Kawasaki Robotics

ASTORINO

Operation Manual

Preface

This manual describes the handling of the 6-axis robot “astorino" and the associated "astorino" software.

The ASTORINO is a learning robot specially developed for educational institutions. Pupils and students can use the ASTORINO to learn robot-assisted automation of industrial processes in practice.

This manual is valid from firmware version 3.8.1 and astorino software version 1.8.9

1. The "astorino" software included with the ASTORINO is licensed for use with this robot only and may not be used, copied or distributed in any other environment.
2. Kawasaki shall not be liable for any accidents, damages, and/or problems caused by improper use of the ASTORINO robot.
3. Kawasaki reserves the right to change, revise, or update this manual without prior notice.
4. This manual may not be reprinted or copied in whole or in part without prior written permission from Kawasaki.
5. Keep this manual in a safe place and within easy reach so that it can be used at any time. If the manual is lost or seriously damaged, contact Kawasaki.

Symbols

Items that require special attention in this manual are marked with the following symbols.

Ensure proper operation of the robot and prevent injury or property damage by following the safety instructions in the boxes with these symbols.

⚠ WARNING

Failure to observe the specified contents could possibly result in injury or, in the worst case, death.

[**ATTENTION**]

Identifies precautions regarding robot specifications, handling, teaching, operation, and maintenance.

⚠ WARNING

1. **The accuracy and effectiveness of the diagrams, procedures and explanations in this manual cannot be confirmed with absolute certainty. Should any unexplained problems occur, contact Kawasaki Robotics GmbH at the above address.**
2. **To ensure that all work is performed safely, read and understand this manual. In addition, refer to all applicable laws, regulations, and related materials, as well as the safety statements described in each chapter. Prepare appropriate safety measures and procedures for actual work.**

Paraphrases

The following formatting rules are used in this manual:

- For a particular keystroke, the respective key is enclosed in angle brackets, e.g. <F1> or <Enter>.
- For the button of a dialog box or the toolbar, the button name is enclosed in square brackets, e.g. [Ok] or [Reset].
- Selectable fields are marked with a square box ☐.
  If selected a check mark is shown inside the symbol ☑.

ASTORINO Operation Manual

List of contents

1 Nomenclature in this manual

The author of the manual tries to use generally valid terminology while achieving the greatest possible logical sense. Unfortunately, it must be noted that the terminology is reversed depending on the point of view when considering one and the same topic. Also it is to be stated that in the course of the computer and software history terminologies developed in different way. One will find therefore in a modern manual no terminologies, which always satisfy 100% each expert opinion.

2 Overview of ASTORINO

The ASTORINO is a 6-axis learning robot developed specifically for educational institutions such as schools and universities. The robot design is based to be 3D printed with PET-G filament. Damaged parts can be reproduced by the user using a compatible 3D printer.

Programming and control of the robot is done by the "astorino" software.

The latest software version and 3D files can be downloaded from the KAWASAKI ROBOTICS FTP server:

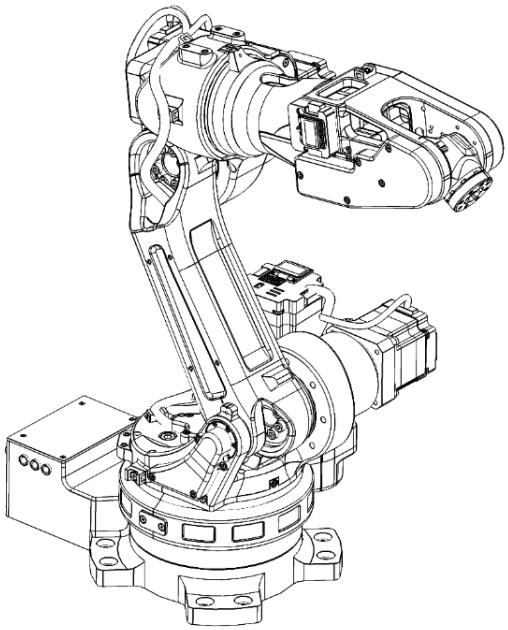
https://ftp.kawasakirobot.de/Software/Astorino/

Just like Kawasaki's industrial Robots the ASTORINO is programmed using AS language. Providing transferable programing skills from the classroom to real industrial applications.

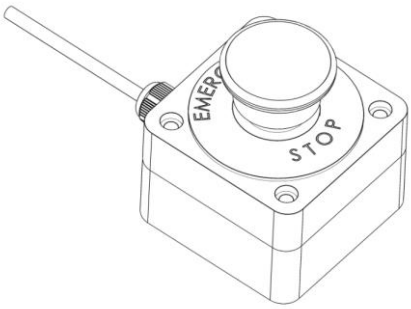
3 Technical specifications

Characteristics		ASTORINO
Type		6-axis robot
Max. lifting capacity		1 kg
Number of axes		6
Max. range		578 mm
Repeatability		±0.1 mm
Motion range	Axis 1 (JT1)	±158°
	Axis 2 (JT2)	-90°÷127°
	Axis 3 (JT3)	0°÷168°
	Axis 4 (JT4)	±240°
	Axis 5 (JT5)	±120°
	Axis 6 (JT6)	±360°
Max. single axis speed	Axis 1 (JT1)	38°/s
	Axis 2 (JT2)	26°/s
	Axis 3 (JT3)	26°/s
	Axis 4 (JT4)	67.5°/s
	Axis 5 (JT5)	67.5°/s
	Axis 6 (JT6)	128.5°/s
Allowable moment	Axis 4 (JT4)	6.2 Nm
	Axis 5 (JT5)	1.45 Nm
	Axis 6 (JT6)	1.1 Nm
Working environment	Temperature	0–40°C
	Humidity	35–80%
Controller		Teensy 4.1
Inputs/Outputs		8/8 (PNP 8 mA, NPN 15 mA)
		2/2 (24V PNP on the JT3)
Max. current consumption		144 W
Power supply		100–240 V, 50–60 Hz
Weight		12 kg
Mounting position		Floor
Material		PET-G
Colour		Black
Communication		MODBUS TCP, TCP/IP, UDP, SERIAL
Collision detection		Accelerometer
Power loss safety		Brakes on JT2 and JT3
Options	24V I/O-module	8 × Inputs / Outputs
	7^{th} axis	Linear Track
	Vision system	OpenMV
	Belt tracking	Max. 2 Encoder

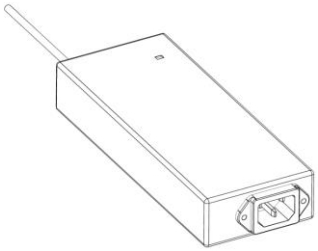
4 Robot package contents



astorino Robot

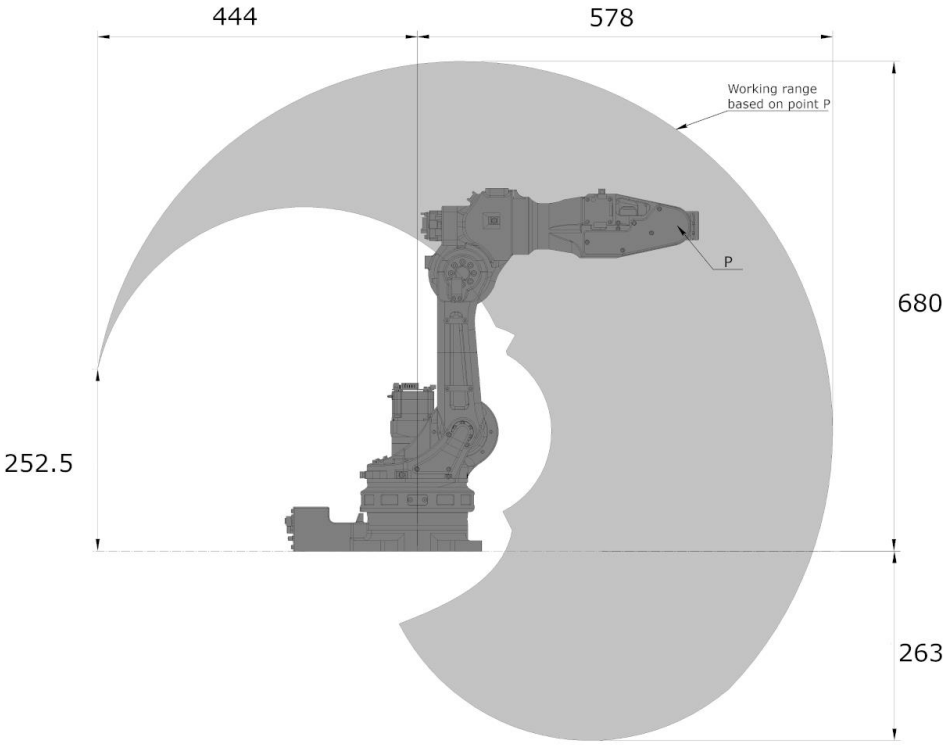


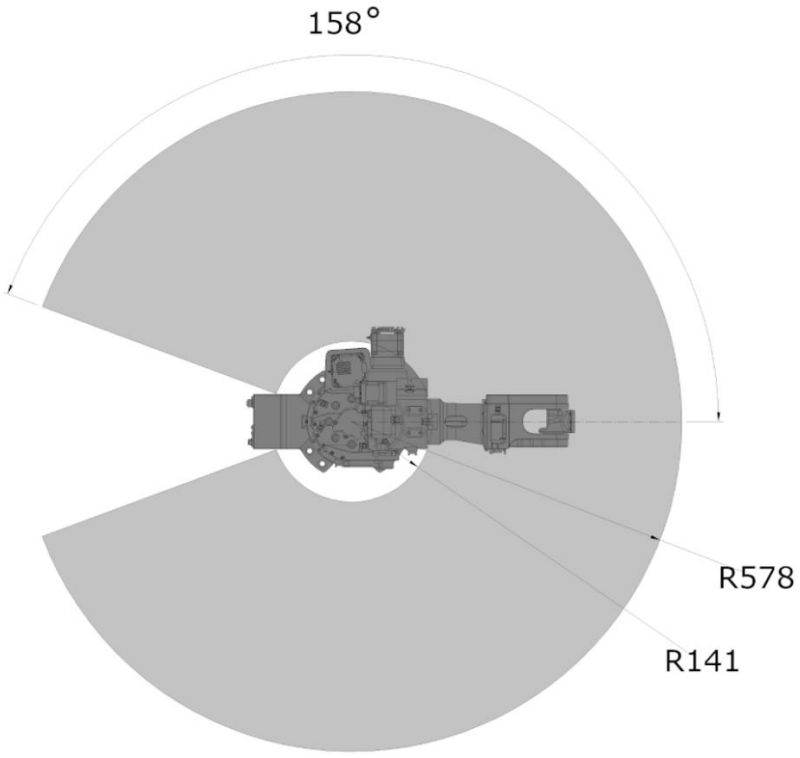
External emergency stop
in pushbutton housing



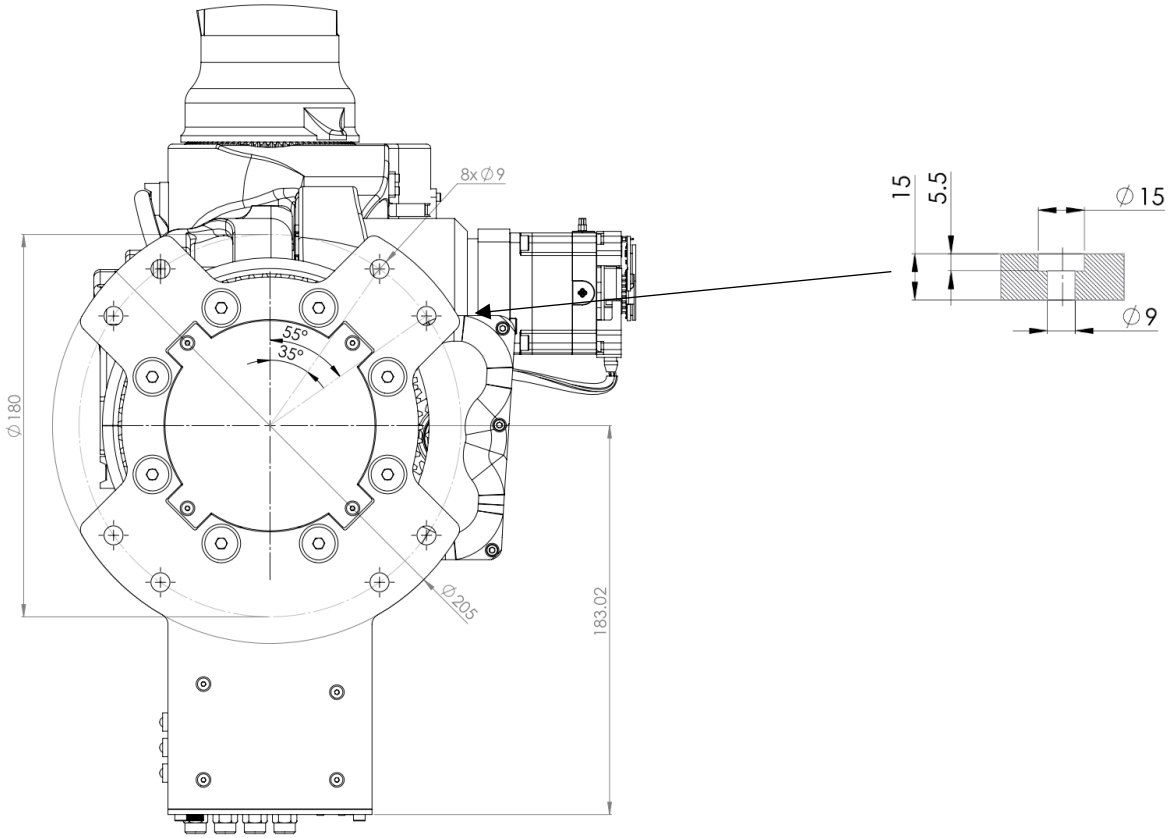
24V/DC power supply, USB cable
and USB stick

