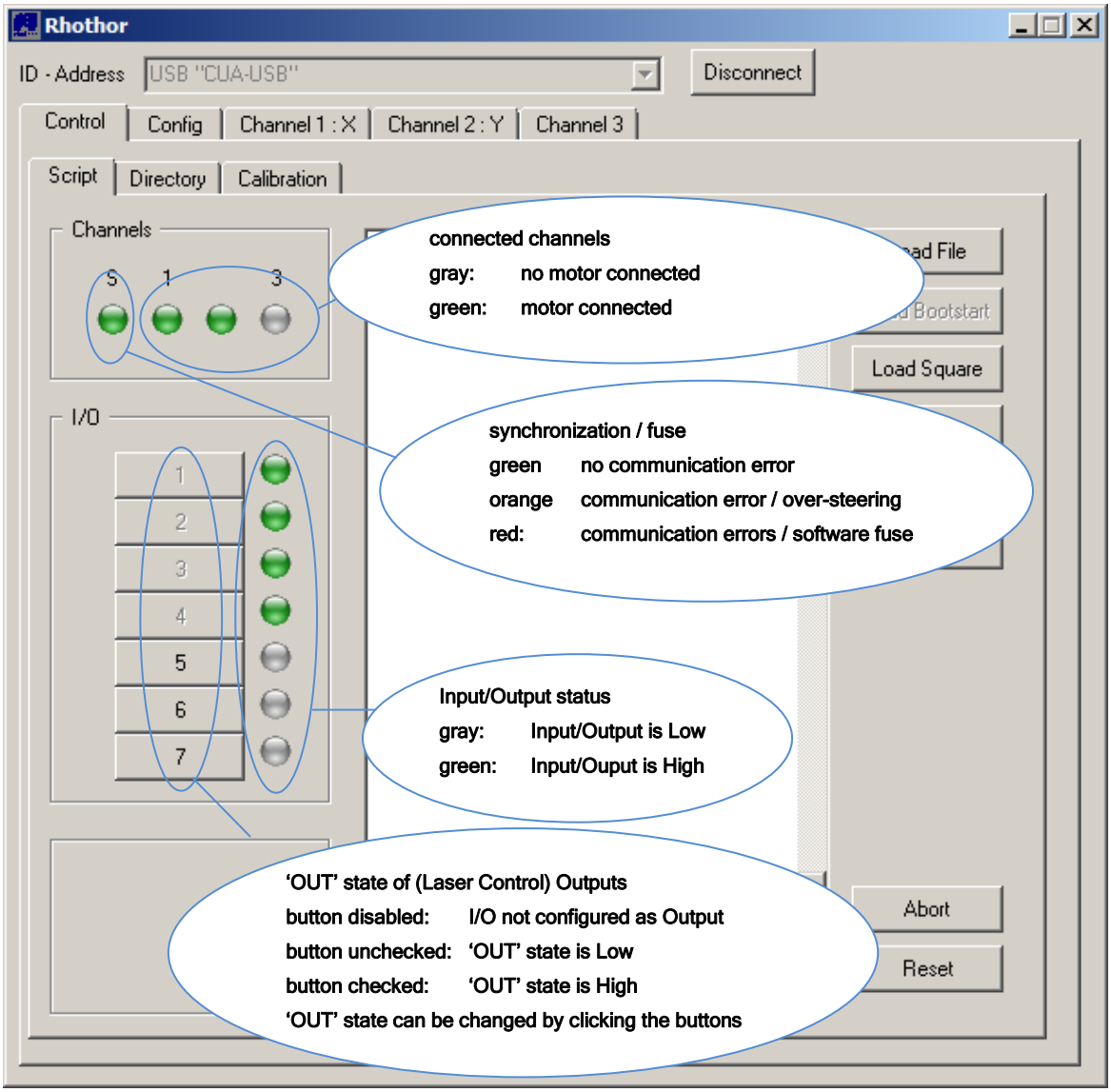
1. IN: use as an input
2. OUT: use as an output
3. OUT*Osc3: binary AND-function of out and free running oscillator 3
4. OUT*Gate: binary AND-function of out and gate
5. OUT*Gate*Osc3: binary AND-function of out and GATE and free running oscillator 3
6. OUT*Gate*Osc3 (Sync): Same as OUT*Gate*Osc3 but Oscillator 3 will be synchronised with the rising edge of gate signal. This configuration can be used to control lasers with external triggering.
7. OUT*Gate*Osc3 (Burst): Same as OUT*Gate*Osc3 but Oscillator 3 will be synchronised with the rising edge of gate signal. The gate signal will be extended so the last pulse will have the pulse width defined by oscillator 3.