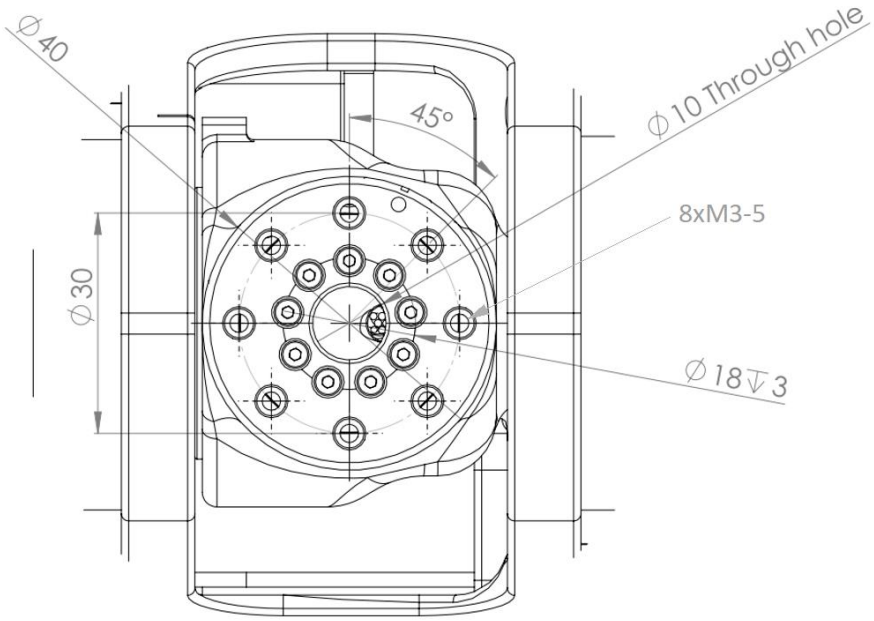
5 Range of motion



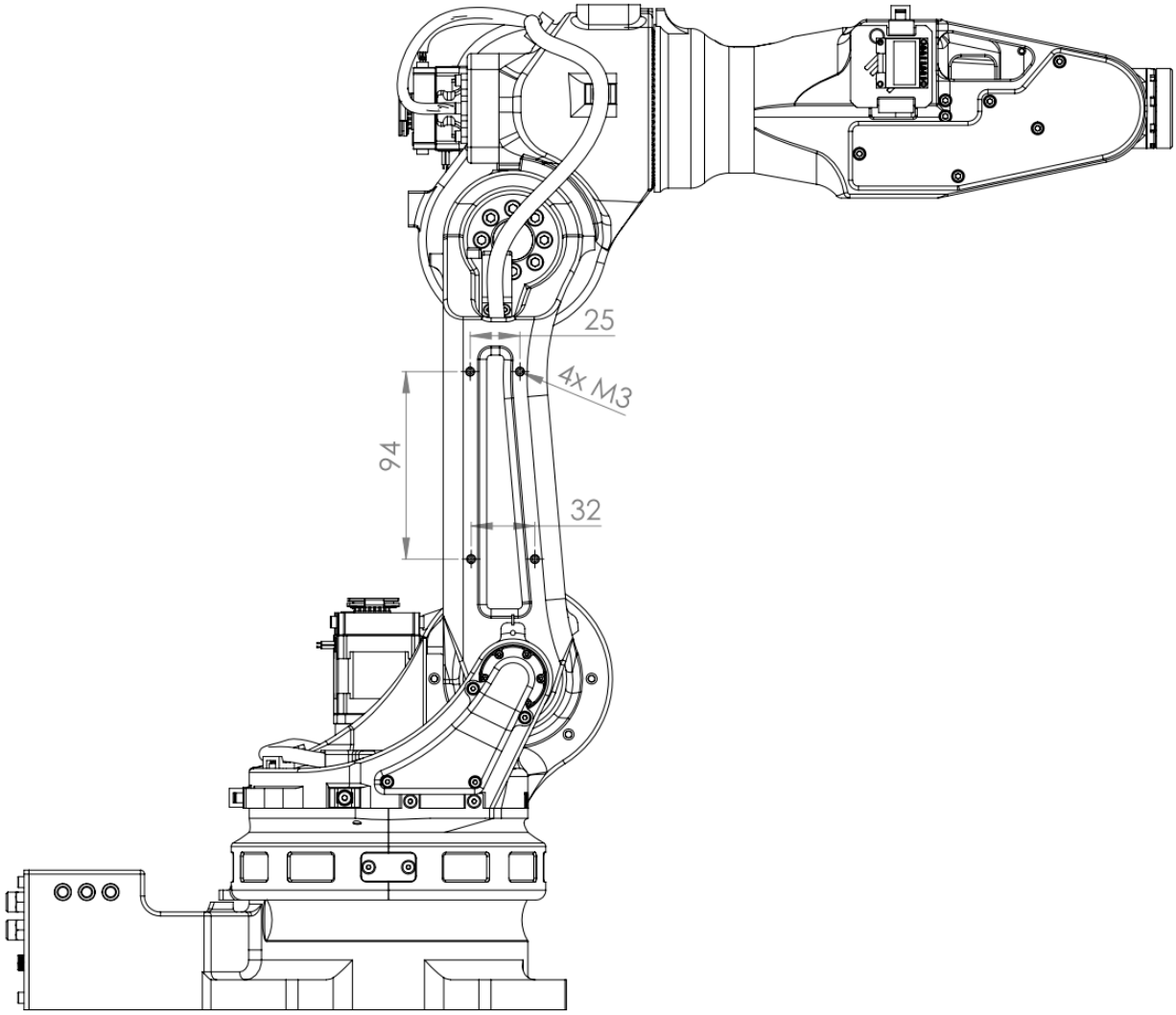


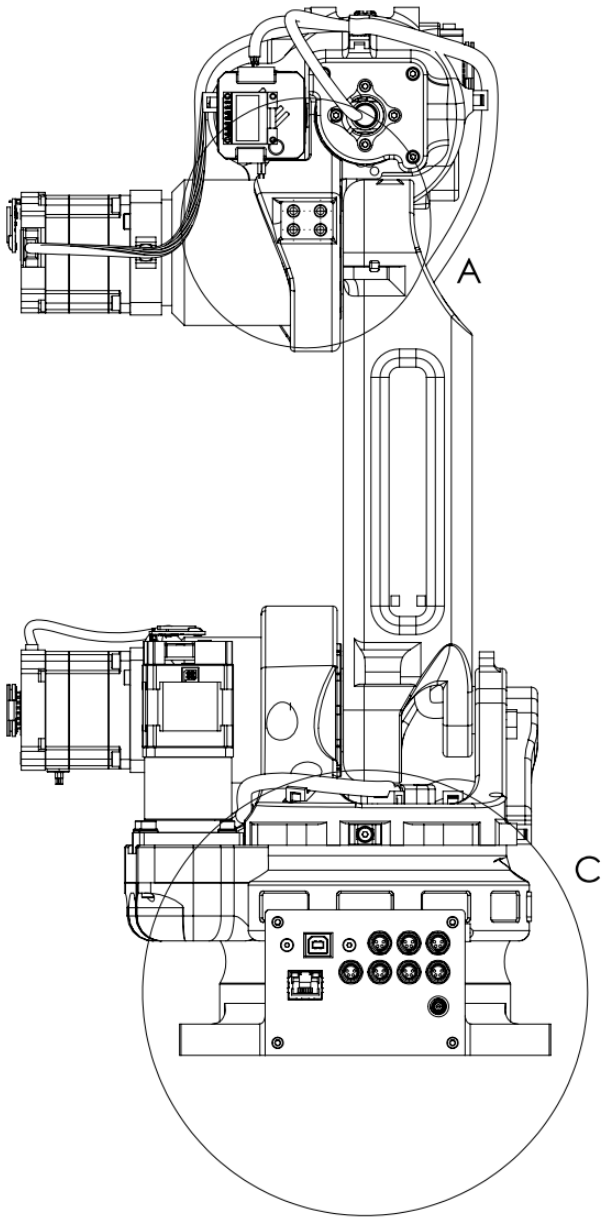
6 Mounting dimensions

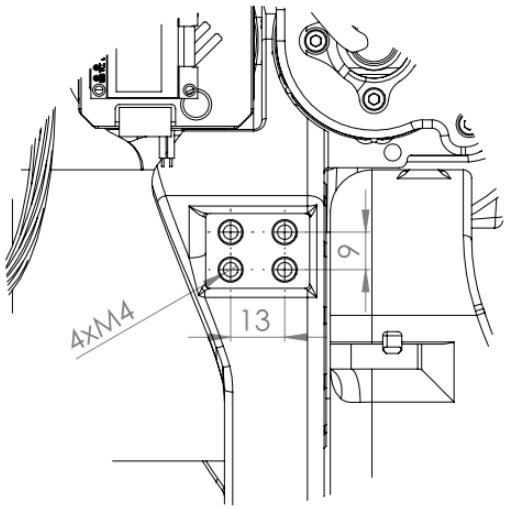




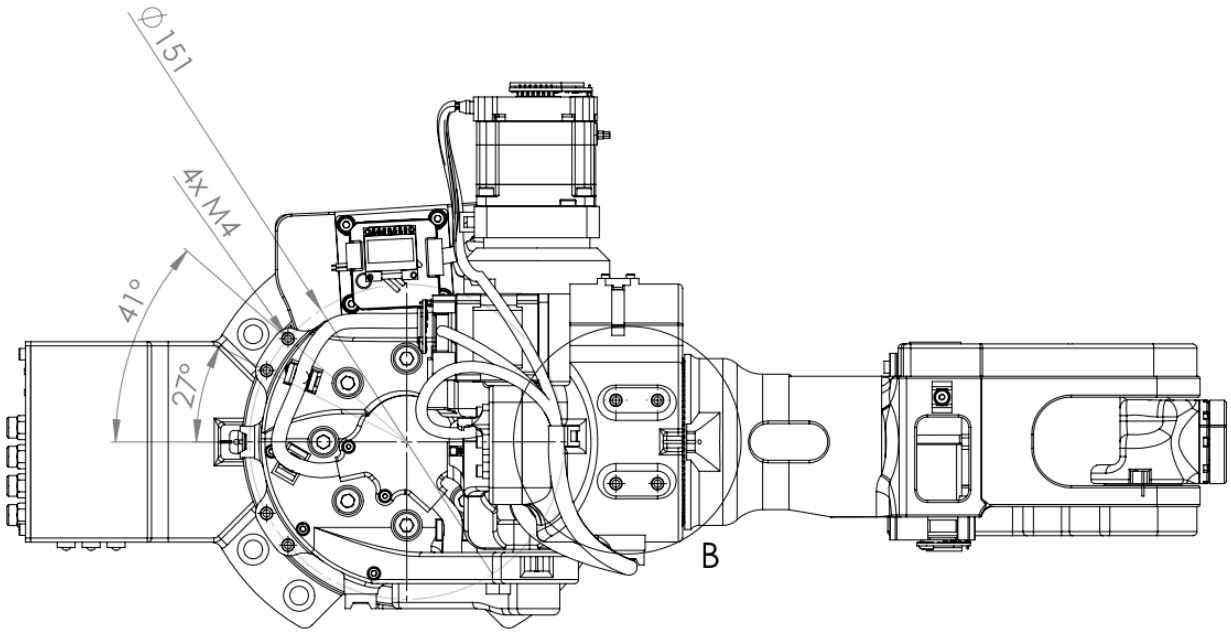
7 Installation points for accessories

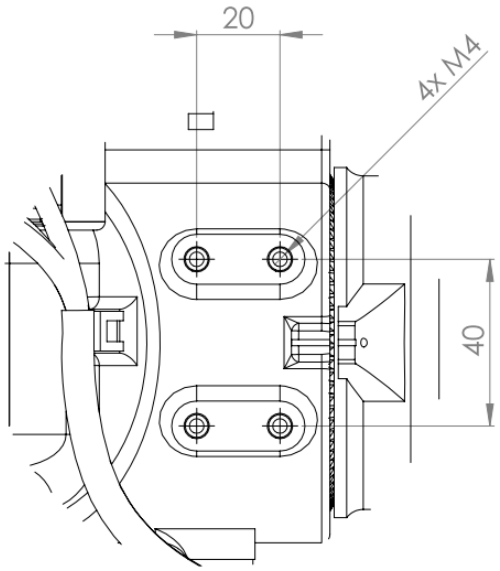




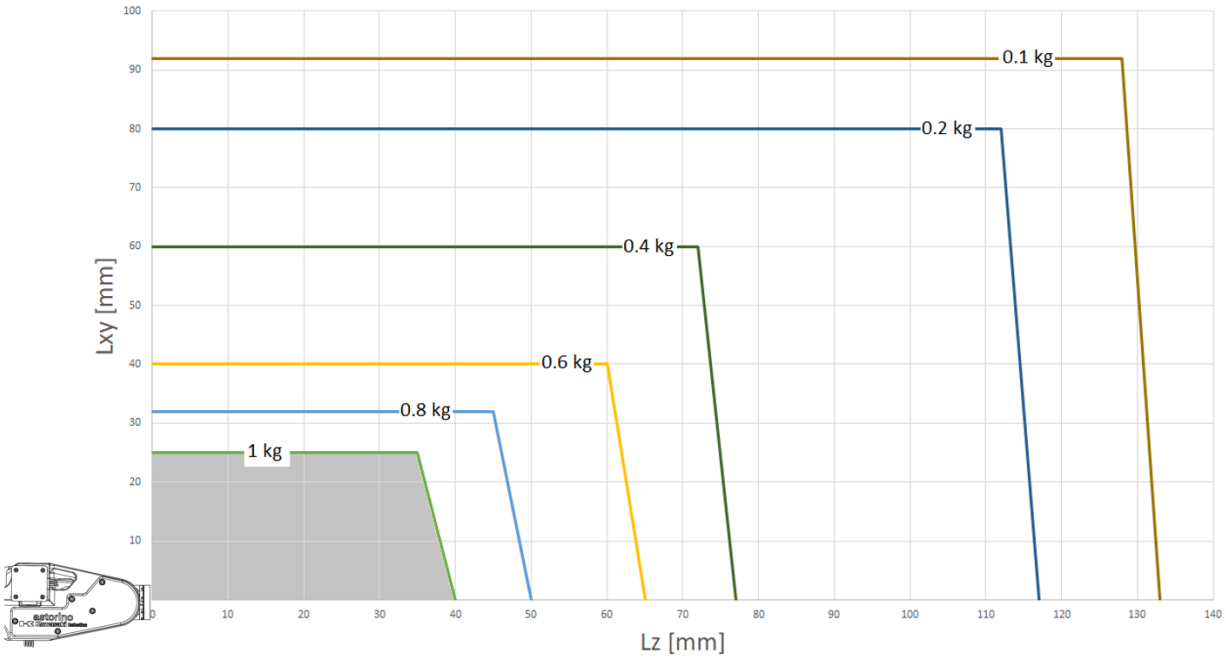


A

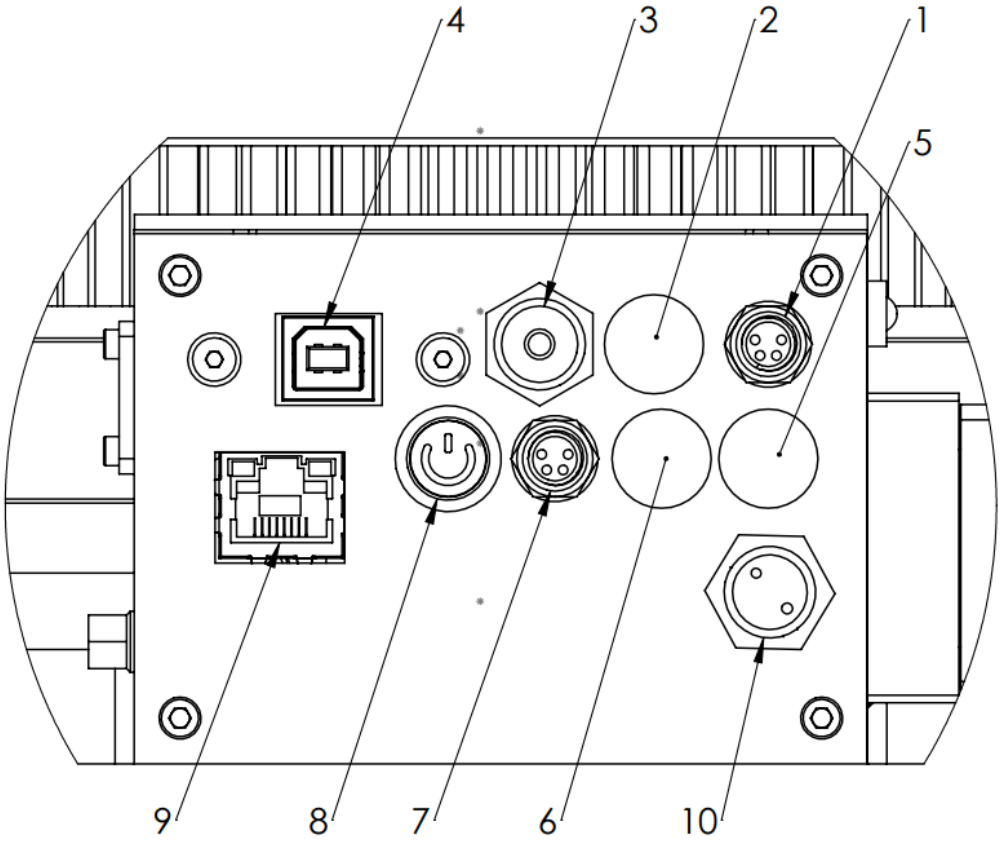

Ø151
4x M4
41°
27°
B


20
4x M4
40

8 Payload chart



9 Electrical connections



1.	M8 socket 4-Pin – external emergency stop (E-Stop)
2.	*Safety Fence (OPTION)*
3.	*Pressure inlet Ø4.0 mm*
4.	USB-B port
5.	*OPTION 2 (Encoder 2 – Conveyor 2/JT7)*
6.	*OPTION 1 (Encoder 1 – Conveyor 1)*
7.	*Vision-System/Serial-Communication*
8.	Power ON/OFF switch
9.	Ethernet port (RJ45)
10.	Power supply

10 Safety notes

[**ATTENTION**]

Always pay attention to the personal safety of the user and other persons when operating the robot arm!

- In its basic version, the robot does not have any safety-relevant components for the robot workstation. Depending on the target application, such components may be required. The basic version of the robot is equipped with an external emergency stop button (8), which must be connected before the first start-up!
- CE marking: If the robot arm is used in a factory, it must undergo a risk assessment and comply with the applicable safety regulations to ensure the safety of persons. Depending on the outcome of the assessment, other safety features should be integrated. These typically include safety relays and door switches. The commissioning engineer is responsible for this. For educational applications, no additional safety components are required.
- The robot controller contains a 24V power supply unit that must be supplied with mains voltage (100/230V). Observe the label on the power supply unit. Only qualified personnel may connect the power supply unit to the mains and put it into operation.
- Work on the electronic components of the robot should only be carried out by qualified personnel. Observe the applicable guidelines for electrostatic discharge (ESD).
- Always disconnect the robot from the power supply (100/230V) when working on the robot base or on electronic components connected to the robot controller.
- Hot plugging is prohibited! This could cause permanent damage to the motor modules. Do not install or remove any modules or connectors (e.g. emergency stop buttons, DIO modules, motor connectors) while the power supply is switched on.
- The robot arm must stand on a stable surface and be screwed down or otherwise secured
- Use and store the system only in a dry and clean place.
  The recommended room temperature is 15° to 32°C.

11 Unboxing and starting-up

Once the robot is removed from the packaging, secure it to a solid surface.

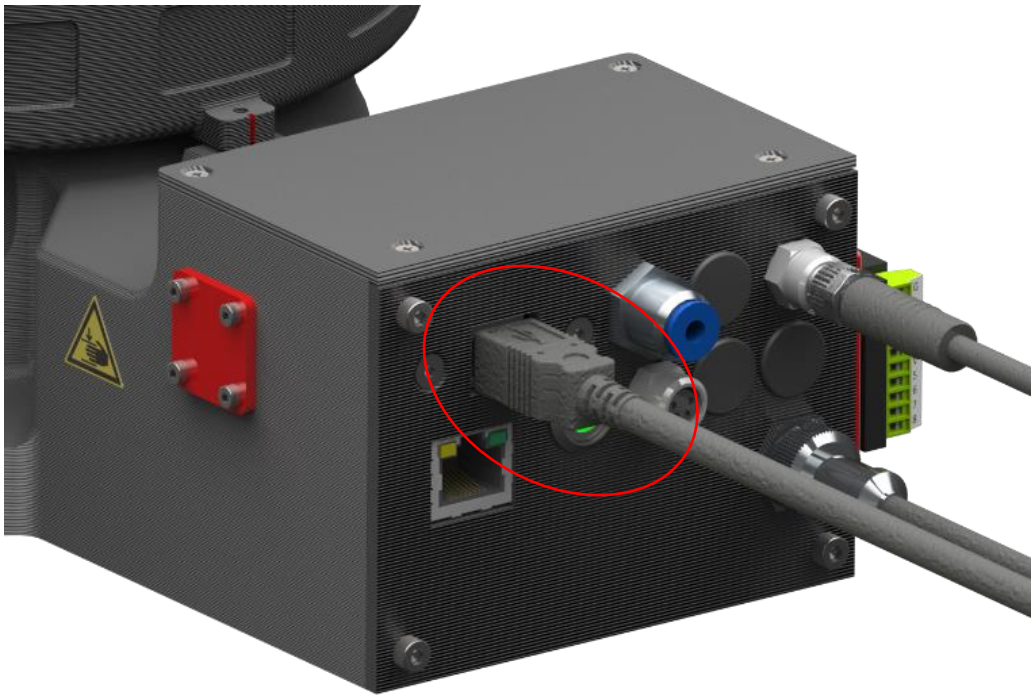
11.1 Connecting accessories

- Mount the robot on a suitable base, table or metal plate.
  The robot without gripper or other accessories independently performs basic operations such as zeroing and teach movements in the immediate vicinity of the robot base.

- Connect the power supply unit and the external emergency stop button to the connections on the robot base.

- Turn on the robot by pressing the illuminated button.

- Connect the USB cable to the USB-B port of the robot base, then connect it to a computer.



11.2 System Requirements

Before installing astorino software, ensure that the computer meets the following hardware and software requirements.

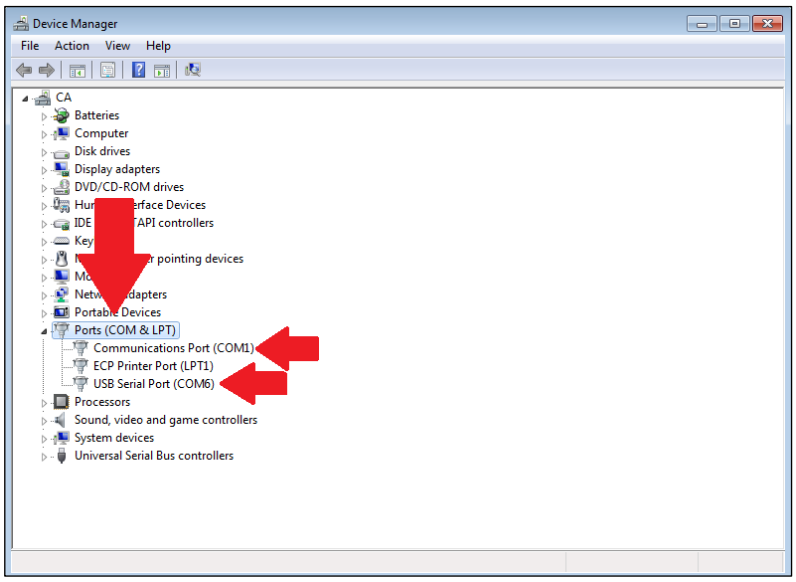
Part	Requirements
CPU	2.0 Ghz or faster processor
Memory	4 GB minimum
Disk	100 MB free space
Graphics card	Any
Display settings	1280 x 720 pixels minimum resolution, 100 % display scaling recomended
Mouse	Three-button mouse

System	Version
Windows	7, 8, 8.1, 10, 11

11.3 Driver installation

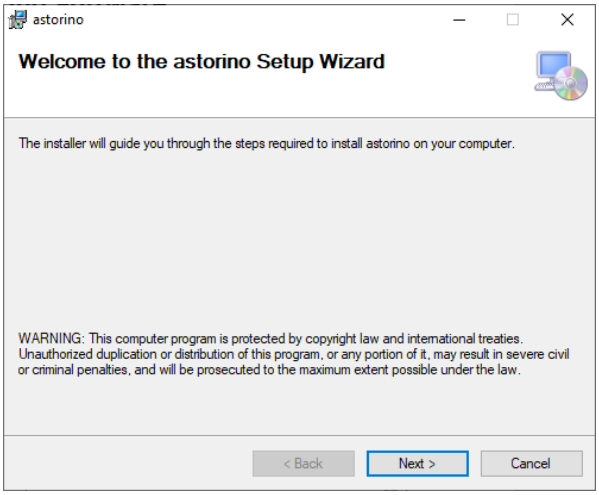
The required drivers install automatically since Windows 8. After successful installation, the robot will appear in the Device Manager as <USB Controller>. If using Windows 7 install the drivers before connecting the robot to the PC (downloaded from Kawasaki FTP server or from USB stick).

Open device manager via <**Windows + R**> ⇨ devmgmt.msc
or by clicking the icon in the selection menu via <**Windows + X**>.

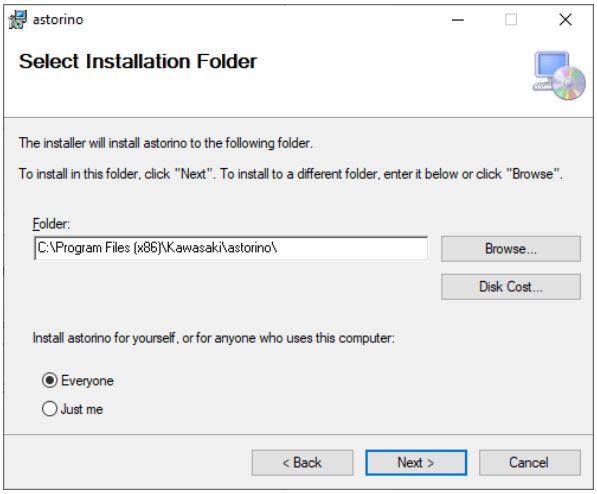


11.4 Installing the astorino Software

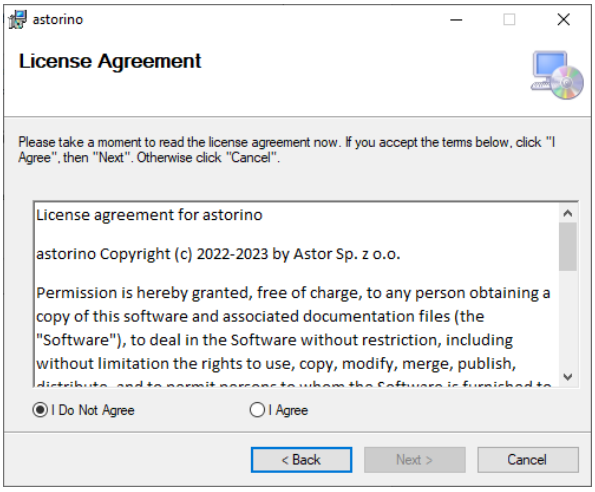
Run astorino_x.x.x.exe



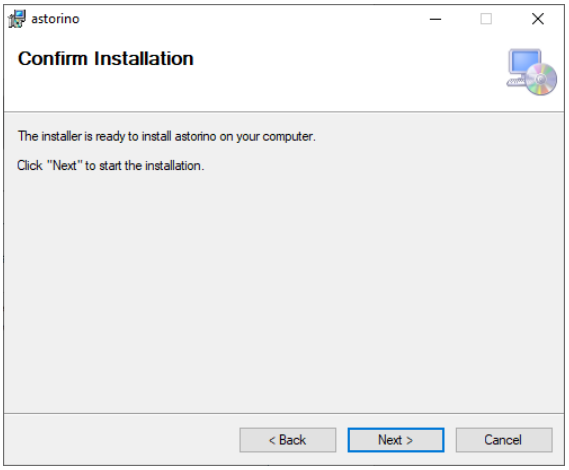
Confirm or specify installation directory



Read and accept the license agreement

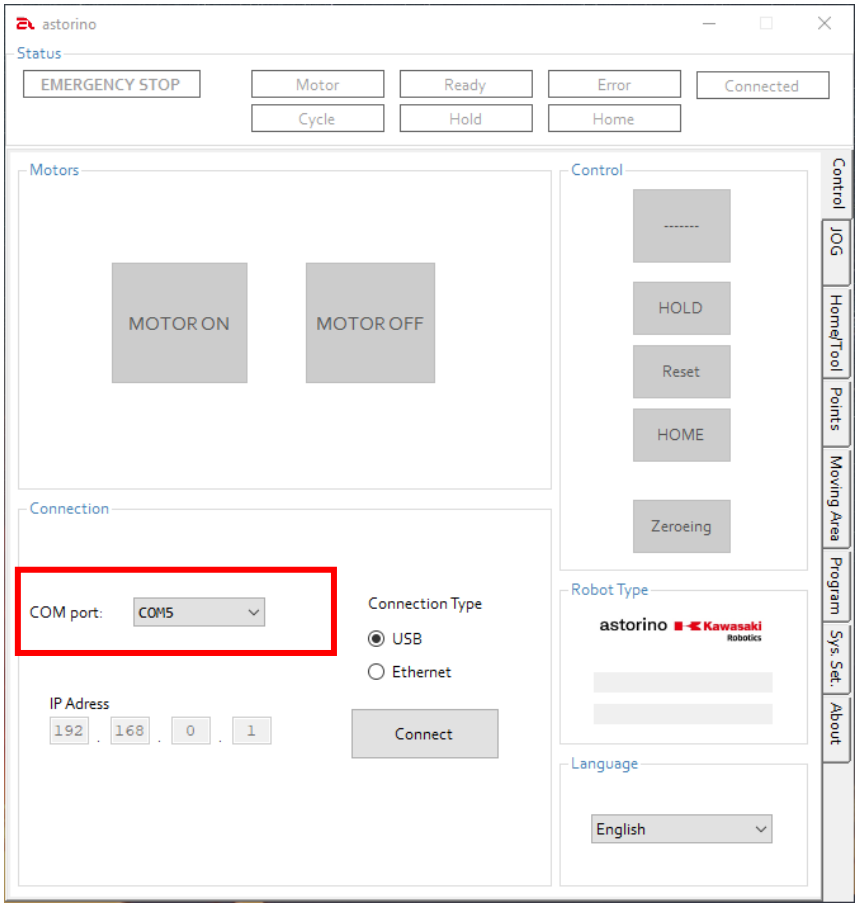


Start the installation

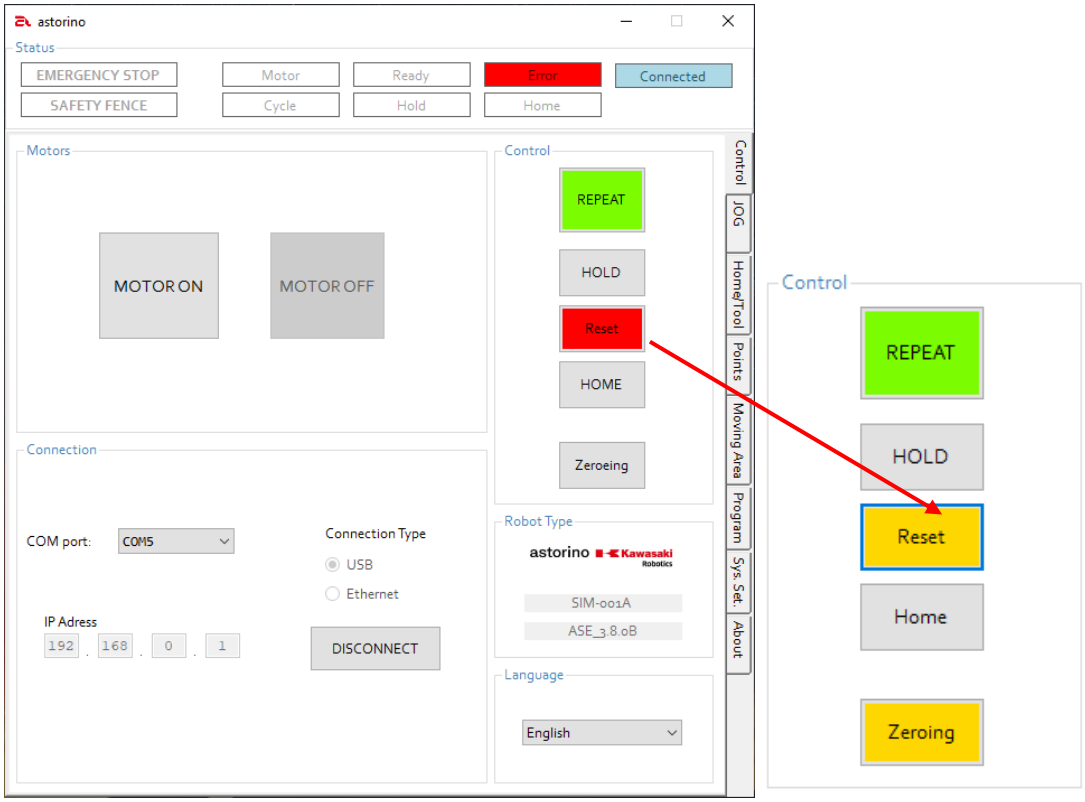


11.5 Making the astorino ready for operation

- Open the astorino software.
- The COM port to which the robot is connected will automatically appear in the drop-down list in the [Control]-menu Connection area.