4 CONTROL TAB

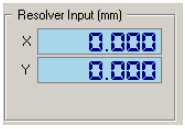
The control page contains interface tools to the user. It allows to control the deflection head by running command scripts and allows to verify the overall status of the deflection system.

4.1 STATUS



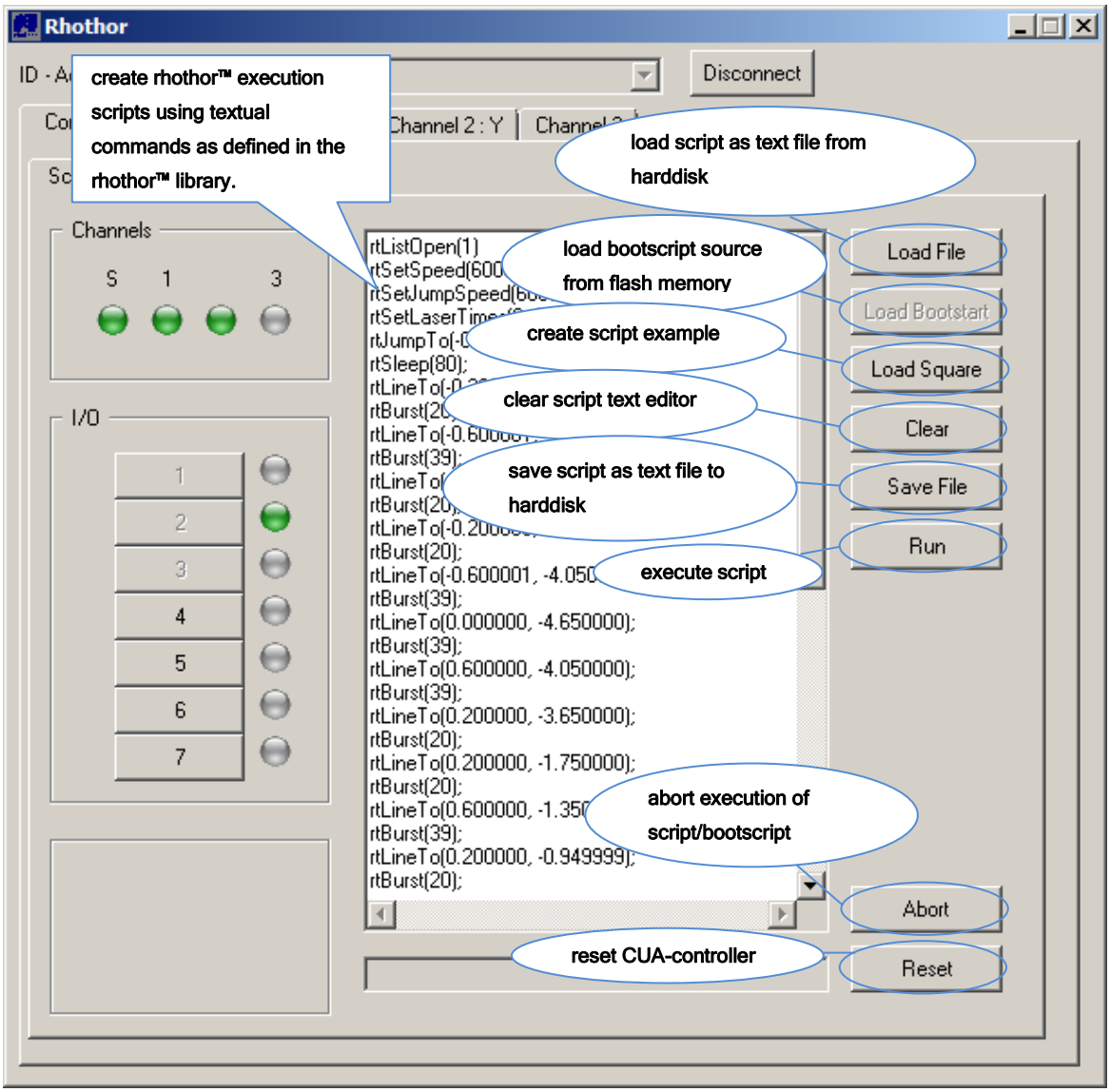
The Sync LED shows the overall status of the deflection system. This LED should be indicating everything is OK. When there are communication errors on the coax cable it blinks orange and it will turn red when the motor is shut down by the control board. The motor is shut down as precaution because of over-steering or an instable coax connection.