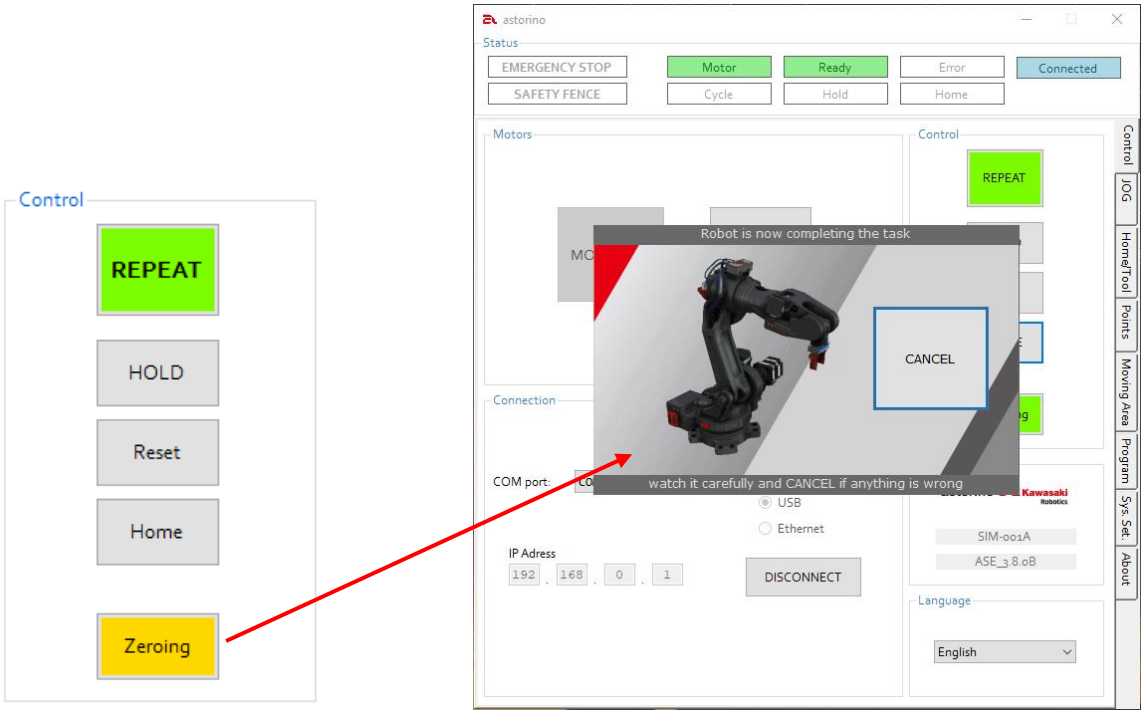
- Click [Reset], when this button is red (check the emergency stop button)



- The motors are powered by clicking on the gray [MOTOR ON] button.
- Click on the yellow flashing field [Zeroing] to perform the zeroing process.

  Zeroing must be performed each time the robot is powered or the motors are disabled.

- Make sure that the robot wont collide with any devices while zeroing is performed! Default zeroing procedure is described in the appendix of this document.



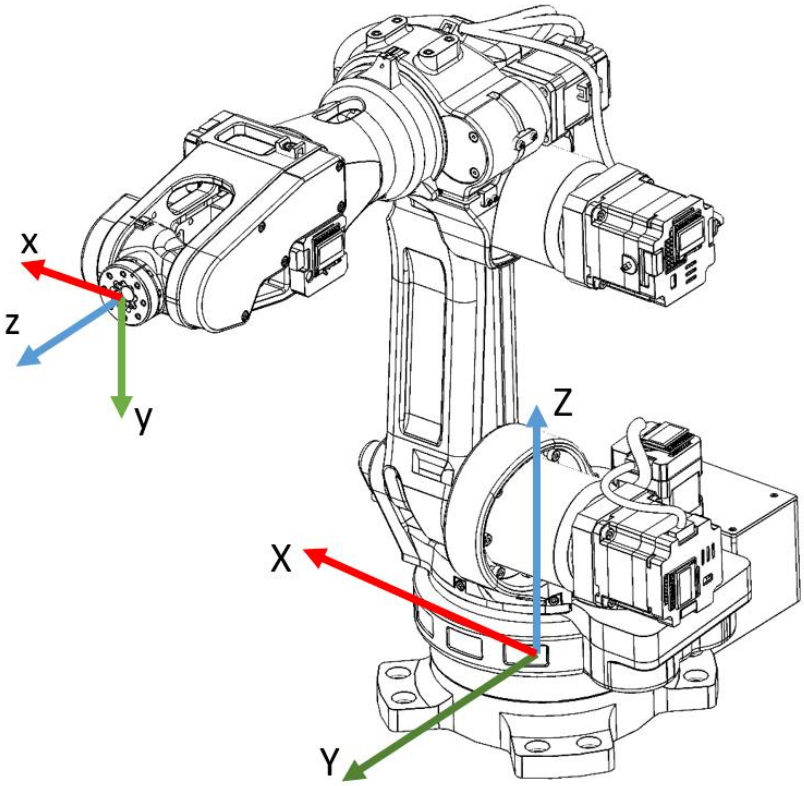
- When zeroing is complete, the robot stops at 0 degrees on each axis (with default zeroing procedure) and is ready for programming.

ATTENTION!

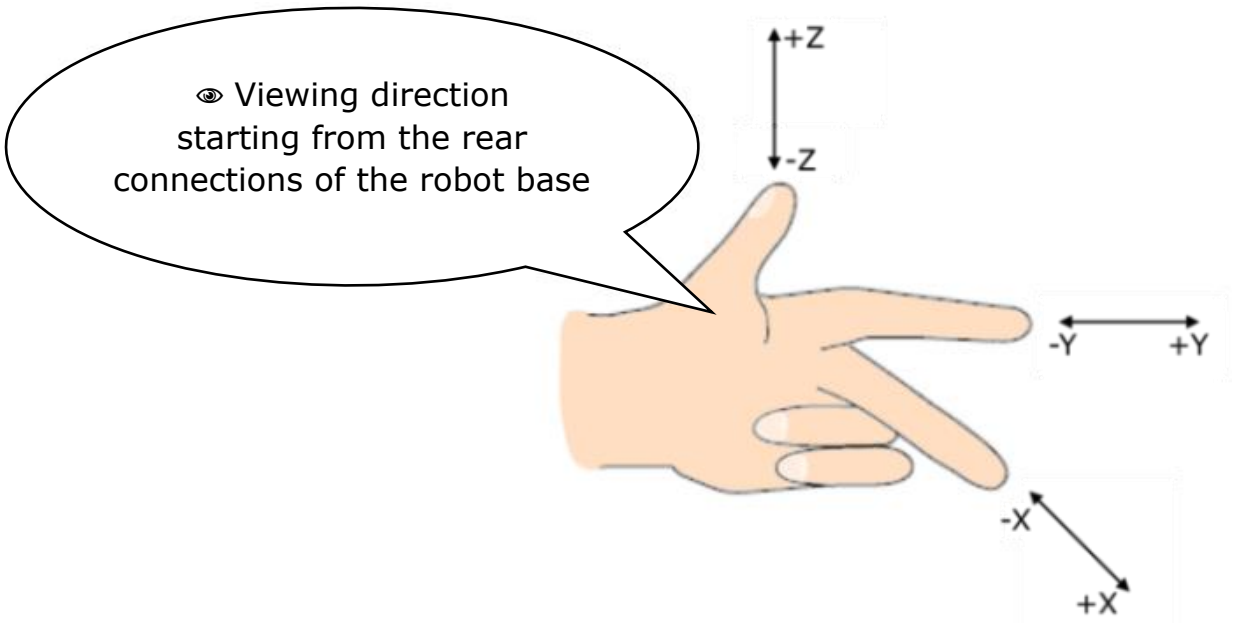
The upright position is the standard position of the arm after the zeroing process is complete. If the settings of the reset procedure have been changed, the end position may be different!

12 Coordinate systems

12.1 The BASE coordinate system

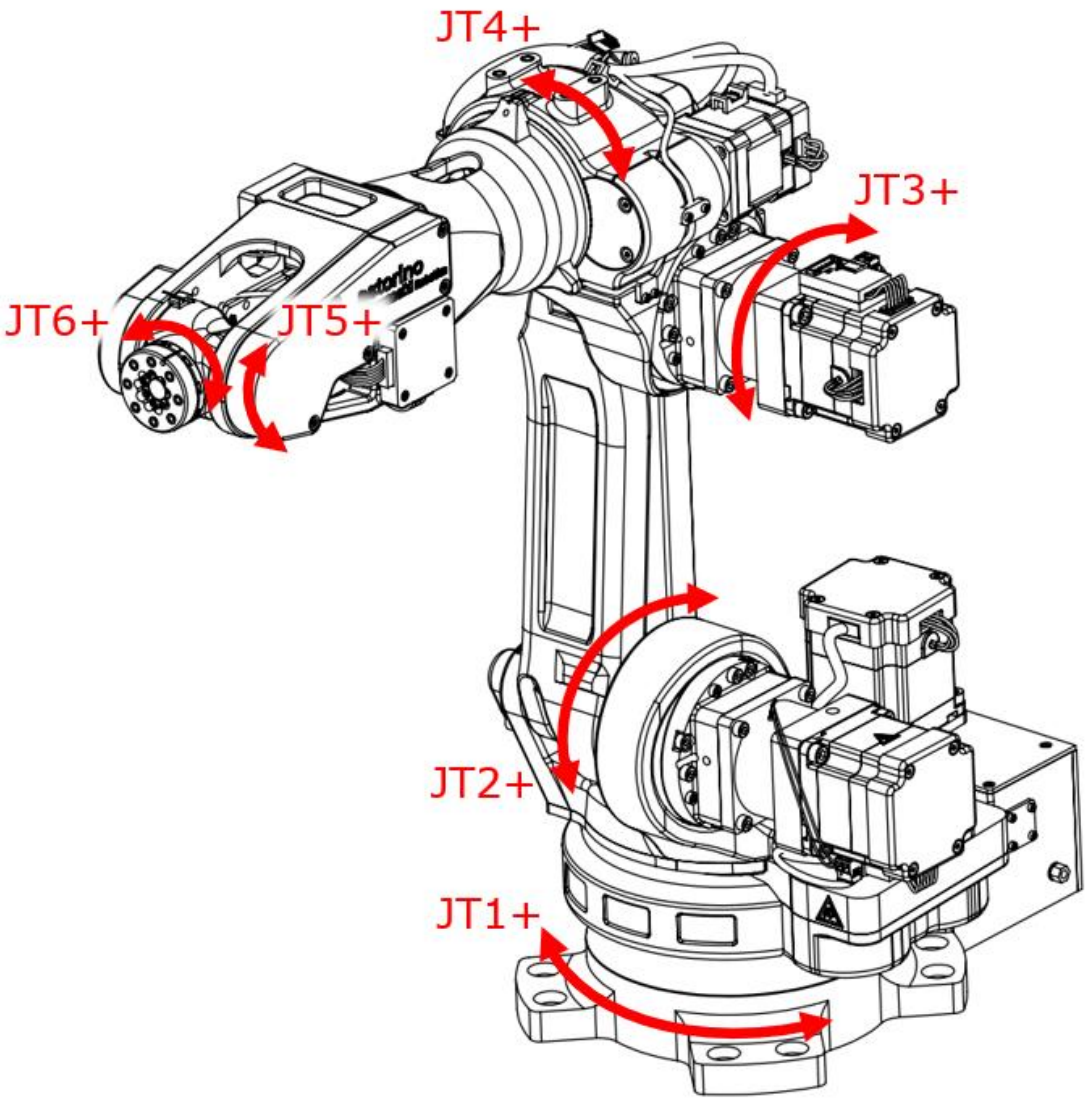


The left hand rule helps to remember the axis directions:



12.2 The JOINT coordinate system

The individual joints are numbered in ascending order, starting from the robot base. JT stands for joint.



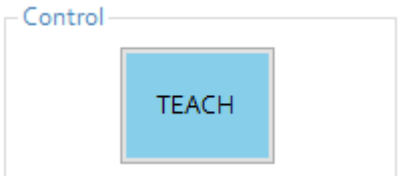
13 Robot operation modes

13.1 Teach Mode

This mode enables manual operations of the robot like moving, teaching. In this mode the maximum speed is limited to 60 mm/s and 12 deg/s for each axis.

Teaching is defined as programming the robot to do the required tasks and recording the positions data.

Robot is in the Teach Mode when [REPEAT/TEACH] switch's background is blue.



In Teach Mode when Safety Fence input is High robot operations are not restricted.

13.2 Repeat Mode

Repeat operations plays back the contents of a program that was taught to the robot. In this mode speeds are not restricted and robot can move at maximum of 250 mm/s.

Robot is in the Repeat Mode when [REPEAT/TEACH] switch's background is green.

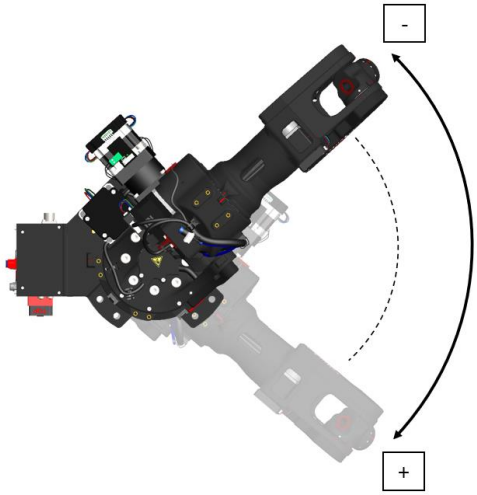
In Repeat Mode when Safety Fence input is High robot operations are restricted.

14 Manual operation of robot

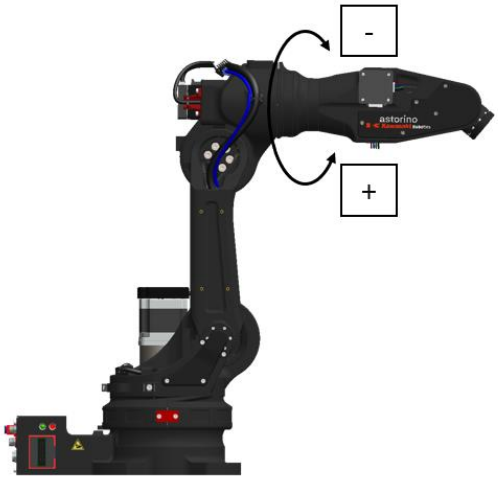
In Teach mode based on the currently selected motion mode (BASE, JOINT, TOOL) manual movent of the robot arm is possible.

14.1 JOINT

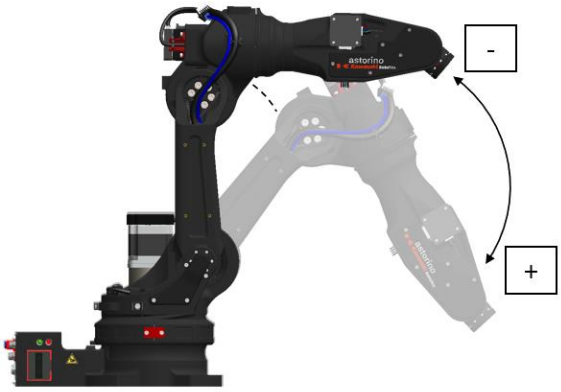
JT1: Left and right rotation of an arm



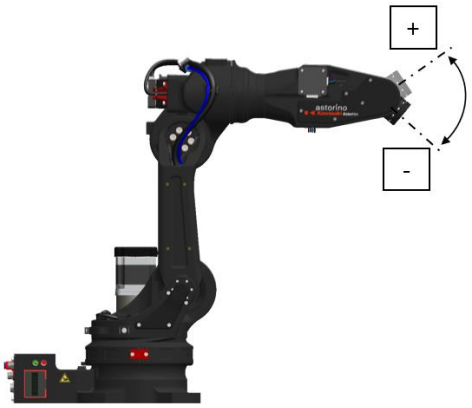
JT4: Rotation of a wrist axis (1)



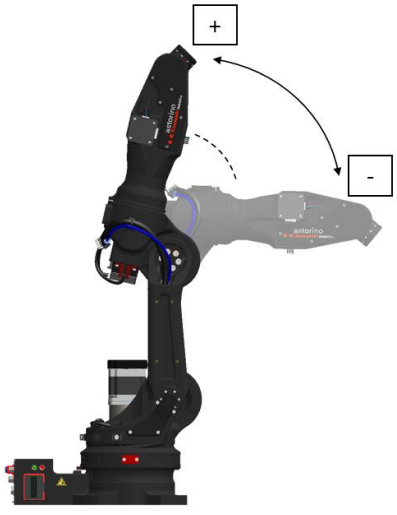
JT2: Back and forth motion of an arm



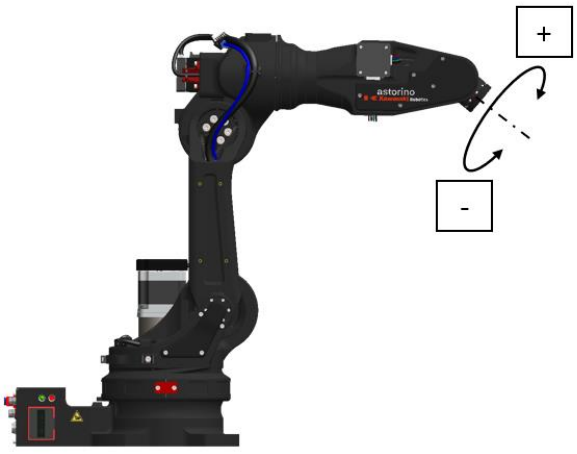
JT5: Rotation of a wrist axis (2)



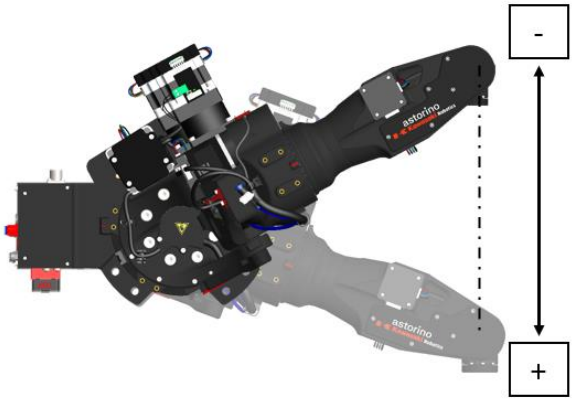
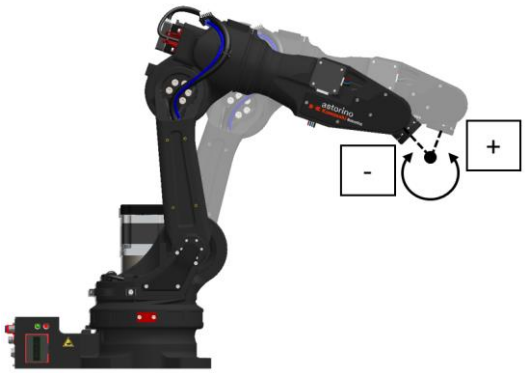
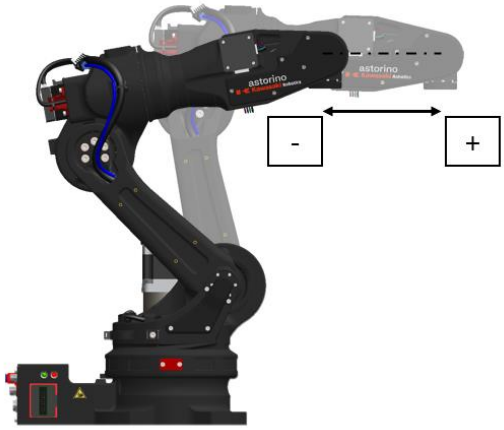
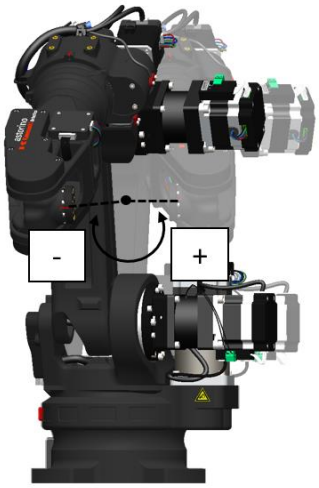
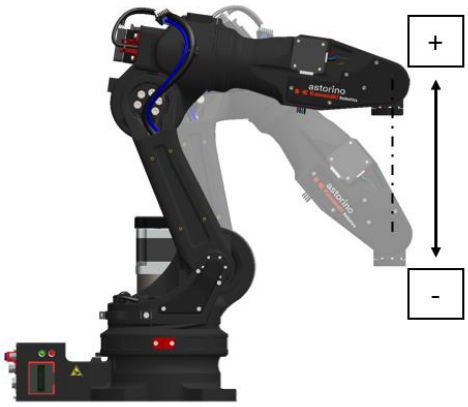
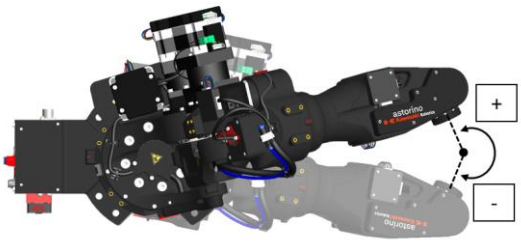
JT3: Up and down motion of an arm



JT6: Rotation of a wrist axis (3)

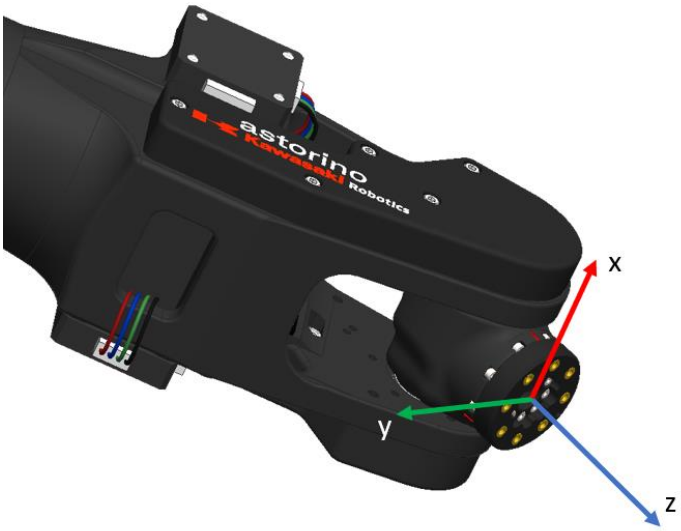


14.2 BASE

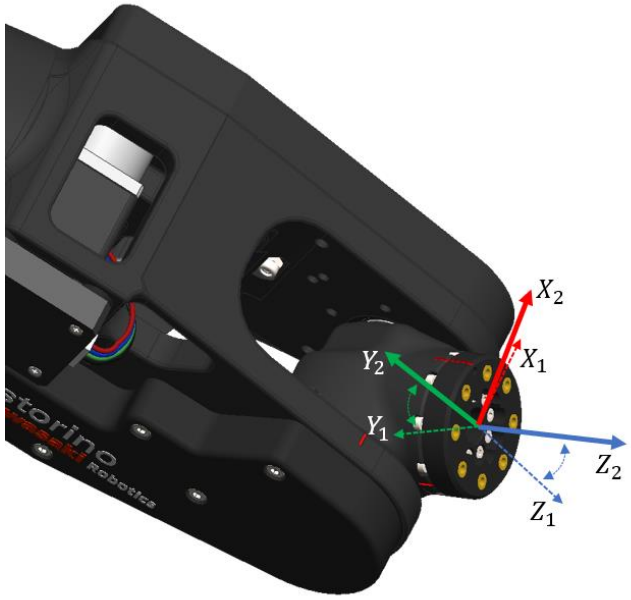
X: Motion parallel to base X coordinate (wrist orientation is constant) 	RX: Rotation around base X coordinate (TCP does not move) 
Y: Motion parallel to base Y coordinate (wrist orientation is constant) 	RY: Rotation around base Y coordinate (TCP does not move) 
Z: Motion parallel to base Z coordinate (wrist orientation is constant) 	RZ: Rotation around base Z coordinate (TCP does not move) 