On the script tab the live status of all IO pins can be seen. This status is updated every 50 msec and is a nice tool to verify connections and system behaviour. When encoder function blocks are active also the current encoder positions can be seen in the left bottom of the script tab. The encoder are initialised using the "rtSetResolver" library command.



4.2 SCRIPTING

The script text editor is an easy and fast way of controlling the deflection system without the need for extra application software. A script is build up by using the list commands provided by the rhothor™ dynamic library interface.



4.2.1 Rhothor™ commands

The set of commands used in the script editor are the same as the rhothor™ software library and are defined in the manual “Applicatons for CUA-USB and CUA ETH control units”. Below a summary of most common script commands.

rtAbort	*rtEndIf*	*rtMoveTo*	*rtSetResolver*
rtArcTo	*rtFileFetch*	*rtResetResolver*	*rtSetResolverRange*
rtBurst	*rtFileOpen*	*rtSetIO*	*rtSetSpeed*
rtCircle	*rtFileClose*	*rtSetJumpSpeed*	*rtSetWobble*
rtDoLoop	*rtJumpTo*	*rtSetLaserTimes*	*rtSleep*
rtIfIO	*rtLineTo*	*rtSetLoop*	*rtWaitIO*
rtIndexFetch	*rtListClose*	*rtSetOffsXY*	*rtWhileIO*
rtElse	*rtListOpen*	*rtSetOscillator*	*rtWaitResolver*

4.2.2 Creating a script

A rhothor™ script starts with a “rtListOpen” command and ends with a “rtListClose” command. When you click the “Run” button the script in the text editor is interpreted and executed. During execution the “Run” button will be disabled. A running job can be stopped at all time by pressing the "Abort” button.

A script example can also be loaded by pressing the “Load Square” button. This execution script will mark a square using the complete field size.

Script example : marking a square with compensation delays

```
1.    rtListOpen(1)                 // start rhothor command list
2.    rtSetIO(112,112)              // enable laser IO’s 5, 6 and 7
3.    rtSetJumpSpeed(1000)          // set jumping speed
4.    rtSetSpeed(1000)              // set marking speed
5.    rtSetLaserTimes(80,80)        // implement gate shift
6.    rtJumpTo(-10,-10)             // jump to position
7.    rtSleep(80)                   // implement jump delay
8.    rtLineTo(10,-10)              // mark a line
9.    rtBurst(80)                   // implement polygon delay
10.   rtLineTo(10,10)               // mark a line
11.   rtBurst(80)                   // implement polygon delay
12.   rtLineTo(-10,10)              // mark a line
13.   rtBurst(80)                   // implement polygon delay
14.   rtLineTo(-10,-10)             // mark a line
15.   rtSleep(80)                   // implement mark delay
16.   rtListClose()                 // end rhothor command list
```

4.2.3 Creating a flash job

CUA control card have a flash memory which allows to store different marking jobs on the flash. The flash contains 250 sectors of 256 bytes. Flash jobs can be created by starting the script with a “rtFileOpen” command and end it with a “rtFileClose” command.

Script example : creating a job to store in flash

```
1.  rtFileOpen(FlashJob1)           // create rhothor flash job “FlashJob1”
2.  rtSetJumpSpeed(1000)            // set jumping speed
3.  rtSetSpeed(1000)                // set marking speed
4.  rtSetLaserTimes(80,80)          // implement gate shift
5.  rtJumpTo(0,0)                   // jump to position
6.  rtLineTo(10,4)                  // mark a link
y.  …                               // #n rhothor commands
z.  rtFileClose()                   // end of rhothor flash job
```

When the flash job is created it will appear in the flash explorer and the index (the job’s first sector) can be used to fetch the job from flash.

Script example : fetching a Flash Job

```
1.  rtListOpen(1)                   // start rhothor command list
2.  rtSetIO(112,112)                // enable laser IO’s 5, 6 and 7
3.  rtIndexFetch(1)                 // execute flash job located on flash sector 1
4.  rtListClose()                   // end rhothor command list
```

4.2.4 Creating a bootscript

When the system is operated without central application computer, it must be able to start list processing automatically. Control functions can be stored to flash under the bootsector. When the deflection control system is powered on, it scans the flash if this bootsector is filled in with a bootscript. When the file is found, its execution list is started automatically. This can also be useful for having a machine boot-up sequence.

A boot script is created by creating a list execution script using list mode 3. This execution script will then be stored on the bootsector. A boot script may contain conditional functions like rtIfIO, rtElse and rtEndIf or iteration functions like "rtSetLoop" and "rtDoLoop" in order to create a loop sequence.