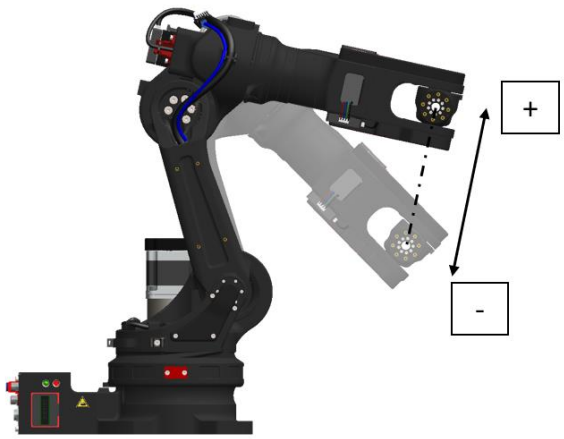
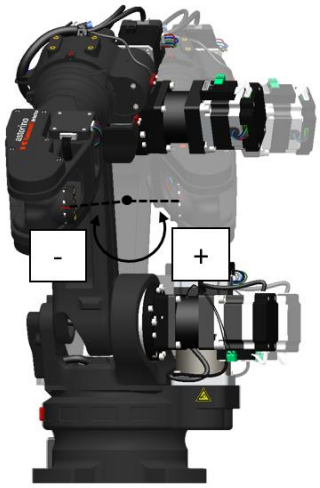
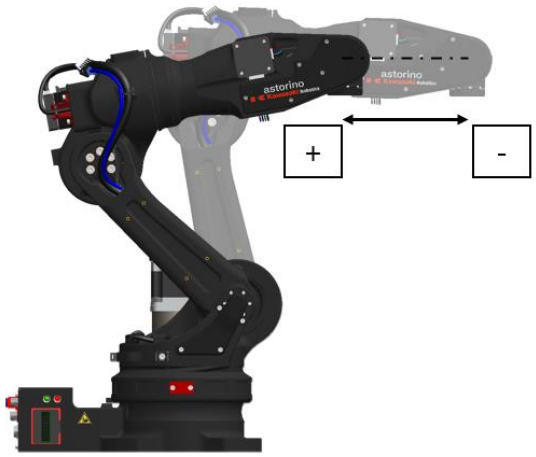
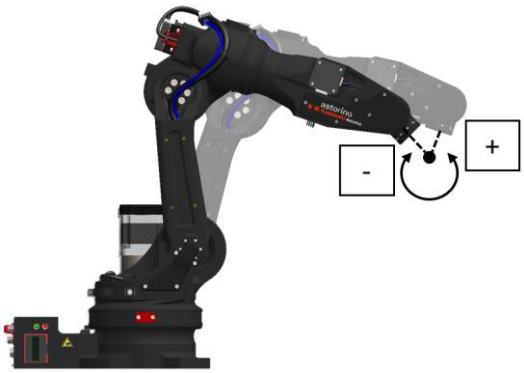
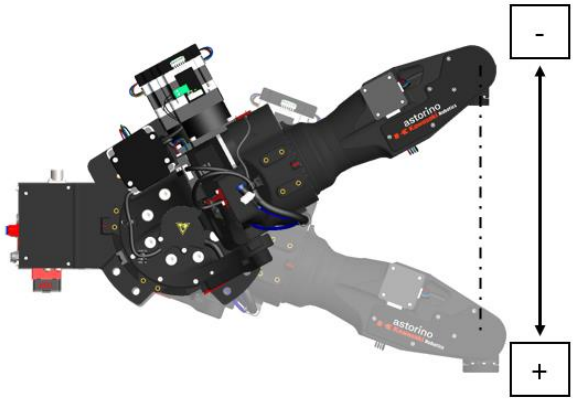
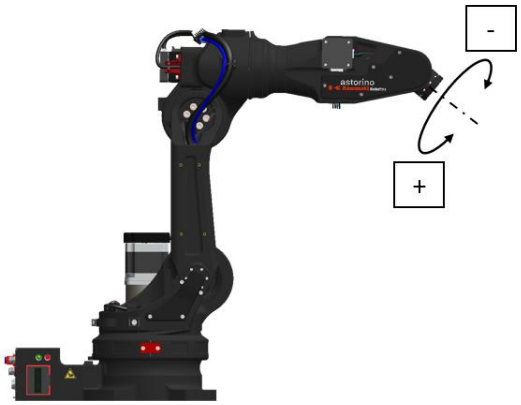
14.3 TOOL

Tool coordinate system is defined on the tool which is installed in JT6. Operations based on this tool coordinate system will differ in motion direction depending on the coordinates transformation to the null-tool coordinates. Tool coordinates also change when wrist orientation changes as shown in figures below, even though only the forearm moves without moving the wrist axes.



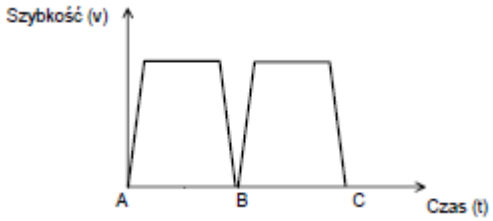
Arm at different location and orientation



x: Motion parallel to tool X coordinate (wrist orientation is constant) 	rx: Rotation around tool X coordinate (TCP does not move) 
y: Motion parallel to tool Y coordinate (wrist orientation is constant) 	ry: Rotation around tool Y coordinate (TCP does not move) 
z: Motion parallel to tool Z coordinate (wrist orientation is constant) 	rz: Rotation around tool Z coordinate (TCP does not move) 

15 ROBOT MOVEMENT

Acceleration for the second segment begins after the execution of the first segment is completed, when the current position is at the target point. The slope of the speed rise is determined by the ACCEL parameter and the braking edge by the DECEL parameter.



Astorino robot can move in three different ways. These ways are called interpolations. We can distinguish:

- Linear interpolation
- Joint interpolation
- Circular interpolation

In an anthropomorphic robot arms (6 axis) there exists some positions that are called singularities. A singular position where problem of structurally uncontrollable position might occur exists when for example JT4 and JT6 are parallel to each other, or JT1 and JT6 are parallel to each other. These configurations return multiple mathematical solution of inverse kinematics and therefore the motion through these points might be unpredictable and introduce a lot of very fast joint movements.

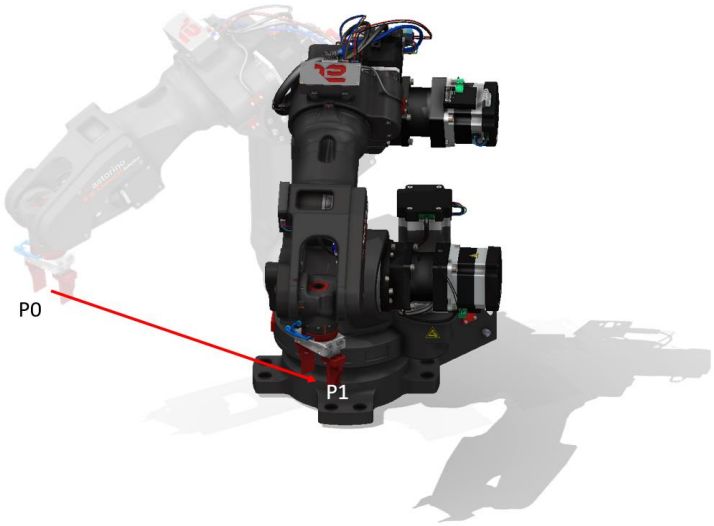
Examples of singular positions

JT4 and JT6 are parallel

JT1 and JT6 are parallel

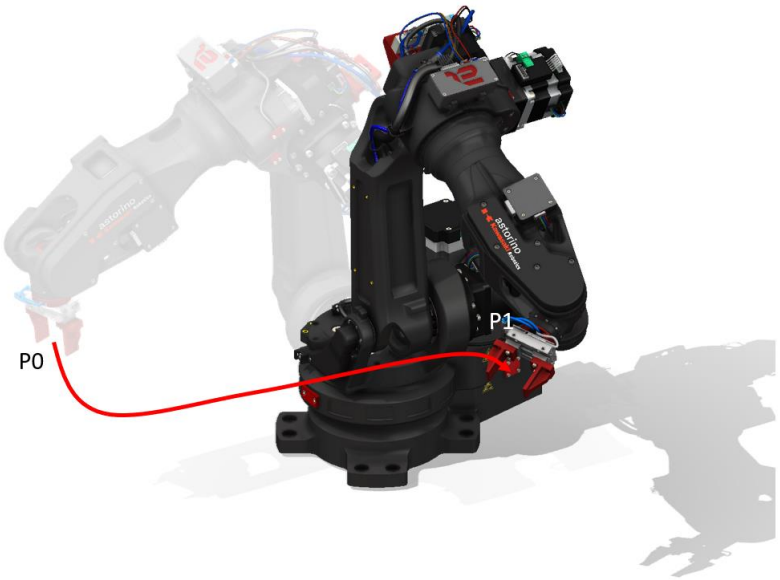
15.1 LINEAR INTERPOLATION

In this type of interpolation robot moves from the current position to the destination in that way that the TCP moves along straight line in a 3D space.



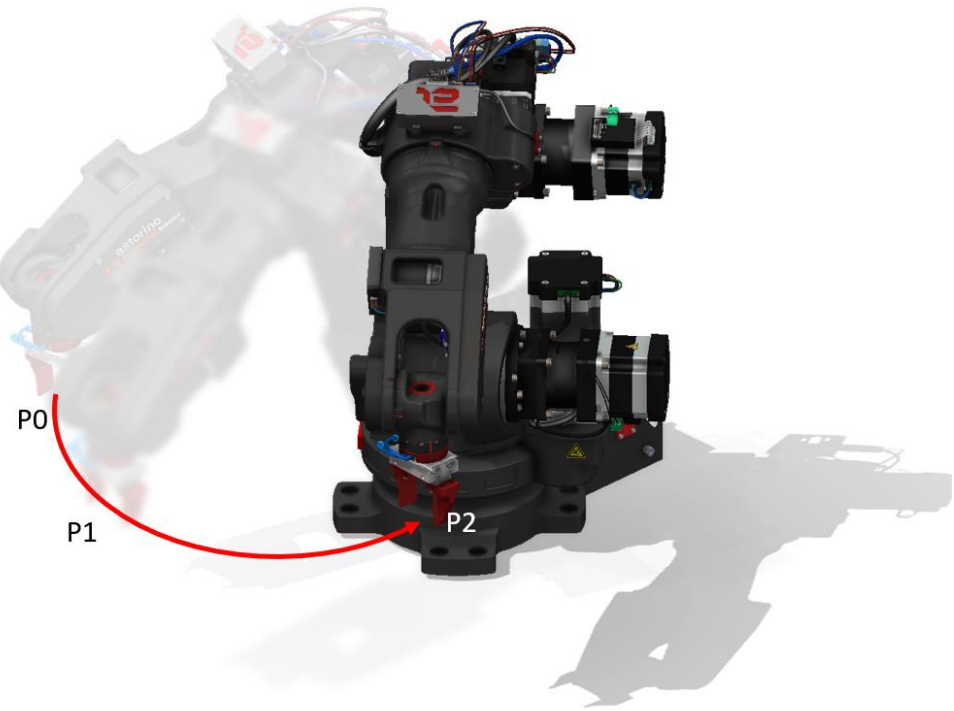
15.2 JOINT INTERPOLATION

In this type of interpolation robot moves form the current position to the destination in that way that all axes end motion at the same time. This movement creates an unpredictable TCP path in a 3D space. This motion in not effected by singularity points.



15.3 CIRCULAR INTERPOLATION

In this type of motion robot moves from the current position to the destination through the middle point in that way that the TCP creates a 3D circular line in a 3D space.

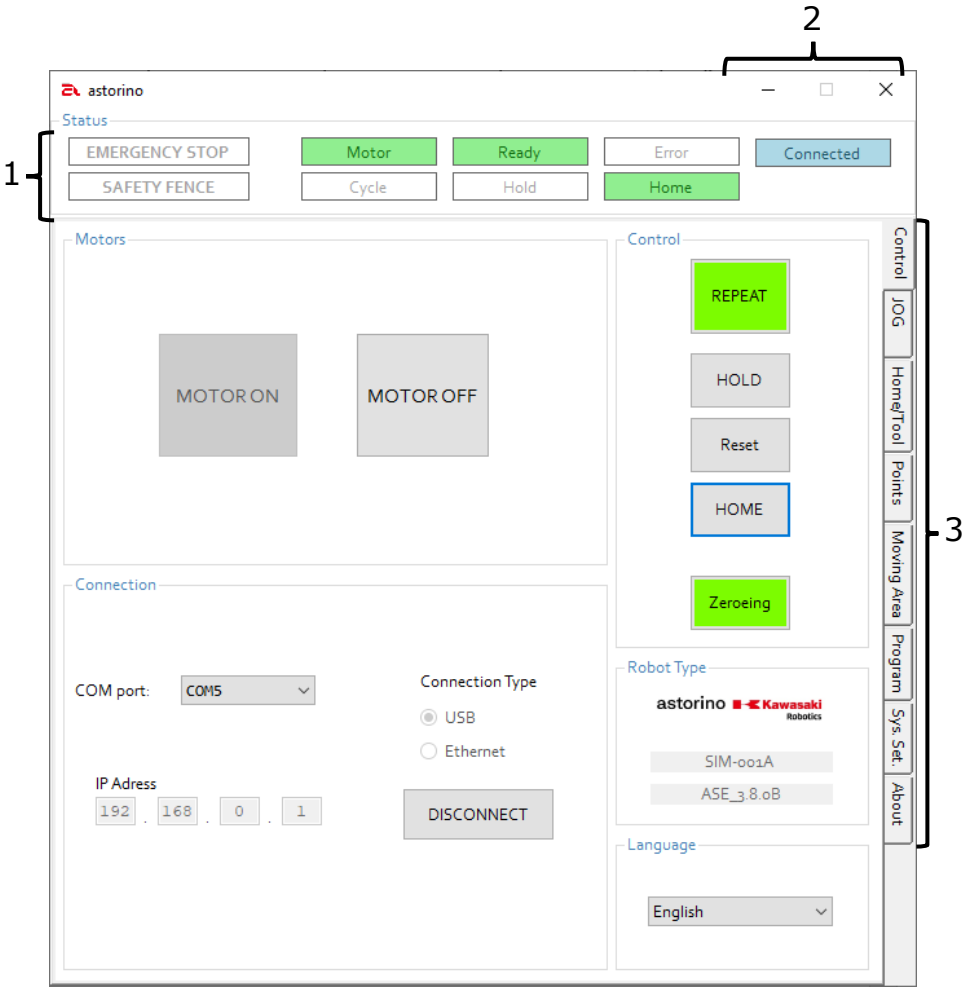


16 astorino Software

16.1 Basic information

All data is stored on the internal micro SD card, which is located on a microcontroller board inside the robot base. If the robot is switched off user data is not deleted.

Main window overview.

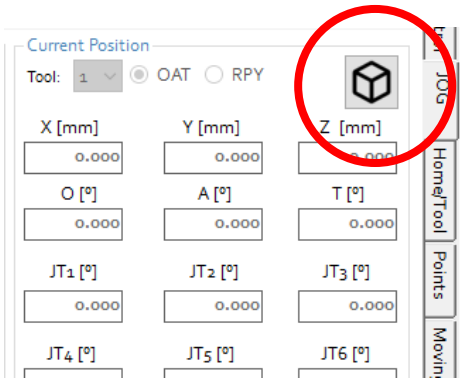


1. Status	Current status of connected robot
2. Application controls	Closing or minimalizing the astorino application
3. Operations tabs	Switching between different operations tabs

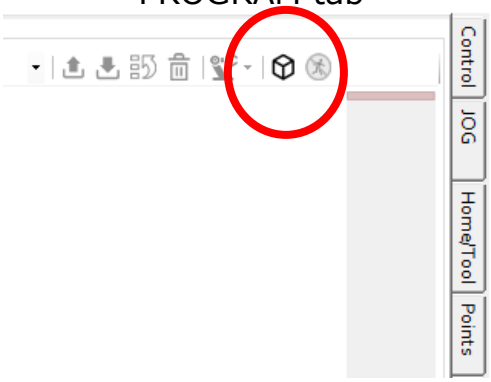
16.2 Visualization Window

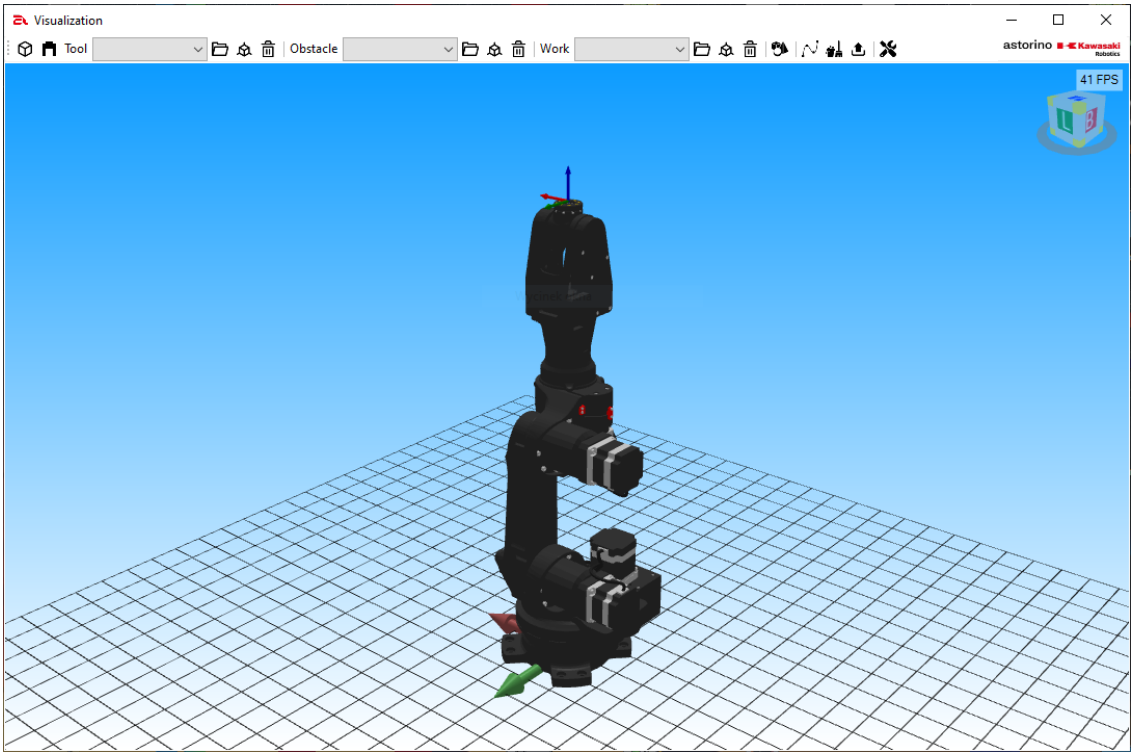
To open the visualization window and see the operation of the Astorino robot in real time, click one of these two buttons

JOG tab



PROGRAM tab





16.2.1 Visualization window handling

The visualization window allows you to add 3D objects to the scene with the robot. The program supports stl files and allows you to add basic three-dimensional shapes. You can add each feature as one of three object types:

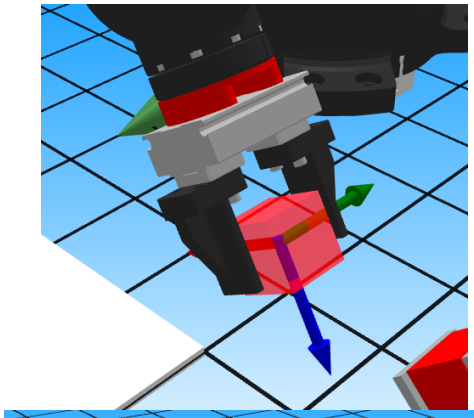
- Obstacle – objects of this type are static objects of the scene
- Work – objects of this type can be moved by a robot
- Tool – objects of this type always move with the robot flange.

The visualization window menu consists of the following elements::

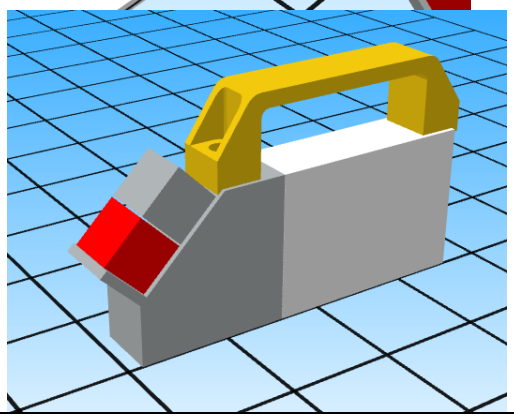
	Enables or disables the Working Space view
	Enables or disables the 3D model of a standard gripper
	Lists Tool, Obstacle, or Work objects
	Opens the .stl file and loads it as one of the Tool, Obstacle, or work class objects
	Enables the object modification menu, allows you to change the position of the object or change its color
	Deletes the currently selected object in the drop-down list
	Enables the menu of the 3D Simple Shapes Generator
	Enables the generation of robot trajectory visualizations
	Disables and clears the visualization of the robot's trajectory
	Saves robot trajectory visualization points to .traj files
	Enables the visualization window settings menu

16.2.2 Object types

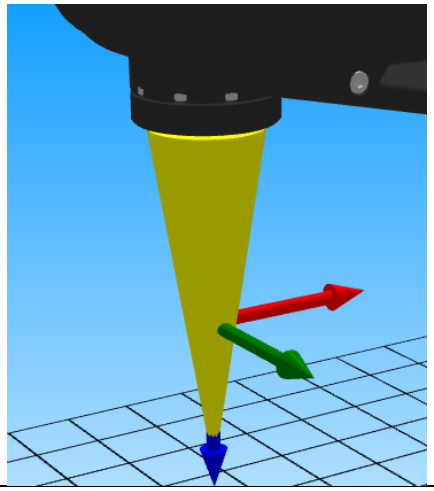
Work – Work class objects can be moved by a robot. For an object to be captured, the TCP point must be inside the work object and the control signal must be in a high state.



Obstacle – Obstacle objects are static visualization elements. They allow you to build a visualization scene, are a visual aspect and potential obstacles.



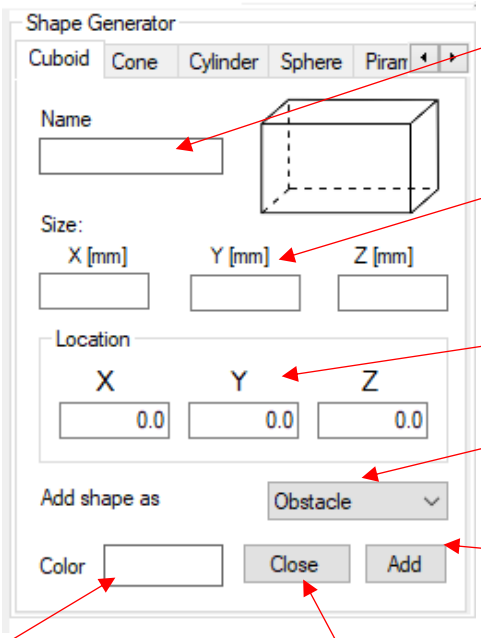
Tool – Tool class objects are objects that are permanently attached to the robot flange. Thanks to these objects, you can create your own tools, which are mounted on the robot flange.



16.2.3 Simple Shape Generator

The generator of simple three-dimensional shapes allows you to generate the following elements:

- Cube,
- Cuboid,
- Cone,
- Cylinder,
- Sphere,
- Piramid,
- Pipe.



Object name, when not entered – assigns another free name to the object automatically.

The size of the object, depending on the figure, should be given from 1 to 3 parameters.

The position under which the figure is to be created.

Selecting the class of the object to be created: Tool, Obstacle, Work.

Add an object to a visualization

Choosing an object color

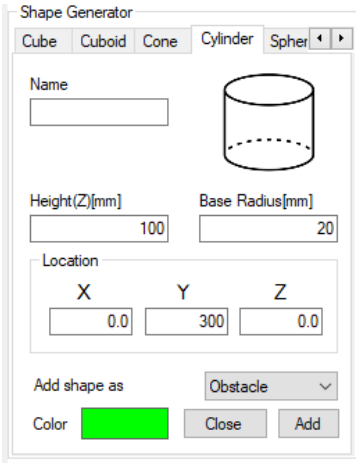
Closing the generator menu

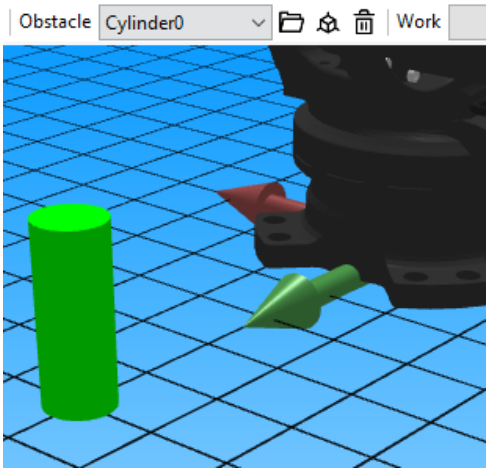
Example

Add to the visualization an obstacle class cylinder with the following parameters:

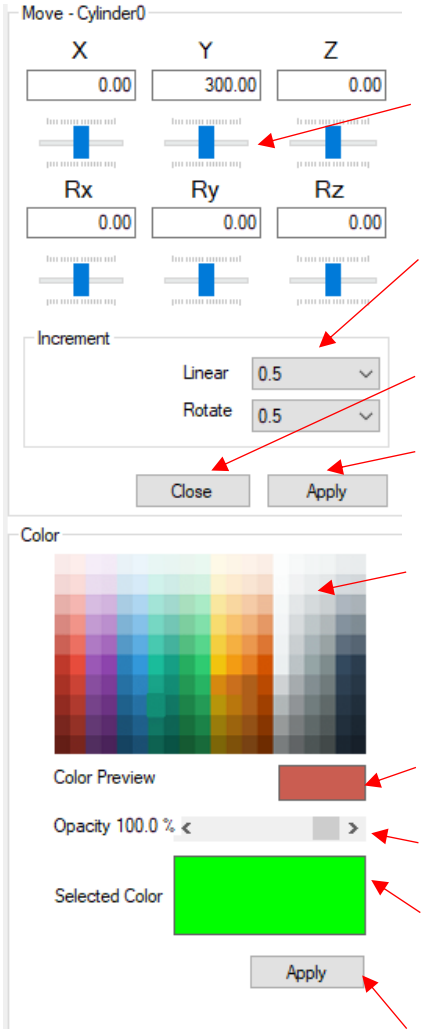
- 100 mm high,
- Base radius 20mm,
- Green
- Start position (0,300,0 [x,y,z])
- Any name

To add such an object, enter the following data in the generator menu and confirm with the [ADD] button. The object is added.





16.2.4 Objects modify menu



The name of the currently modified figure.

Sliders and text boxes to change the position of an object on a visualization.

Setting the resolution of the above sliders

Close the modification window.

Apply settings.

The quick color selection area, clicking the mouse button on a specific color will allow you to select it as the color of the object being modified

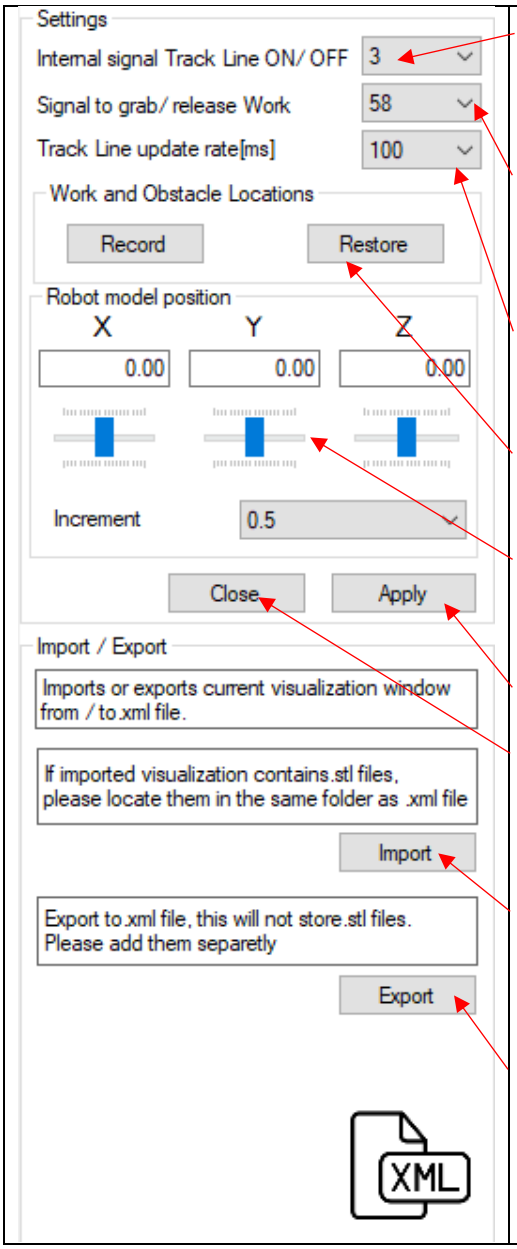
Preview window selected colors.

Set an object opacity.

Currently selected color, clicking on this area will open the advanced color selection window

Apply settings.

16.2.5 Visualization settings menu



Internal signal (from the pool 2001-2016) that enables or disables the generation of trajectory points on the visualization.

Output signal controlling the capture of Work class objects in visualization by the robot.

Time every time another trajectory visualization point is created.

Save and restore the position of elements on the visualization.

Changing the position of the robot on the visualization.

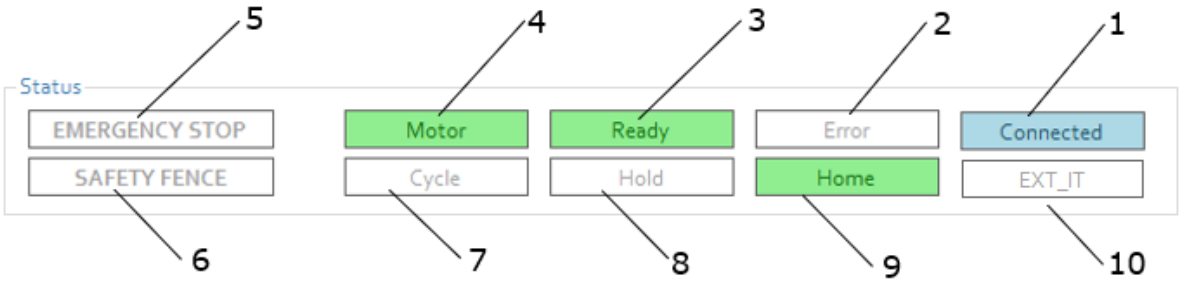
Apply settings.

Close the settings menu.

Import visualizations from an .xml file.

Export visualizations to xml files.
The export does not save the opened stl files. Only their names. Copy the files separately.

16.3 Status



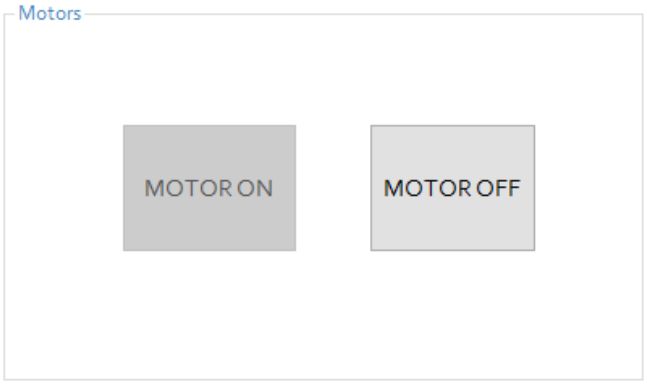
When the background of a field is highlighted, it means that:

1. Connected	A robot is connected to the astorino software
2. Error	An error has occurred
3. Ready	No emergency stop, no errors , the stepper motors are enabled and zeroing is done
4. Motors	The stepper motor drivers are active
5. EMERGENCY STOP	Emergency stop is pressed and active
6. SAFETY FENCE	Safety fence is open
7. Cycle	The program sequence is being executed
8. Hold	The robot is stopped
9. Home	The robot is in its home position
10.EXT_IT	The robot was stopped by an external interrupt

16.4 Control

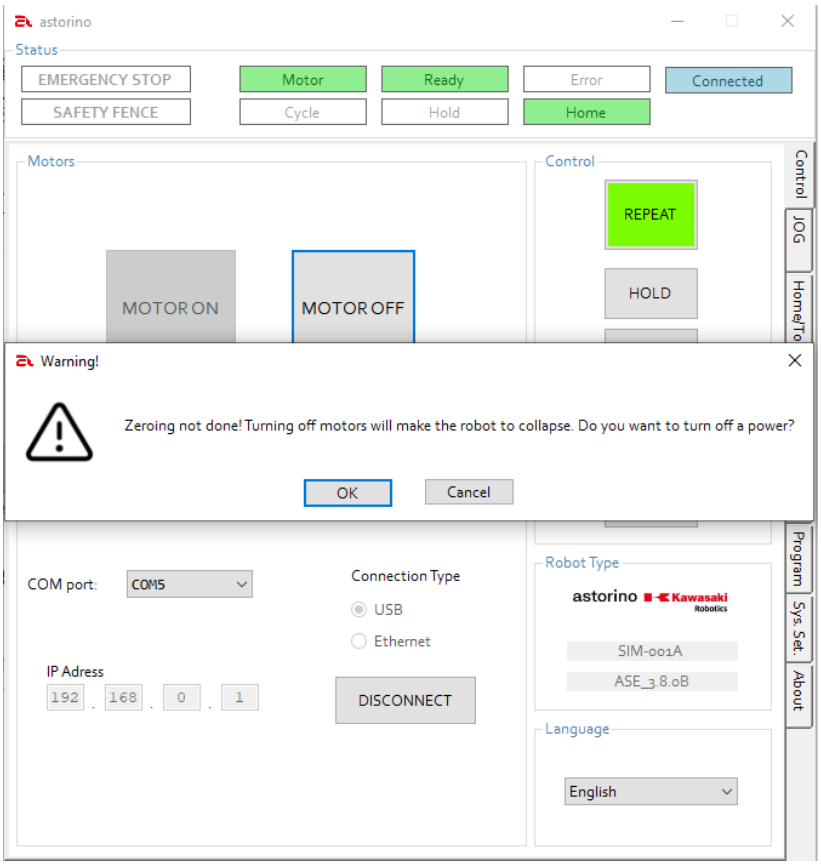
1. Motors — Status and control over the motors
2. Control — Display of operation mode, stop robot, move to home position, error acknowledgement and zeroing
3. Robot Type — Robot firmware version and serial number
4. Connection — Select and configure interface, establish connection or disconnect
5. Language — Selection of the displayed language

16.4.1 Motors (ON/OFF)



Pressing [MOTOR ON] activates the stepper motor drivers. This is only possible if no error is present! (Error-box ☐)

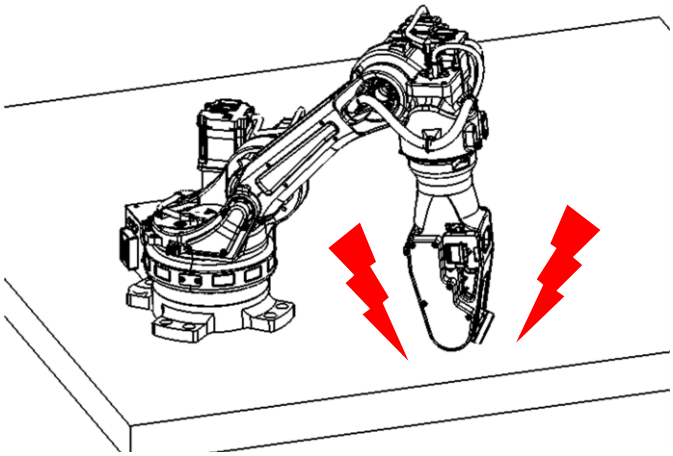
The status indicates the motor state (☑ Motors ON).

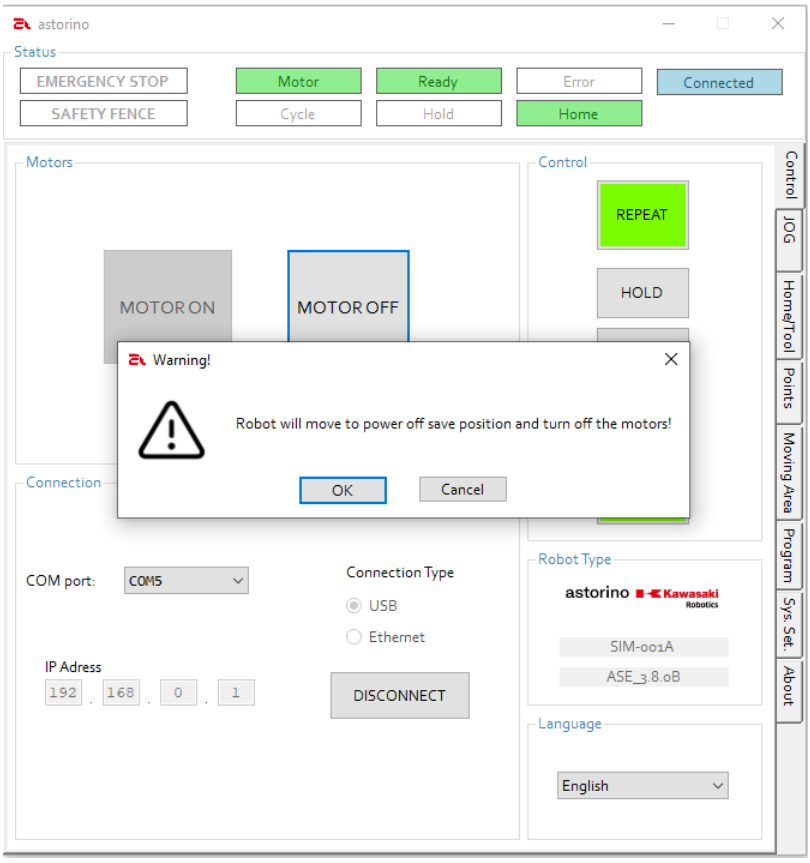


Pressing MOTOR OFF disables the stepper motor drivers.

WARNING

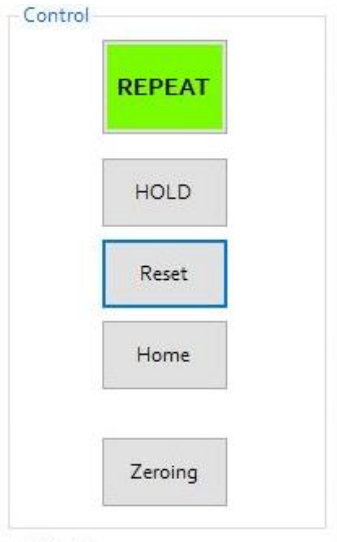
IMPORTANT! If no zeroing of the axes has been performed, be very careful and hold axis 4, as the robot could collapse on itself.





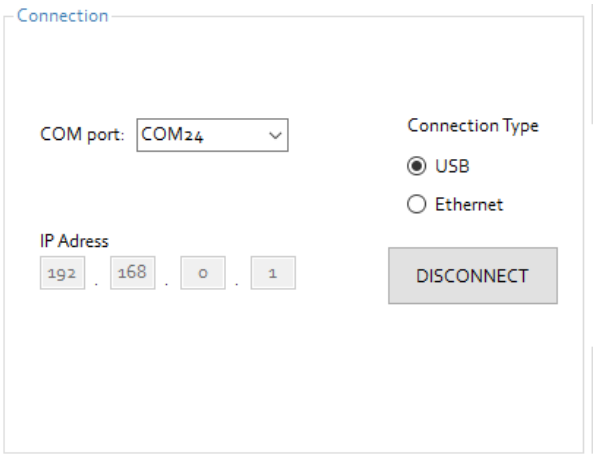
When [Zeroing] is present, the Astorino moves automatically to its safe power off position after the warning message has been acknowledged.

16.4.2 Control



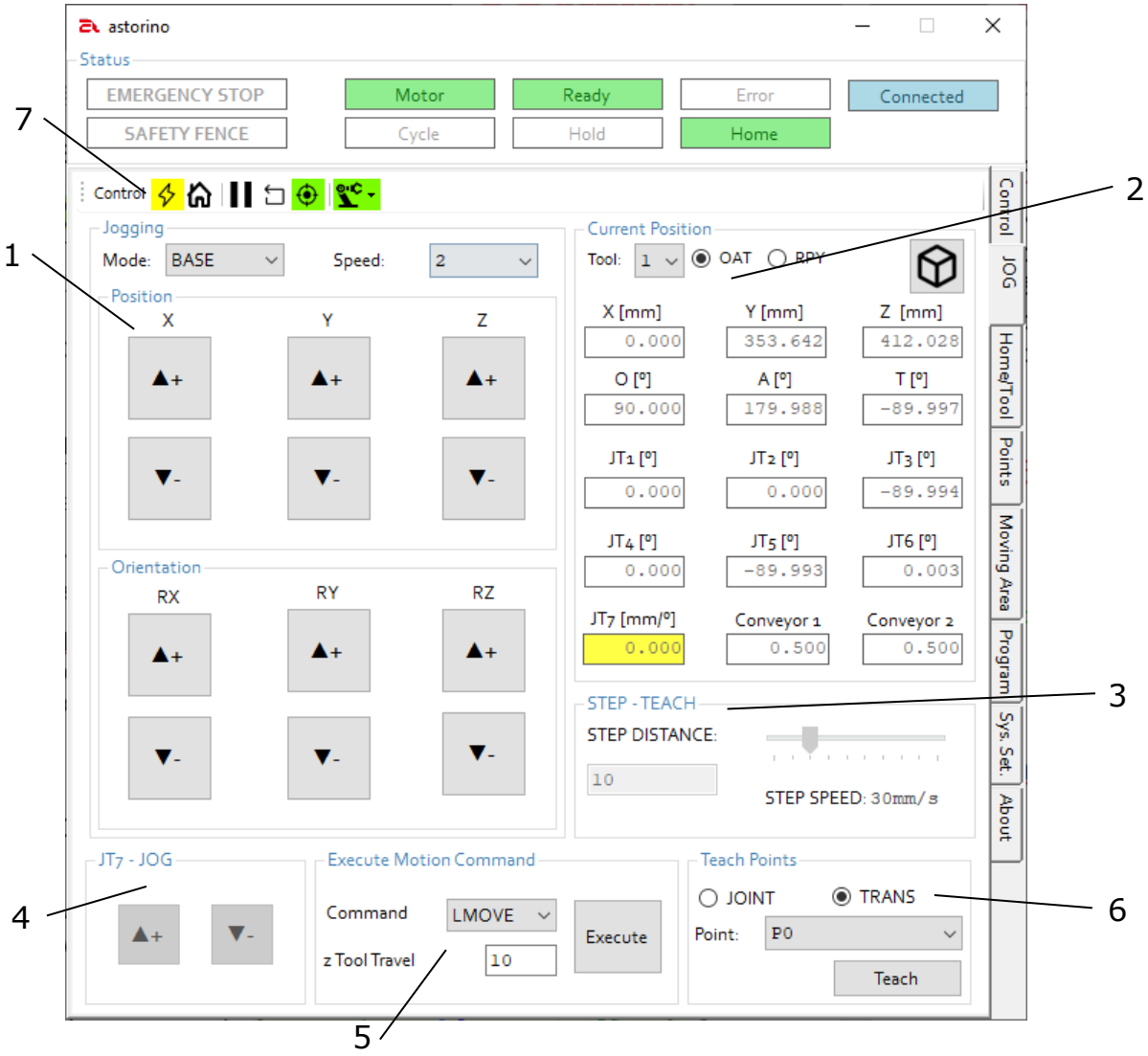
1. [REPEAT/TEACH] – Changes the operation mode. The button turns green when the robot is in automatic mode (REPEAT) and blue when it is in teach mode (TEACH).
2. [HOLD] –Stops the robot.
3. [Reset] – Reset errors.
4. [Home] – Moves the robot to home position (adjustable – see chapter 12.6).
5. [Zeroing] – Zeros the axes of the robot (needs to be done after the motors are enabled).

16.4.3 Connection



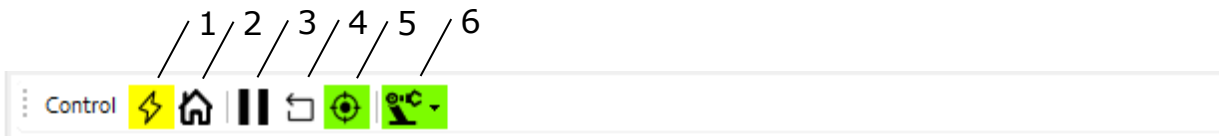
1. COM port - Displays the different ports to which the robot is assigned. This list is empty if no robot is connected to a PC via USB
2. Connection Type - Select the connection method being used (USB cable / network cable).
3. IP Address - enter the IP address of the robot. This is used only for Ethernet communication.
4. [CONNECT/DISCONNECT] – click the button to connect to the robot or to disconnect it.

16.5 JOG



1. Jogging	Specify traverse mode and traverse speed, move the robot
2. Current Position	Tool selection, angle display*, current robot position
3. STEP – TEACH	Set step size and step speed
4. JT7 – JOG	Move linear axis (JT7) - if attached
5. Execute Motion Command	Execute the specified command
6. Teach Point	Specify a point to teach or to move to

7. Control	Duplicated functionality from Control Tab



1. Motor on/off
2. Home
3. Hold
4. Reset
5. Zeroing
6. Switch Teach/Repeat

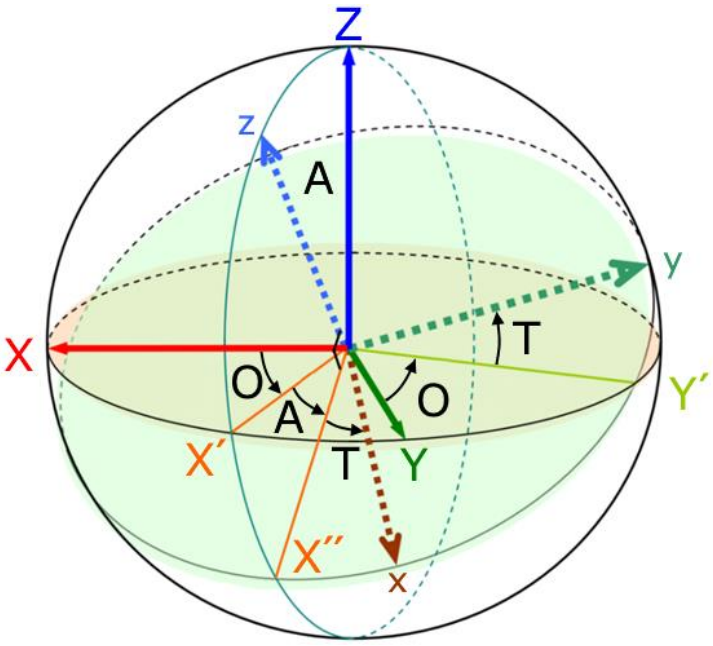
* Angle display or applied rotation sequences

When calculating motion, the robot uses classic EULER (OAT) to calculate the robot path and position.