For CUA-ETH controllers there is also a set of Ethernet commands which allow to control the deflection system remotely from a PLC. Combining the Ethernet command set with a dedicated bootscript allows creating a complete stand-alone process.

Boot script example : job selection and job trigger from PLC

```
1.  rtListOpen(3)                   // start rhothor boot script
2.  rtSetIO(64,64)                  // enable laser IO7
3.  rtSetLoop(0)                    // endless loop start
4.  rtWaitIO(4,4)                   // wait IO3 HIGH (IO3 – Input – job_trigger)
5.  rtSetIO(8,8)                    // set IO4 HIGH (IO4 – Output – job_in_progress)
6.  rtIndexFetch(0)                 // fetch job index 0 => selected from PLC
7.  rtWaitIO(0,4)                   // wait IO3 LOW (job_trigger)
8.  rtSetIO(0,8)                    // Set IO4 LOW (job_in_progress)
9.  rtDoLoop()                      // endless loop end
10. rtListClose()                   // end of script
```

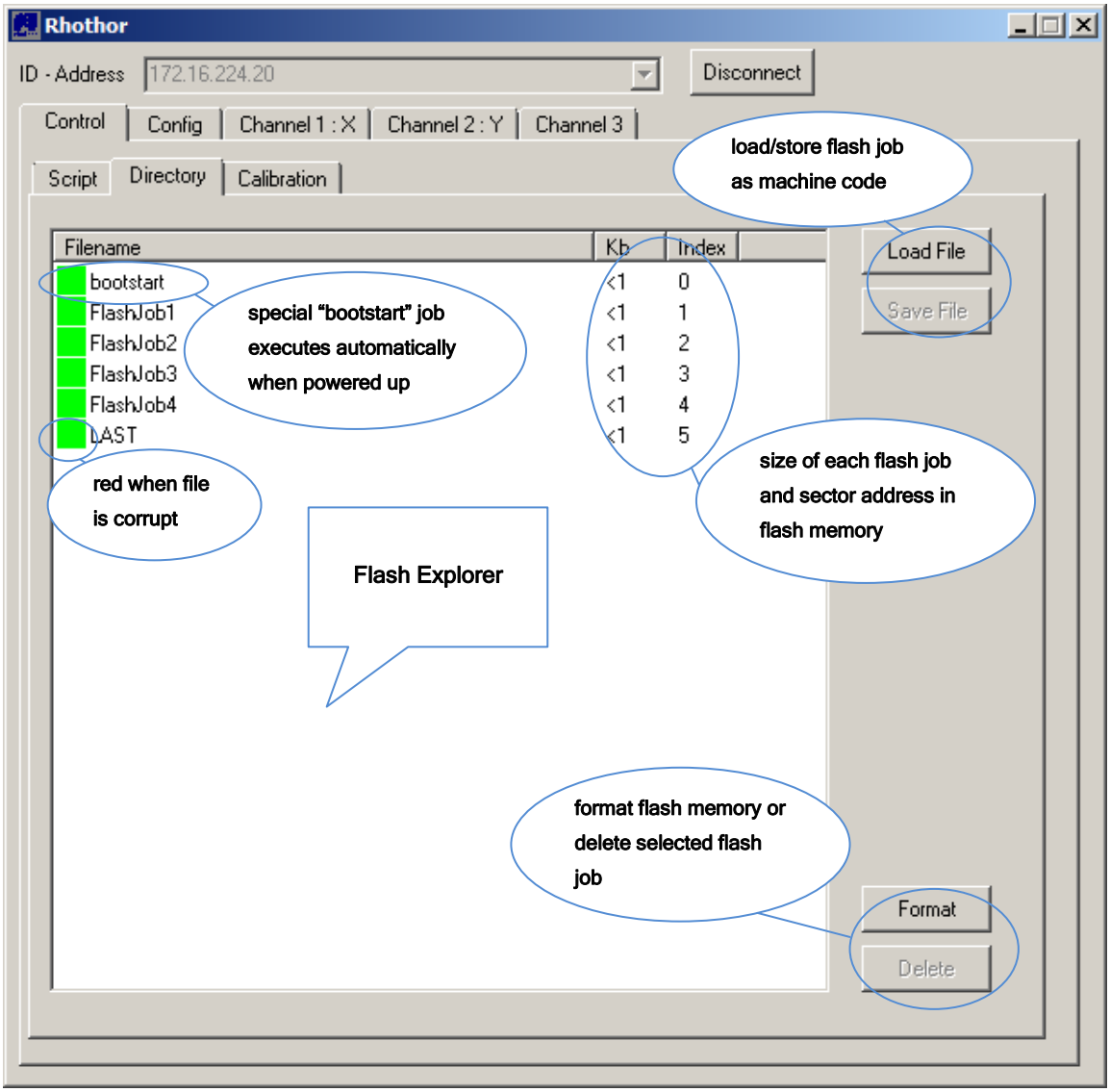
PLC using direct Ethernet commands (see also manual "CUA Ethernet commands")

```
1.  Suspend bootscript [GO_SUSPENDED]
2.  Buildup job list from flash sectors [QUERY_INDEX]
3.  Restart bootscript [RUN_BOOTSTART]
4.  Set job_trigger LOW
5.  Select current job index [SELECT_IMAGE]
6.  Set job_trigger HIGH
7.  Wait job_in_progress HIGH
8.  Set job_trigger LOW
9.  Wait job_in_progress LOW
10. IF other_job then GOTO 5. ELSE GOTO 7.
```