For ease of teaching **R**oll-**P**itch-**Y**aw is used as its more intuitive for the user this is automatically converted to a OAT position by the robot.

Classic Euler O,A,T Angles

The position format (POSE) used by Kawasaki robots consists of a position XYZ in millimeters and an orientation OAT, which is specified by three angles in degrees, where <**O**> is rotated around the Z-axis, <**A**> rotates around the rotated Y-axis (Y') and <**T**> rotates around the rotated z-axis.



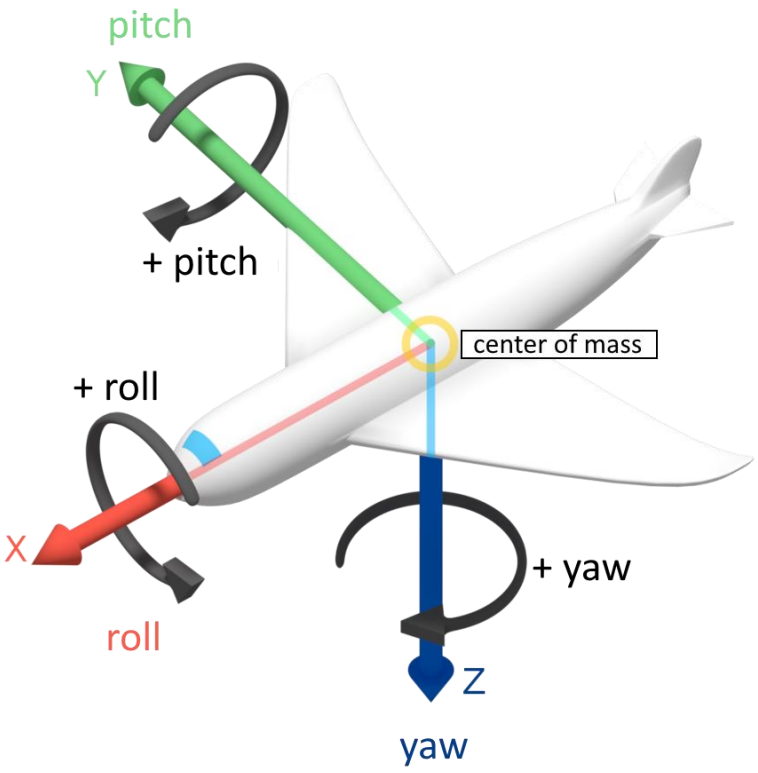
Roll-Pitch-Yaw (RPY):

Roll-pitch-yaw angles are special Euler angles (position angles) that are used to describe the orientation of an object in 3-dimensional space.

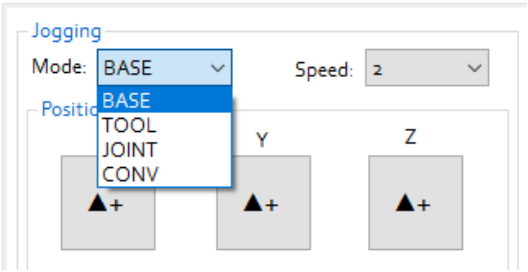
roll axis (longitudinal) X

pitch (cross axis) Y

yaw (vertical axis) Z

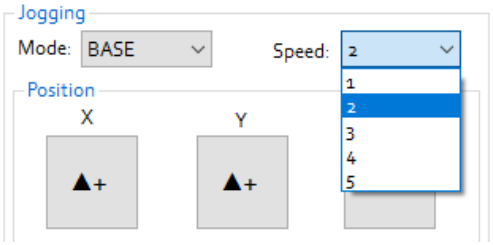


16.5.1 Jogging



Choose reference system:

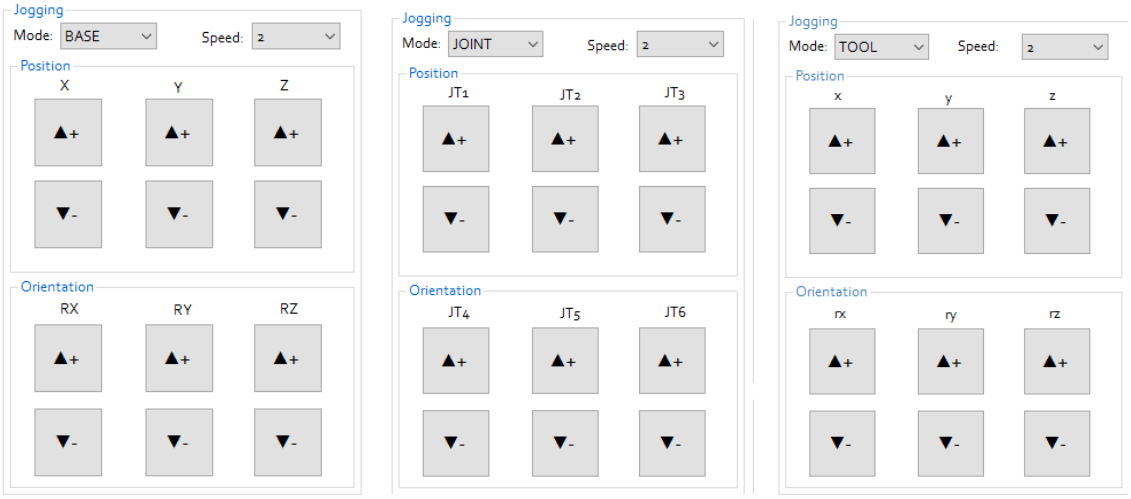
- BASE (base coordinates)
- TOOL (tool coordinates)
- JOINT (single axis mode)
- CONV (synchronous in combination with an optional external encoder - conveyor)



Setting the movement speed of the robot in steps.

Speed	Cartesian mode	Single axis mode (JOINT)
1	Movement by distance	Rotation by fixed angle
2	5 mm/s	2°/s
3	10 mm/s	4°/s
4	30 mm/s	8°/s
5	60 mm/s	12°/s

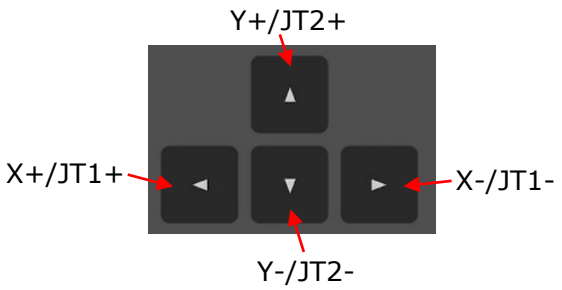
The "▲+" and "▼-" keys move the robot in teach mode at the selected speed:

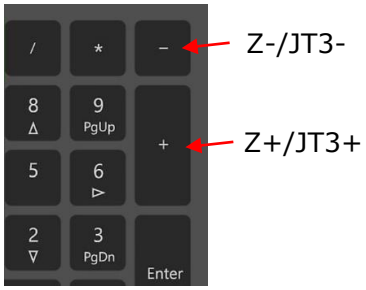


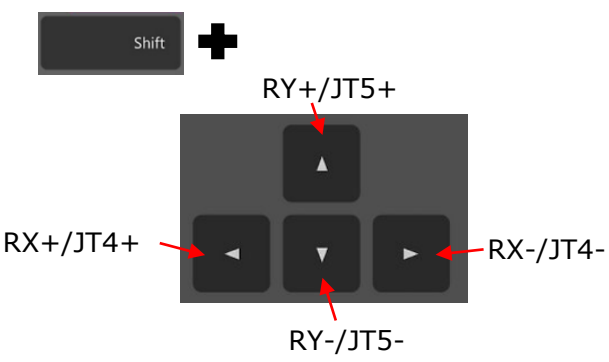
- BASE (Cartesian coordinates): X, Y, Z, RX, RY, RZ
- TOOL (Cartesian coordinates): x, y, z, rx, ry, rz
- JOINT (Single axis): JT1, JT2, JT3, JT4, JT5, JT6

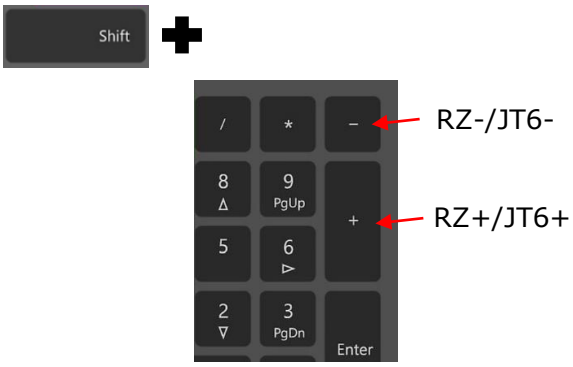
If currently selected speed is 1 (step) then after pressing JOG (+/-) buttons robot executes step motion. Step motion is set in STEP-TEACH section.

In addition, the robot can be moved using a keyboard:

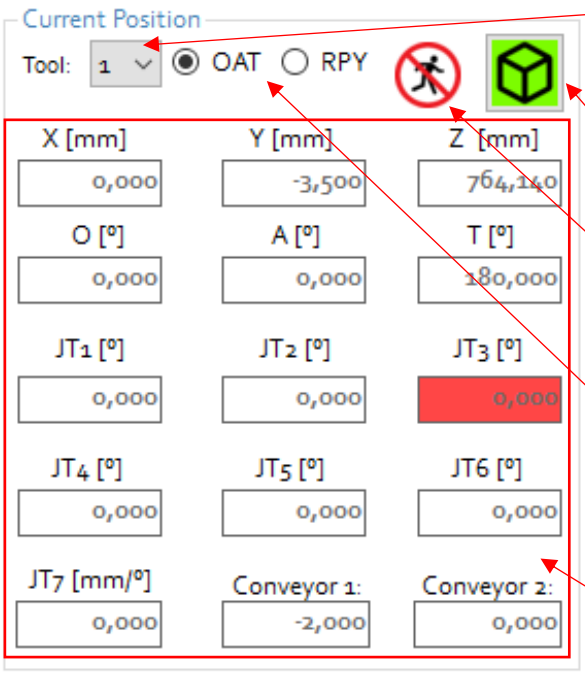








16.5.2 Current Position



Tool selection

1 e.g gripper
2 e.g. tip
3
4 Softwaretool

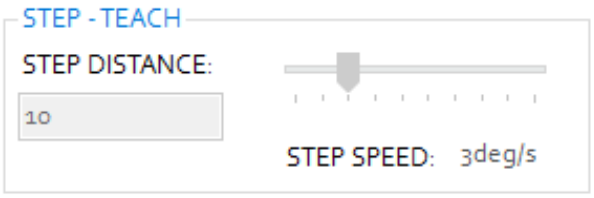
Show visualization

Display for DryRun mode (visible only if DryRun is active)

Specify the display of OAT or RPY angles

Current arm-position

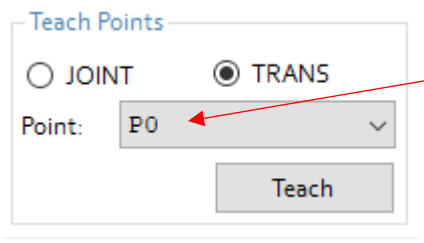
16.5.3 STEP - TEACH



STEP DISTANCE
Value in mm/° for the movement

STEP SPEED
Speed in % or mm/s

16.5.4 Teach Point

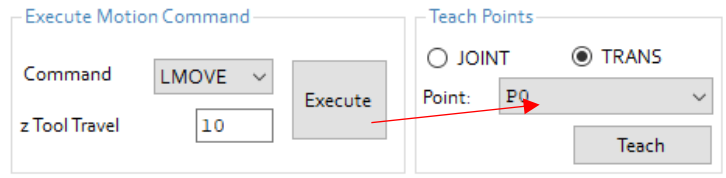


Select a point from the list to be taught.

Select whether the point is to be approached linearly (TRANS) or by the fastest route (JOINT).

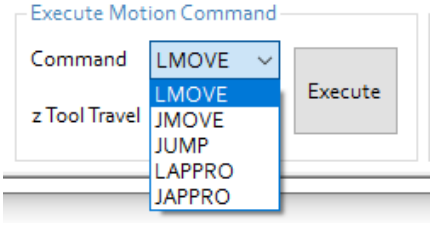
Pressing the [Teach] button saves the point in the robot's memory.

16.5.5 Execute Motion Command

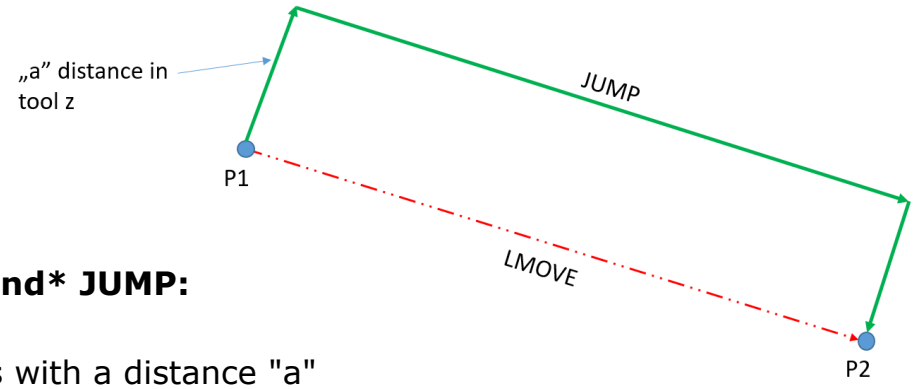


Command (possible movement commands):

- LMOVE (linear)
- JMOVE (fast)
- JUMP (special*)
- LAPPRO
- JAPPRO



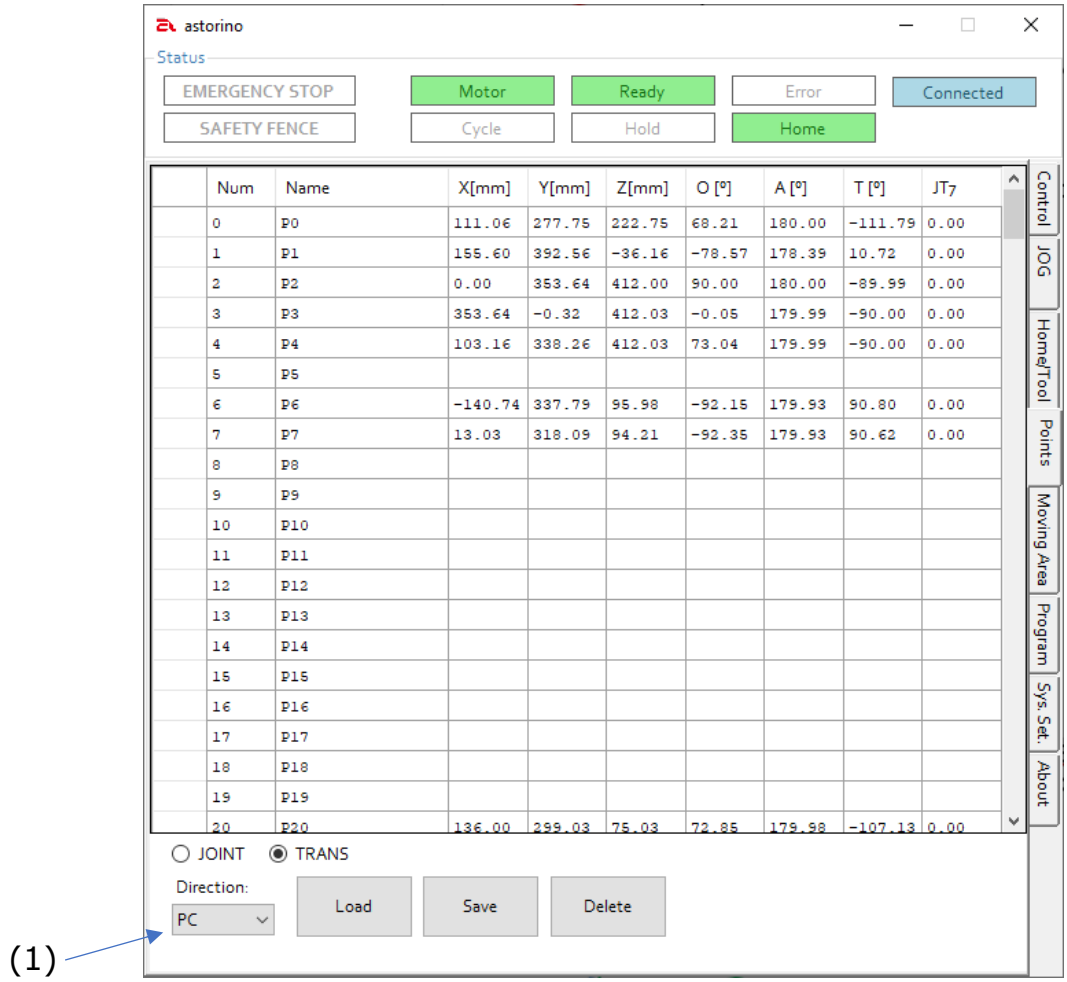
once [Execute] is pressed the specified movement command is executed for the point selected in the Teach Point area.



Special command* JUMP:

The robot moves with a distance "a"
to be defined from one point (e.g. P1) to the next (e.g. P2).
The distance "a" is the Z-direction in tool coordinates (TOOL).

16.6 Points



All points stored in the robot are displayed in tabular form.

Either all TRANS or JOINT points can be displayed.Points from 0 to 99 are labeled Px for example P0 or P10, points from 100 to 255 are user points and have name as user specify.

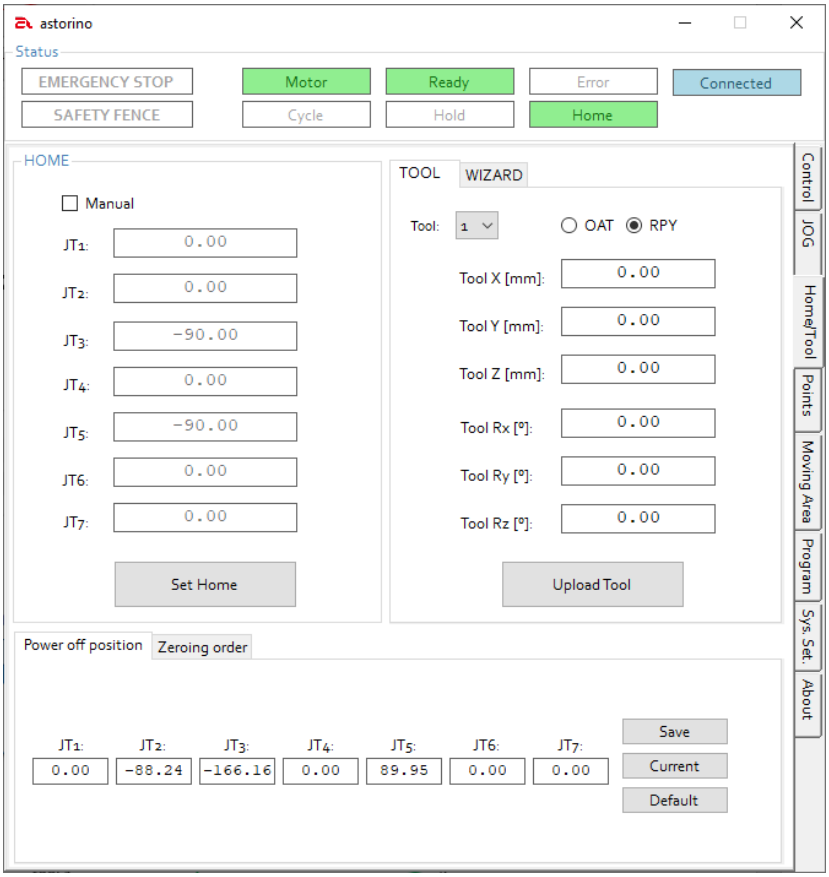
In the drop-down list (1), you can select in which direction the point data is to be loaded or saved. You can select either the connected computer or the robot controller.

The [LOAD] button loads data from the robot memory or from *.loc files into the ASTORINO robot controller.

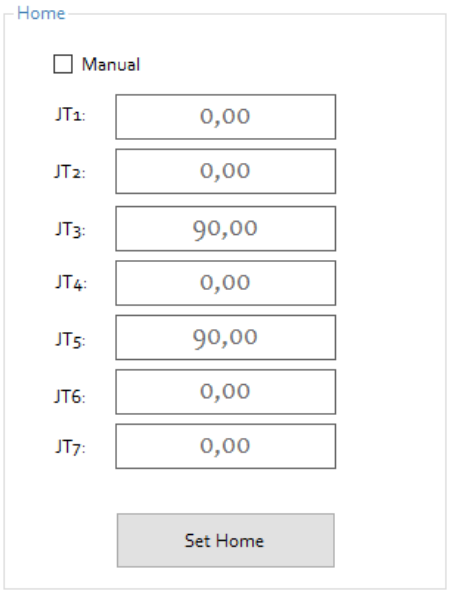
With the [SAVE] button, the data in the robot memory or in a *.loc file on the in a *.loc file on the PC.

The [DELETE] button removes the selected item from the astorino software and from the robot memory at the same time.

16.7 Home/Tool



16.7.1 Home

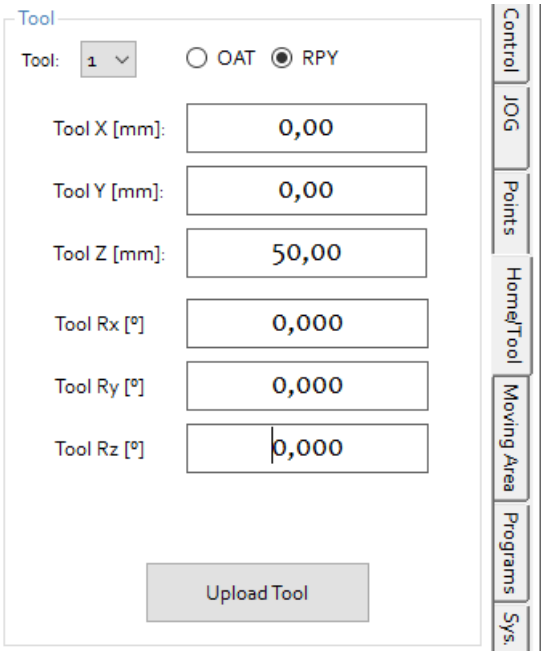


The [Set Home] button saves the current position of the robot as the home position.

Selecting ☐ Manual allows manual entry of data for the HOME position. Pressing [Set Home] button saves the Home position data to the robot's memory.

Pressing the [Home] button on the Control tab causes the robot to return to the saved position in the future!