When the bootscript is created with rhothor.exe, the source code will also be stored into the flash memory. This source code can be reloaded into the script editor from the flash by pressing the "Load Bootstart" button.

A bootscript file aswell as a flash job can be deleted from flash memory on the Directory tab page.

4.3 DIRECTORY



In the flash explorer you see all jobs created on the flash memory, including when a boot script is available. The index of each flash job is the start sector of the flash job. This index is used to execute a certain job. Besides the index also the flash file size is shown.

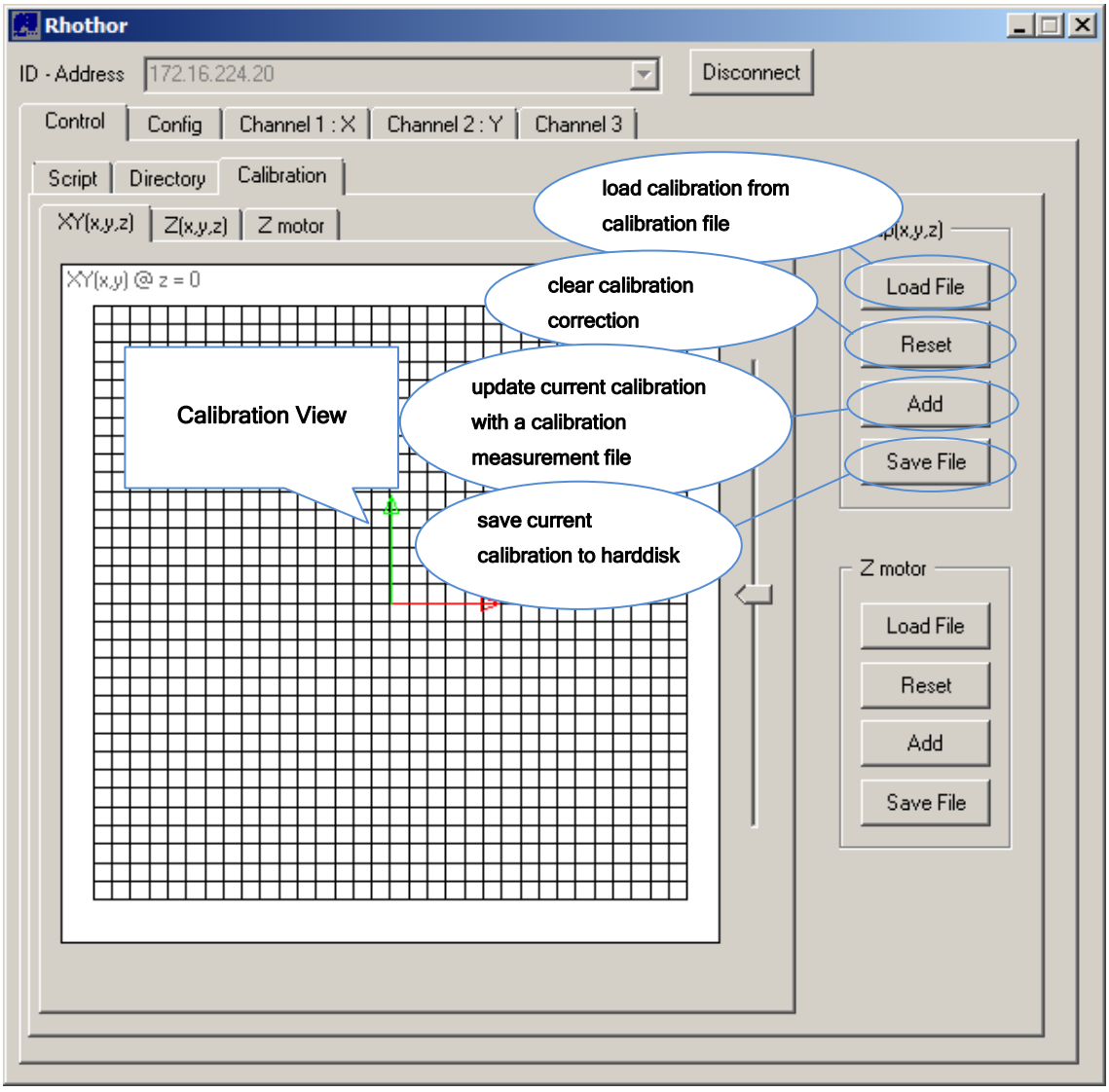
4.3.1 Delete /Format

The flash memory can be completely formatted by clicking the "Format" button. All data will be erased from the flash. A single Flash job can be selected and deleted from the flash by clicking the "Delete" button.

4.3.2 Load /Save File

When the rhothor™ software creates a flash job, it compiles the rhothor™ commands into machine code. Machine code is a block of instructions which can be directly executed by the CUA control board. It is possible to store this block of machine code on the hard disk or to load this