16.7.2 Tool



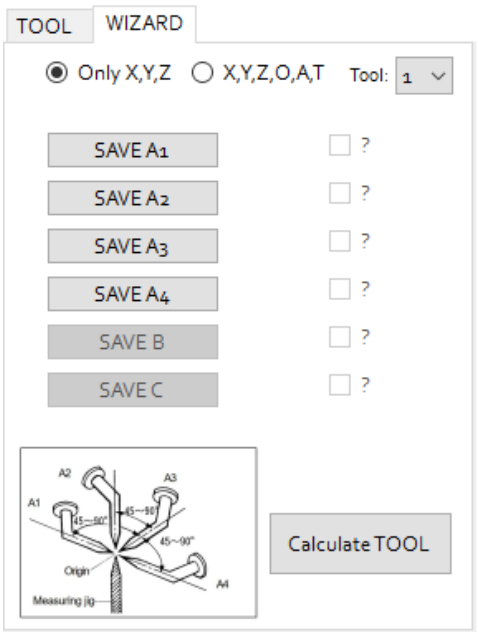
Various tools such as grippers, tips or other can be called up and parameterized via the tool drop-down menu.

Enter the associated tool data manually.

Either this data is already known and documented from the design, or must be determined and entered manually.

Clicking on the [Upload Tool] button saves the entered data to the robots memory.

16.7.3 WIZARD

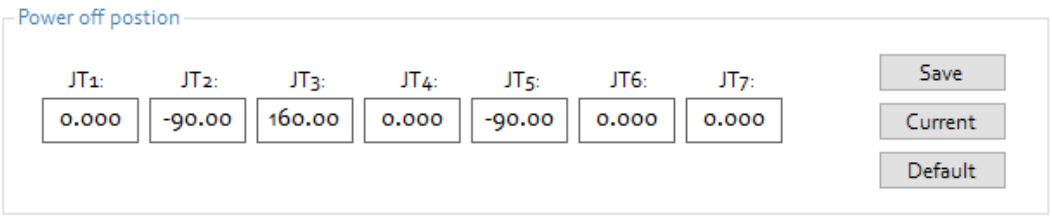


This section allows a user to calculate a new TCP (Tool Center Point) using a 4 points or 6 points method.

The 4 points method calculates x,y,z values of the Tool.

The 6 points method allows to calculate x,y,z and Rx,Ry,Rz values of the Tool. Refer to the TOOL calculation section in this manual.

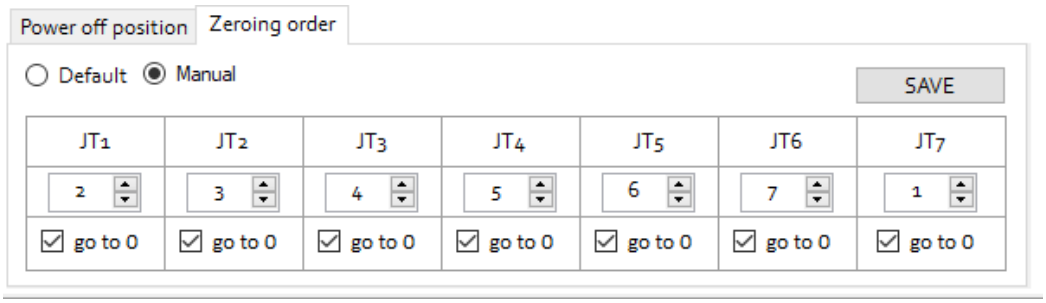
16.7.4 Power off position



This area contains information about the safe power off position.

- The [Save] button saves a manually entered position.
- The [Current] button saves the current robot position as the power off position.
- [Default] resets the values to factory settings.

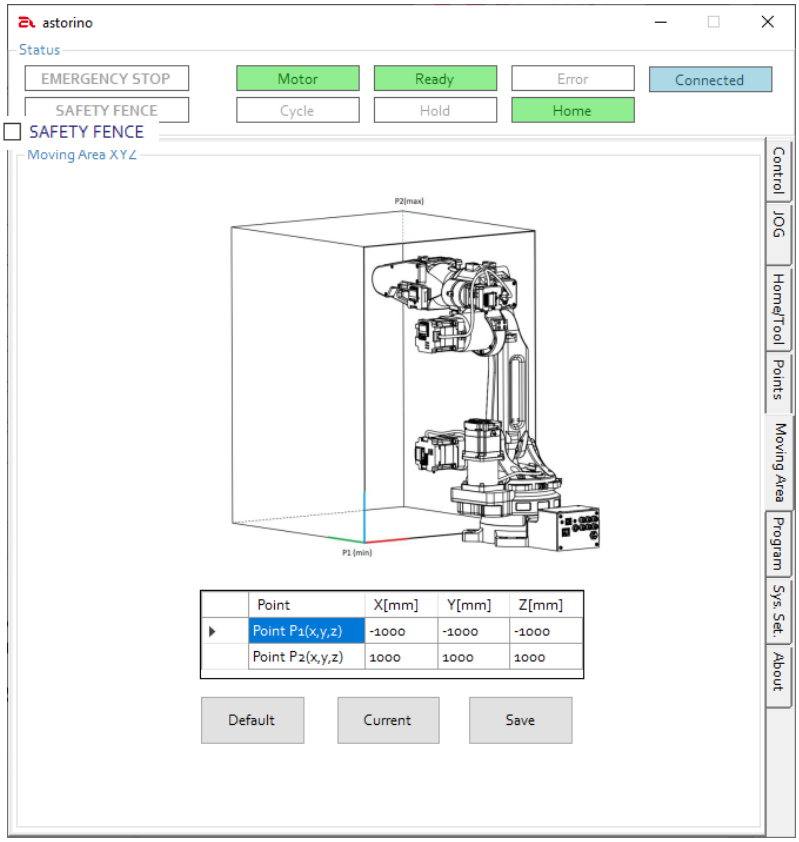
16.7.5 Zeroing order



This area shows information about the zeroing order of axes. The user can set the sequence of zeroing for all axes. Specify [1..7] for the order the axes for all axes (multiple axis can be zeroed in the same step) and select if the axes should or should not go to 0 (zero) position after location is found.

- The [Default] button sets zeroing order to default order.
- The [Manual] button activates the manual settings section.
- The [Save] button is used to save a manually entered position.

16.8 Moving Area



On the [Moving Area] tab the allowed working area of the Astorino robot can be defined.

To modify the allowed workspace, a higher level of access must be entered. To do this, type the command in the Robot Terminal „z_user 3”

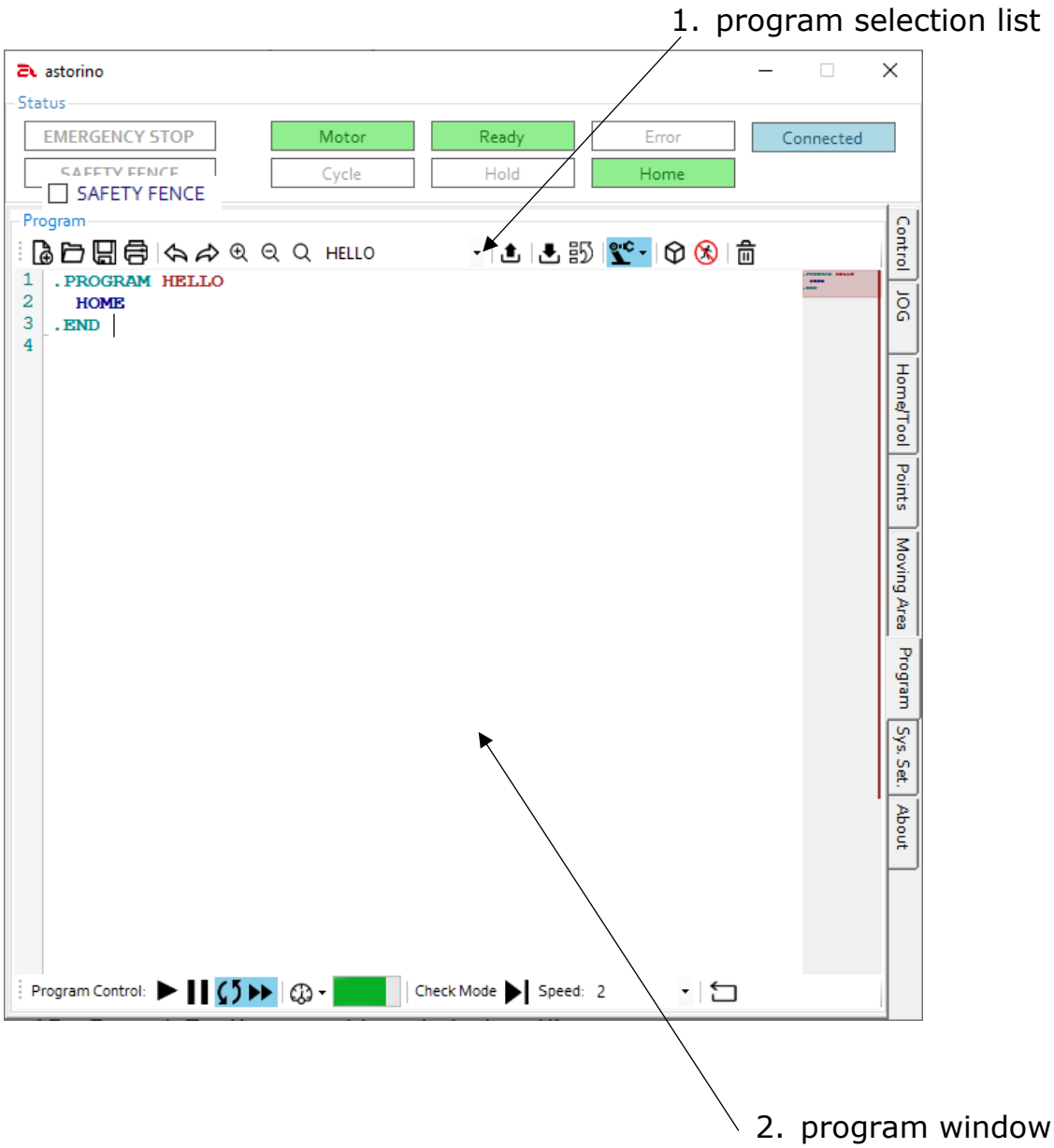
With the help of two points **P1** and **P2**, a virtual rectangular volume is created that defines the area in which the robot is allowed to move.

P1 – shows the minimal values

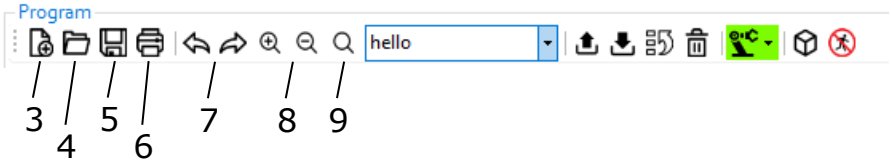
P2 – shows the maximal values

- [Get Current] the current position of the robot is captured and stored in the selected row of the position table.
- The zone can be defined manually by entering values.
- [Default] button resets the values to factory settings.
- [SAVE] the data to the robot memory.

16.9 Programs

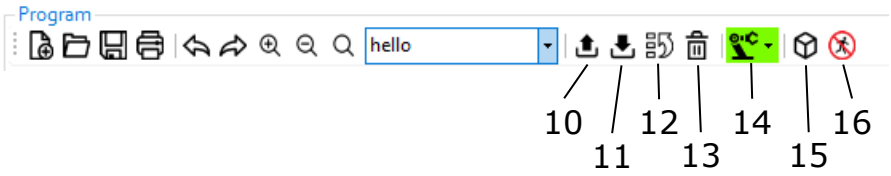


In this tab the robot can be programmed in a simplified version of the Kawasaki AS language.

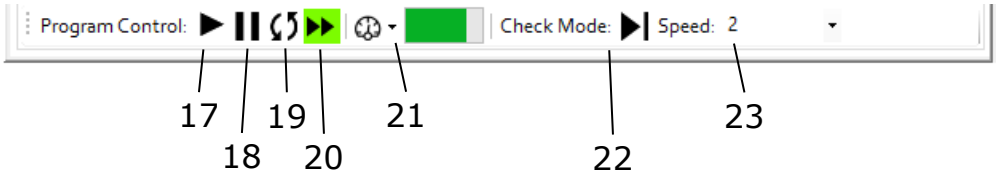


3. Create a new program
4. Select a program from computer and load it on robot
5. Save selected program as *.pg file on PC

6. Print program code
7. Undo / redo
8. Enlarge / reduce contents of program window (zoom)
9. Reset zoom to default setting

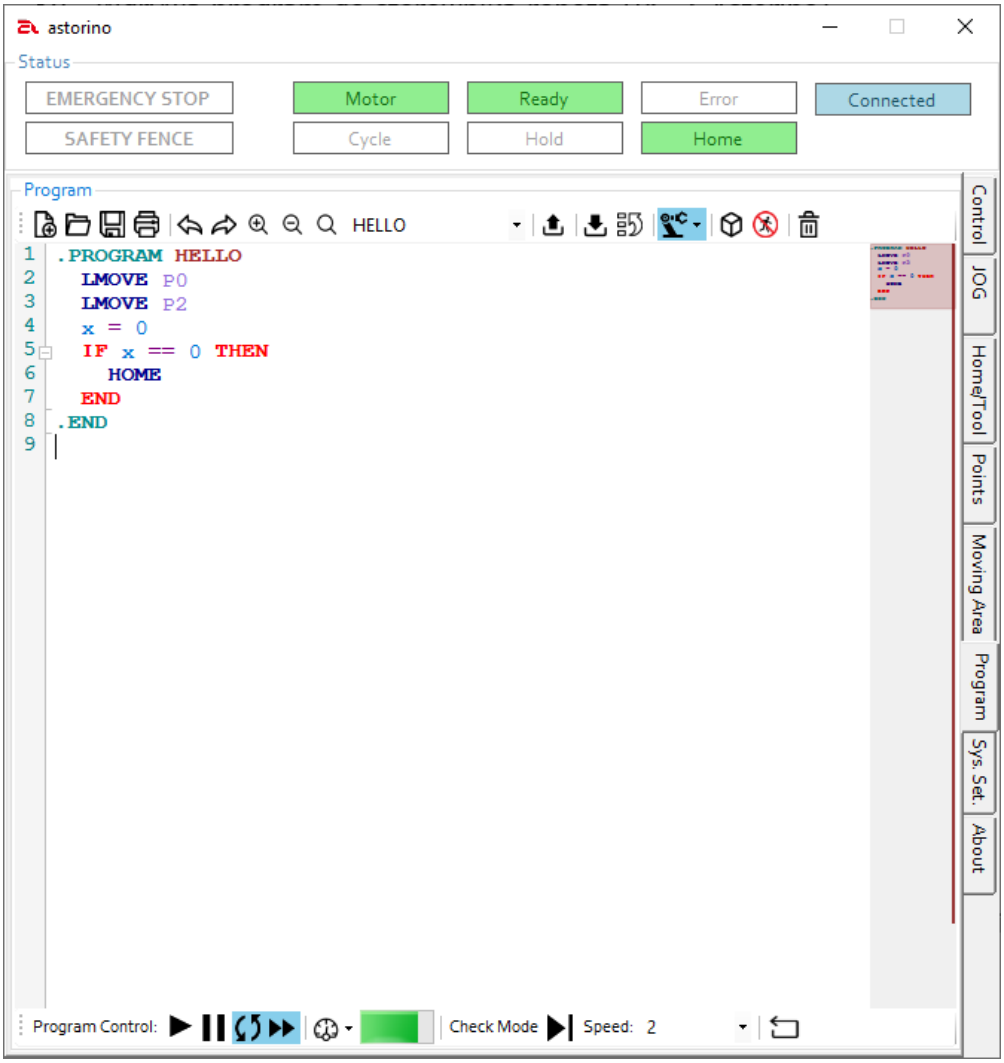


10. Uploads the program to the robot controller (PC ⇨ Astorino)
11. Downloads the selected program from the robot controller
12. Sets the selected program as start program *
13. Deletes the selected program
14. Switching operating mode TEACH / REPEAT (manual / automatic)
15. Open visualization window
16. Activates DryRun mode (without robot movement)



17. Start cycle (program run)
18. HOLD - Stops the running program
19. Repeat continuous - Activates the program loop mode
20. Step continues - Activates the single step mode
21. Monitor speed - Monitor speed
22. Check mode - Displays the currently active mode
    next step in TEACH / REPEAT mode
23. Traverse speed - Set and change in TEACH mode

* When the Astorino is switched on, the program defined as the startup program is loaded into the robot controller's working memory and is directly ready to be executed.



Sample program named hello

If the program is executed, the robot moves to point P1 in a straight line (Linear MOVE).

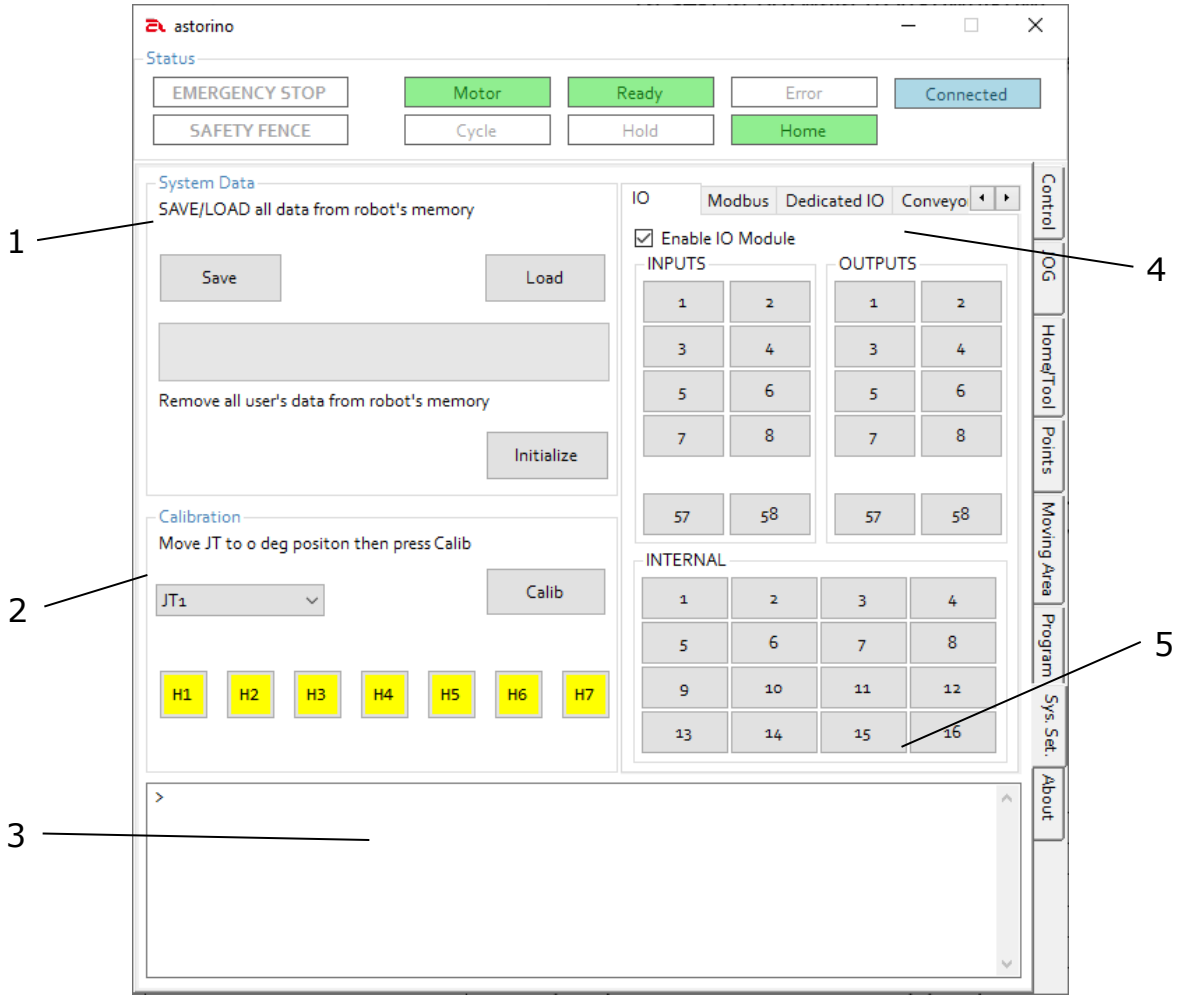
When the point is reached, the robot moves on a direct path to P2.
The variable x is now assigned the value 0.

The IF loop queries if x has the value 0.

If this is the case, the HOME command is executed and the robot moves to home position.

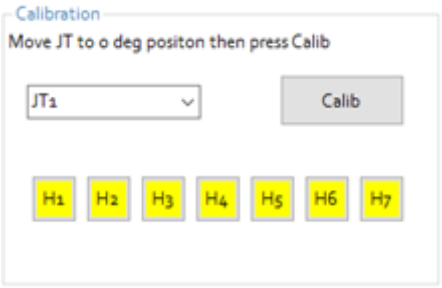
For more information about AS Langage please refere to “astorino – AS language manual”

16.10 System Setting



1. System Data	Save all robot data, load or initialize robot memory.
2. Calibration	Robot arm calibration only required if data on SD card is deleted, card is damaged and replaced, or robot is disassembled.
3. Terminal	Input and display of data.
4. IO, Dedicated IO, Conveyor…	Status of inputs and outputs (I/O), Configure I/O, other settings.
5. INTERNAL	Display and switch internal signals.

16.11 Calibration



This section allows user to calibrate the robot and check the magnetic sensors on the axis. If the buttons with a name Hx, where x is 1..7 are Yellow then the zeroing sensor is active (ON).

16.12 Terminal

The terminal is used to display information from the robot and also give the robot commands.

All move commands like LMOVE, HOME etc. Must be preceded by "DO" and the robot must be READY and in REPEAT mode. For example "DO LMOVE P1".

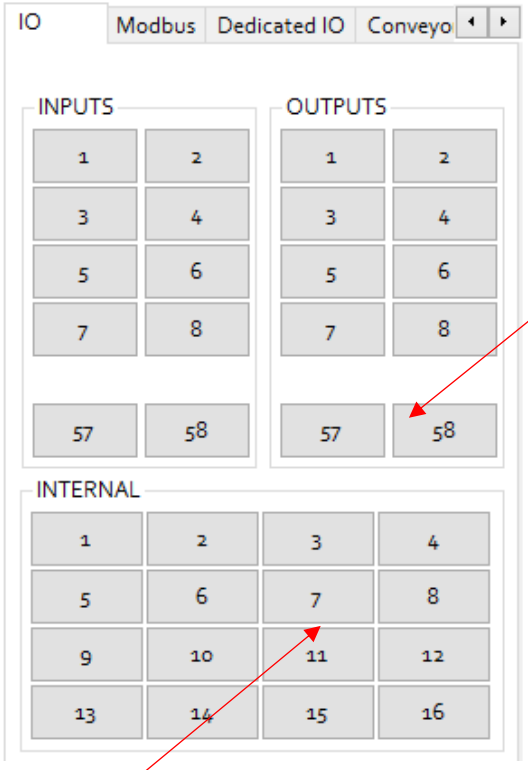
The terminal can be used to read variable values (for example "PRINT x"), teach point(for example HERE P1), set variables (for example x = 10) and so on.