instruction block from a file straight into the memory of the flash. Because these machine code files are compiled for a certain CUA controller, they are only interchangeable with other CUA control boards if they contain the same settings. (same fieldsize, setpoint filter and track delay)

4.4 CALBIRATION



rhothor™ deflection systems use lookup tables to handle projection distortions. The calibration lookup table is stored inside the flash memory of the CUA control board.

4.4.1 Reset

Pressing button "Reset" will reset the calibration currently loaded in the system.

4.4.2 Add

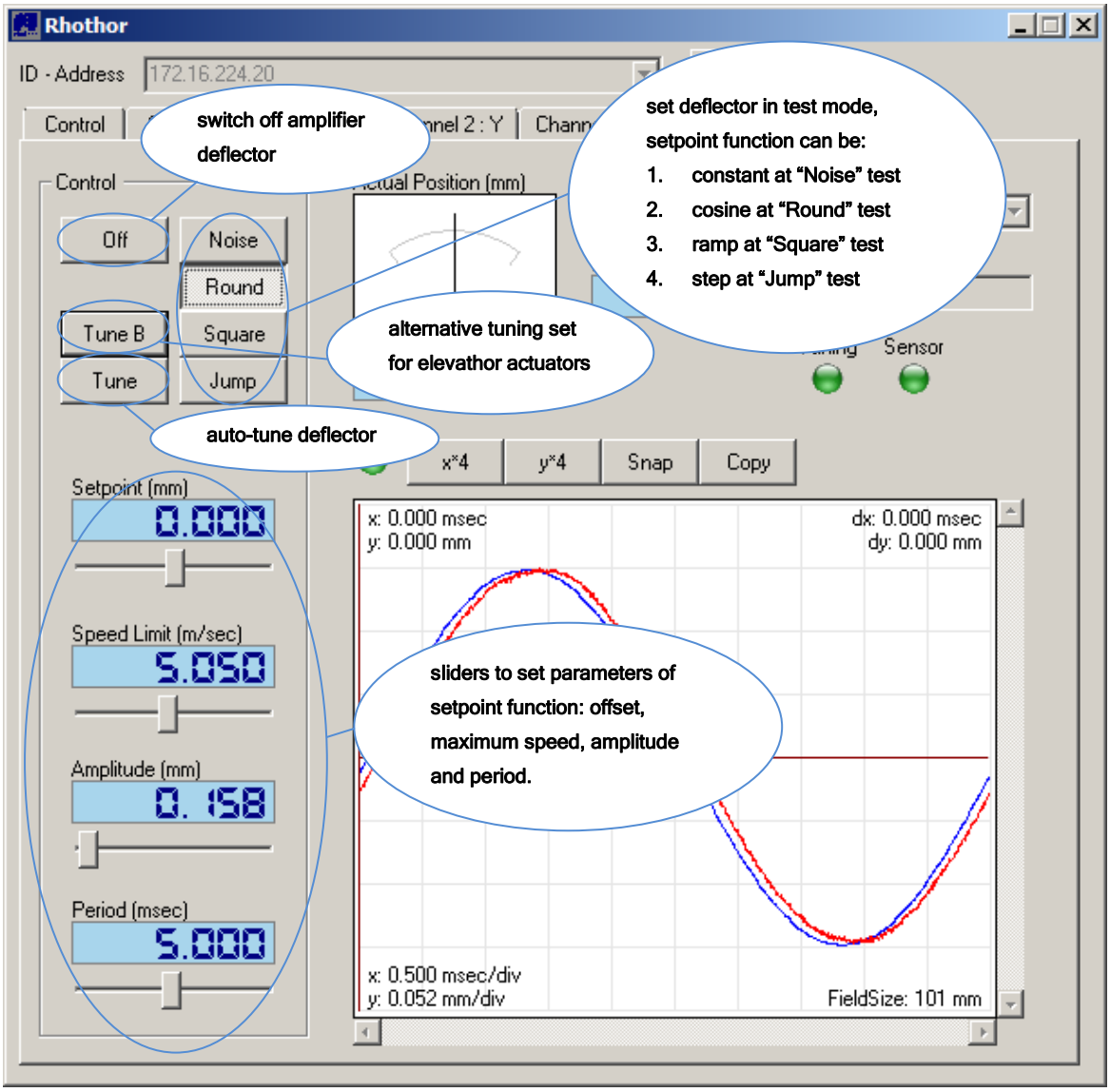
By nature calibration differs from system to system. Therefor errors need to be measured and compensation needs to be adjusted. This can be done by lasering a raster of crosses and measure their positions. The number and position of those crosses is a function of the desired accuracy. The result of those measurements has to be stored in a data file. This data file can be added to the current calibration table by pressing the button “Add”. Data files need to be generated by application software. For more information on the file format of the data see manual “A2G_App”.

4.4.3 Load /Save File

The calibration lookup table can be stored to a file by pressing the button “Save File”. Pressing the button "Load File" will pre-set the calibration table with file data. In general the calibration file is a file generated by the system itself.

5 CHANNEL 1, CHANNEL2, CHANNEL3 TAB

5.1 CHANNEL CONTROL

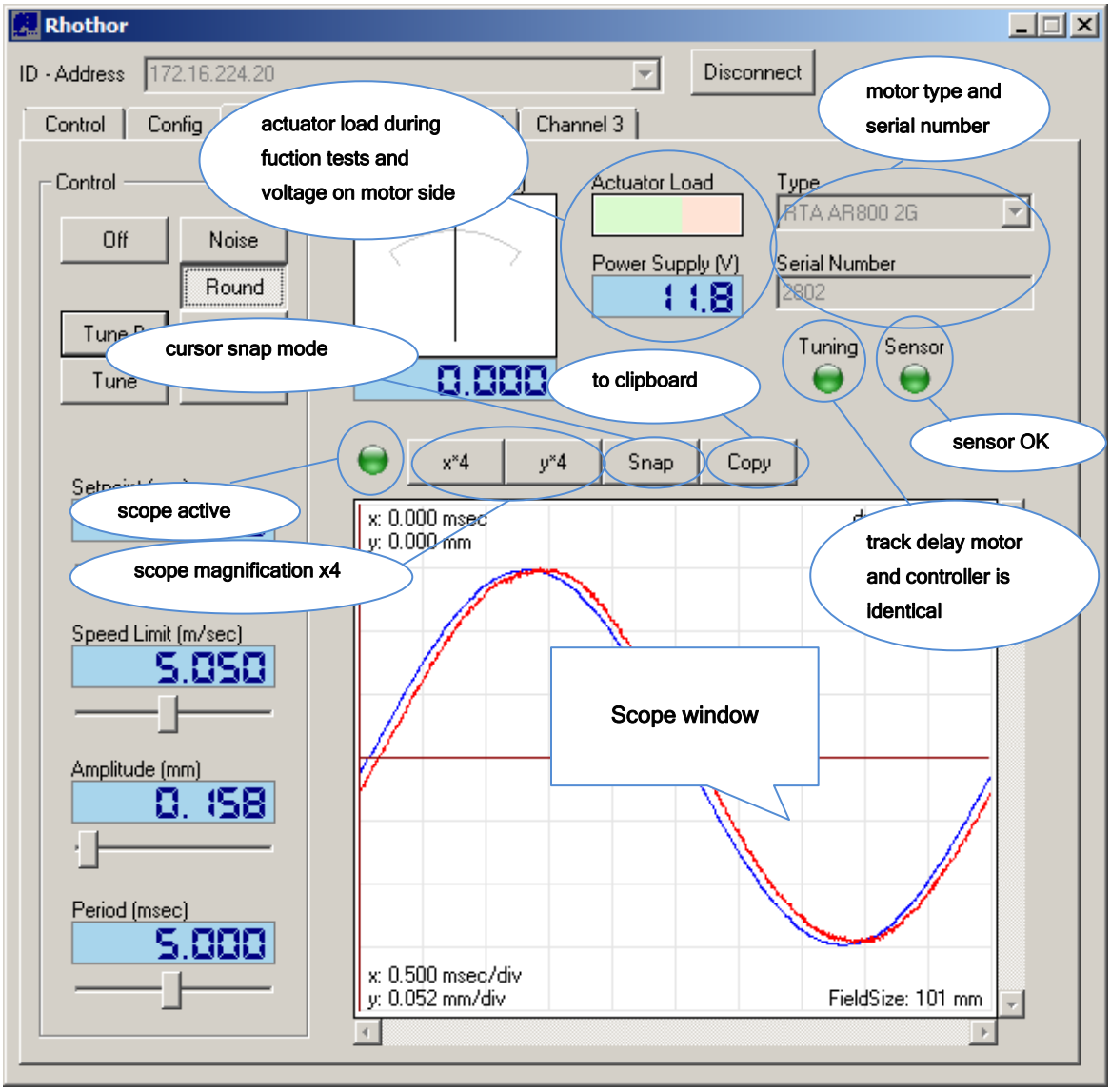


After setting the properties on the "Config" tab and tuning the deflector, its behaviour can be tested. The deflector can be set-up to execute periodically jumps, rounds and squares. An oscilloscope provides live graphical presentation of setpoint (green line) and actual (red line) positions.

The rhothor™ deflection technology supports settable dynamics. The track delay can be set and regulators can be auto tuned by pressing the button "Tune". However there is a relation between system noise, track delay and setpoint value. Use the noise analyse test if the required track delay is low. This noise test should be done at all boundary points of the fieldsize.

Speeds can be clipped by changing the speed limit. This is useful when for instance simulating processing speeds in jump function tests.

5.2 CHANNEL STATUS



The channel tab contains some status items to see the motor status:

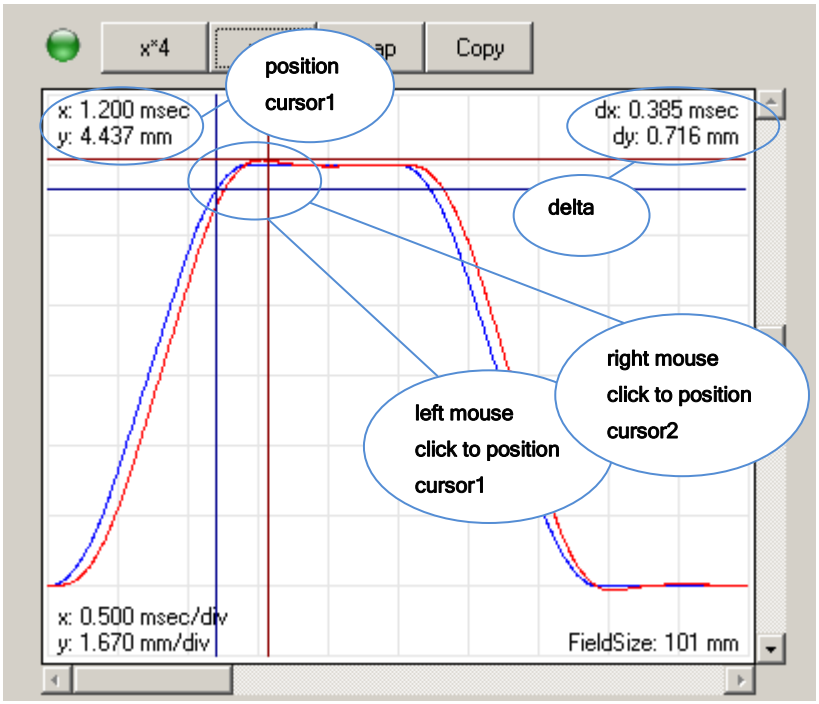
- An actuator Load bar will show when motor is over steered when performing function test.
- Indication of the voltage received by the power supply.
- A green Sensor LED indicates the internal optical sensor is working fine. If this LED turns yellow or red get in contact with Newson.
- A green Tuning LED indicates motor is tuned for the correct settings. Whenever LED is yellow motor needs to be returned by pressing the “Tune” button.
- A position indicator shows the current motor position.
- Motor type and serial can be read out
- An integrated scope window to analyse behaviour of the motor to several function tests. A green LED indicates when the scope is active.

5.2.1 Scope window

An integrated scope window allows to analyse the motor behaviour during function tests. (round, square or jump tests) The blue line shows the setpoint stream send to the motor and the red line show the actual position stream, coming from the feedback of the motor. The time shift between actual positions and setpoint stream is the defined track delay of the system.

It is possible to activate two cursors to allow measuring on the scope window.

Activate two cursors with "Snap Off":



Activate cursors with "Snap On":