Here is a list of Terminal only commands:

CPUTEMP	Prints CPU temperature
FREE	Prints available RAM memory in %
ERESET	Resets error
ZPOWER ON	Turns MOTORS ON
ZPOWER OFF	Turns MOTORS OFF
HOLD	Pauses the currently running program
CONTINUE/RUN	Continues the paused program
ZZERO x	Starts zeroing of a specific axis - x
HALT	Pauses the currently running program
EXECUTE x	Starts currently selected program, if name specified runs chosen program (x)
PRIME x	Selects chosen program by name
STOP	Stops currently running program (Cycle
REP_ONCE ON/OFF	Turn on or off Repeat Once
STP_ONCE ON/OFF	Turn on or off Step Once
STPNEXT	Triggers next step in step once mode program execution

16.12.1 Status und configuration section

16.12.1.1 IO

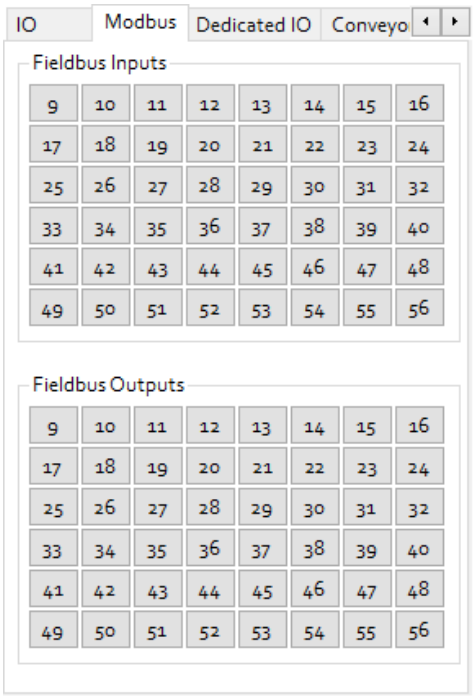


In this window shows the status of the signal INPUTS and switch the OUTPUTS **ON/OFF** by clicking the corresponding buttons.

INPUTS 57,58 and OUTPUTS 57,58 are reserved for B version of the robot and are located on the JT3 arm

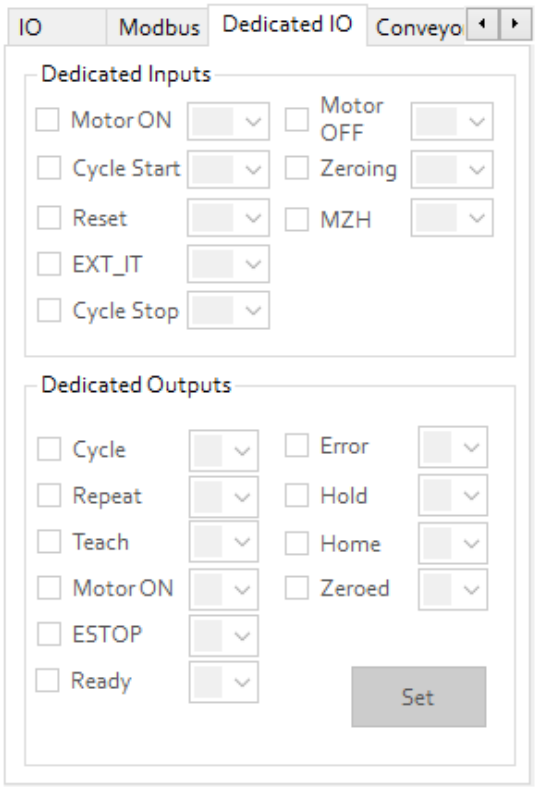
INTERNAL – here you check or force the status of the internal signals. (ON , OFF)

16.12.1.2 MODBUS



In this window shows the status of the Fieldbus Inputs and switch the Fieldbus Outputs **ON/OFF** by clicking the corresponding check buttons. If the INPUT or OUTPUT is ON, the button lights up in yellow colour.

16.12.1.3 Dedicated IO



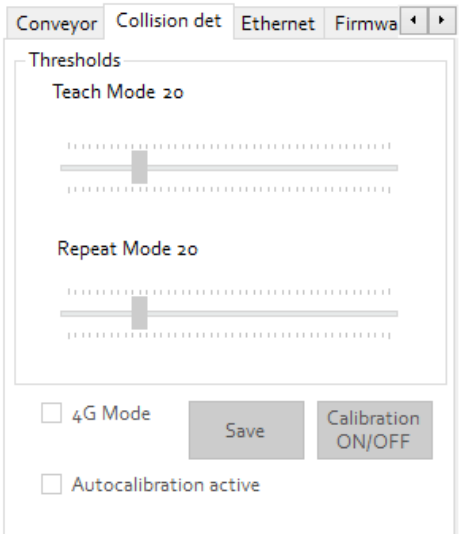
View and configurate dedicated robot signals. The signals have a fixed prescribed function or statement.

Dedicated Inputs are special input signals like "Reset" or "Cycle Stop".

Dedicated Outputs are special output signals like "Motor ON" and "Error".

MZH (MOTOR ON -> ZEROING -> HOME) – this is a special sequence on the astorino designed to HOME the robot with one bit from the POWER ON state.

16.12.1.4 Collision detection (B version of the robot)



Robot is equiped with a accelometer for collision detection.

Here the thresholds for collision detection can be changed.*

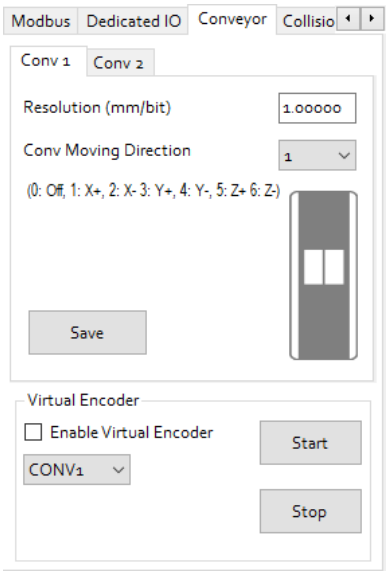
The [4G Mode] checkbox allows the high level thresholds to be turned on/off. Impact sensing is set to maximum of 4G (Earth gravity).

The [SAVE] button saves thresholds to the robots memory.

The [Calibration ON/OFF] button starts or stops the auto calibration of the thresholds.

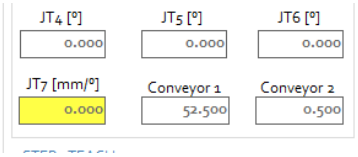
*to change data 3 user level is necessary (type command "Z_USER 3" in the terminall

16.12.1.5 Conveyor



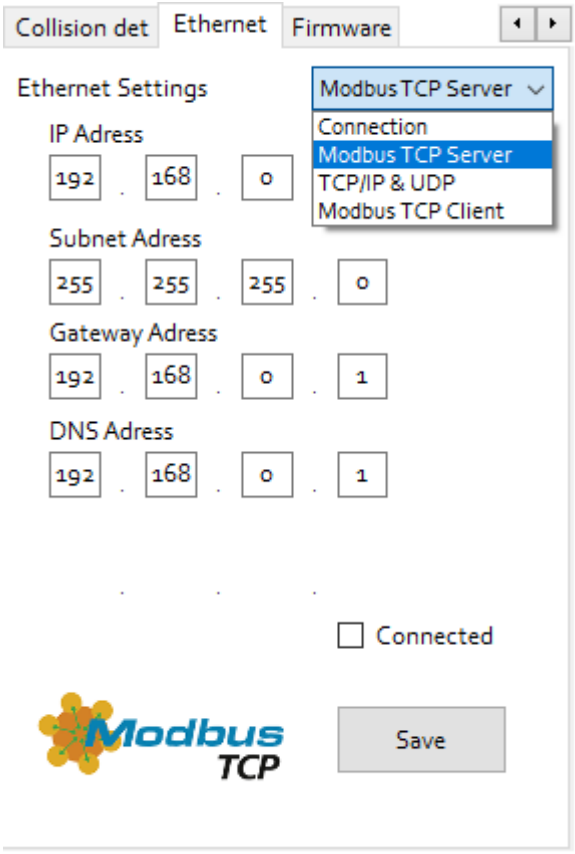
Here you can set the direction of movement of the conveyor belt. We set the direction according to one axis of the BASE system of the robot, as well as the resolution of mm / bit

You can also attach a virtual ecoder. This allows you to simulate applications that use tape tracking. Select from the list which conveyor you want to start (CONV1 or CONV2), attach the checkbox (Enable Virtual Encoder), and then use the [Start] or [Stop] buttons to turn the virtual encoder on or off.



The values of the virtual and physical encoder are displayed from the JOG tab

16.12.1.6 Ethernet



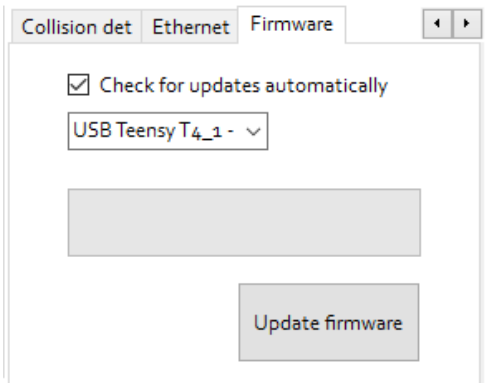
In this area you the settings for Ethernet communication is changed.

The operation of the Ethernet port can be set to:

- Connection to the astorino software
- Modbus TCP Server
- TCP/IP or UDP
- Modbus TCP Client

The [Save] button saves the changes to the memory of the robot.
After saving, a restart of the robot is required.

16.12.1.7 Firmware



In this subtab the Astorino firmware can be updated.

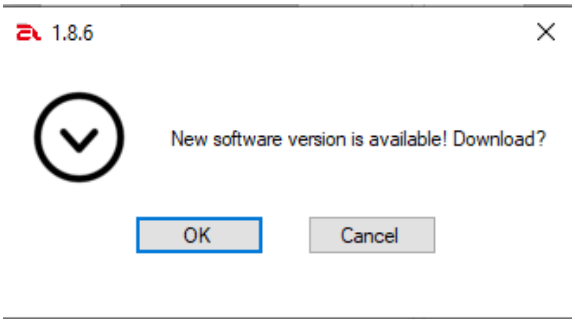
16.13 About

This tab shows the current version of the astorino software and the compatible firmware version to the current astorino software.

16.14 Firmware Update

16.14.1 Basic information

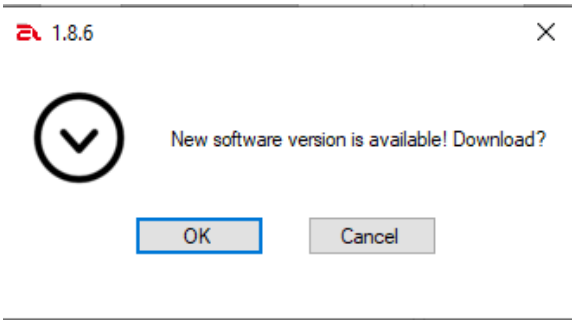
Astorino software after startup will automatically check if there is a new version available and if so, informs user.



Clicking [OK] button will download the new version to user specified location on the hard drive.

Then user needs to uninstall the old version from a PC and install a new one.

After coonecting to the robot astorino software will check if the firmware on the robot is up to date. If the new firmware is available, apllication will inform user about it.



Clicking [OK] button will download the new to user specified location on the hard drive.

The latest firmware version can also be downloaded from KAWASAKI ROBOTICS FTP server: https://ftp.kawasakirobot.de/Software/Astorino/

or contact technical support:
Tech-Support@kawasakirobot.de

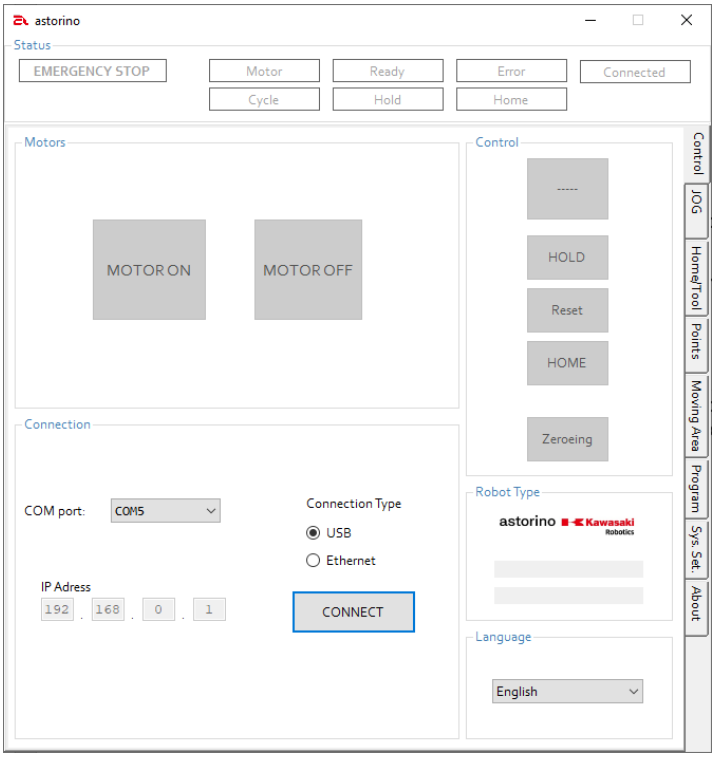
16.14.2 Update procedure

To update the firmware, start the astorino software. Connect the robot to the PC via USB cable.

[**ATTENTION**]

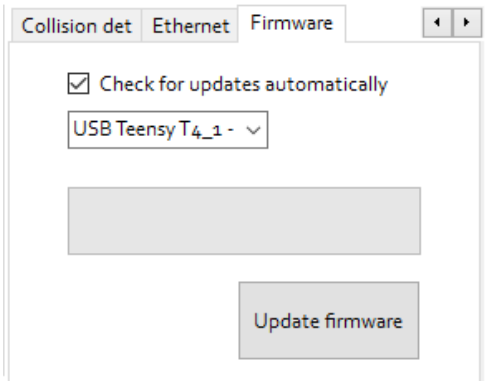
Make sure that the software is **not** connected to the robot. The motors must be switched off! Connect button was not pressed.

Interrupting the process might damage the CPU, please do not turn off the robot during update procedure!



Navigate to the [System Setting] Tab and to the IO configuration area.

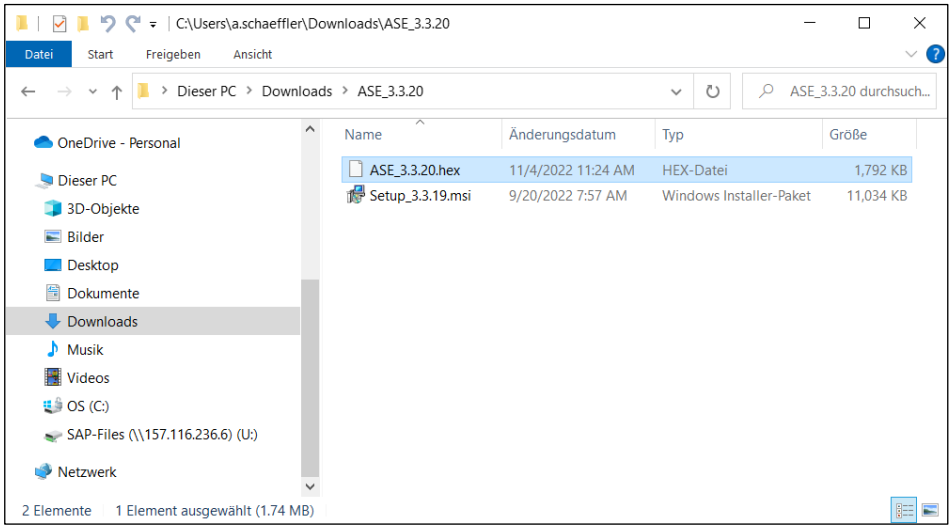
Click on the right arrow symbol until the [Firmware] subtab is visible.



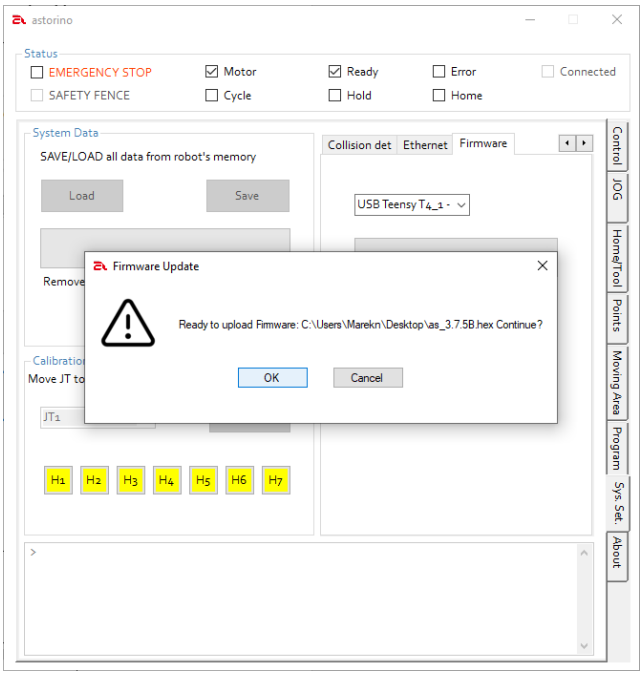
Press [Update Firmware] to open the file selection window.

Select the *.hex file which contains the new firmware.

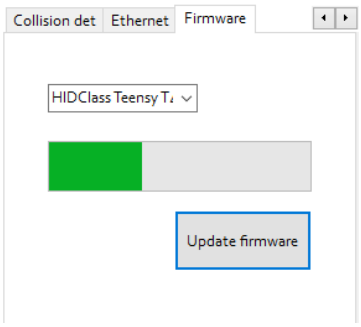
File selection window:



Confirm the upload to start loading the new firmware to the robot's memory:

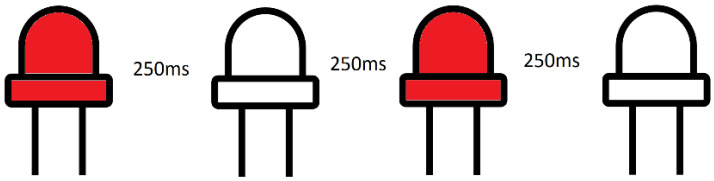


The firmware update is being performed.



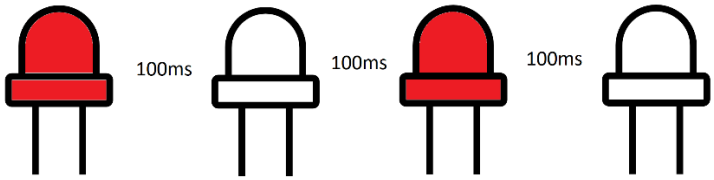
After the firmware is installed observe the red (Error) led on the robots base.

If the red led starts to flash slowly (around 2 times per second).

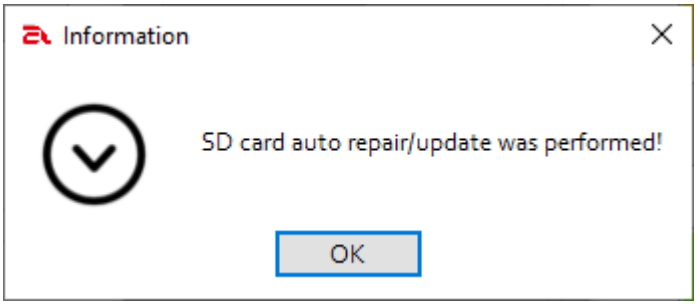


Turn off the robot and turn it on again. This is an error from the SD card inside robots base, CPU was not able to restart the card after firmware update. Resetting the power solves the problem.

If the red led starts to flash fast (around 5 times per second).

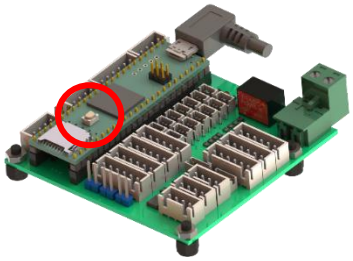


This means that the robot needs to update the data on the SD card. The procedure will perform automatically. When the procedure is complete, the red LED will turn off. And when connected to the computer, there will be a message that the SD card data repair/update procedure is complete

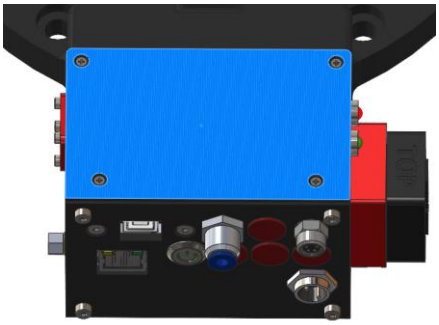


16.15 Update fail recovery

If the update process is interrupted then the CPU might not work and Windows will not detect the robot. Reset the CPU to the factory settings by pressing the white reset button on the CPU board for 13s to 17s.



The white button is located on the CPU-board inside the robot base.To access it unscrew and remove the rear-top cover.



Do not use any metal objects inside the robot base during CPU factory reset.

Red led on the CPU unit will start to blink. Once the hard reset is finished the orange led on the CPU board will blink slowly and the red (Error) led on the robot base will blink slowly.

16.16 AS-language

The astorino can be programmed using a basic version of the Kawasaki AS language, which is used on all Kawasaki Robotics industrial robots.

Current list of supported commands and functions:

(x,y,z represent values - e.g. SPEED 100 ALWAYS
p for points or point names - e.g. JMOVE P10)

Name	Description
ACCEL x	Robot acceleration in % for next motion command
ACCEL x ALWAYS	Robot acceleration in %
ALIGN	Align TOOL z-axis to closest BASE axis
C1MOVE p	Determines the intermediate point of circular interpolation
C2MOVE p	Moves the robot to point p in circular interpolation by passing through the point specified in the C1MOVE p command; the C1MOVE command must be used before the C2MOVE command
CVCOOPJT x	Sets cooperation between conveyor number 1 or 2
CVDELAY x	The robot maintains the current conveyor position for the time x
CVLAPPRO p,x	Moves in the -Z direction of the tool for a given distance x from point p linearly with conveyor tracking
CVLDEPART x	Moves the robot from current position at a specified distance x from the current position along the -Z axis of the tool with conveyor tracking
CVLMOVE p	Linear motion to the point p with conveyor tracking
CVRESET x	Resets external encoder counter to x value
CVWAIT x	Waits until external encoder counter gets to x value
DECEL x	Robot deceleration in % for next motion command

DECEL x ALWAYS	Robot deceleration in %
DISTANCE(p,p)	Calculates distance between two points
DLYSIG x,y	Activates signal x (1–8 or int. 2001–2016) after y time passed in seconds
DRIVE x,y,z	Move single axes, by x- axis, y – degree, z - speed
DRAW x,y,z	Linear motion with respect to x,y,z according to BASE
$DECODE(x,y)	The function searches the string x for the separator y and extracts all characters that are before the separator. These characters are output again as a string and at the same time removed from the original string!
$ENCODE(x)	Changes number to a string
ERESET	Reset error
EXISTCOM	Status of HOST communication data ready
HERE p	Save the current position of the robot to point x
HOME	Moves the robot to HOME position
INRANGE(p)	Checks if point is in range of a robot arm
JAPPRO p,x	Moves in the Z direction of the tool a certain distance x from the joint p
JUMP p,x	Special command: JUMP to position p, where x is a joint or a Cartesian point, x corresponds to the stroke height.
JMOVE p	Motion of the robot along the p (joint) position, where p is a joint or Cartesian point
LMOVE p	Linear motion to the point p
LAPPRO p,x	Moves in the Z direction of the tool for a given distance x from point p linearly
LDEPART x	Moves the robot from current position at a specified distance x from the current pos. along the Z axis of the tool
POINT p	Creates variable x of the point
PRINT x	Print data/text on the Terminal

PULSE x,y	Activates signal x (1–8 or int. 2001–2016) for y time (sec.)
SEND x	Send data to HOST (Serial communication)
SHIFT(p BY x,y,z)	Creates a new point based on the displacement of p Example: POINT TST = SHIFT(P1 BY 10,0,0)
SIG(x)	Checks the state of the x signal — returns TRUE or FALSE Example: IF SIG(2001) == TRUE THEN
SIGNAL x	Activates signal x (1–8 or int. 2001–2016)
SIGNAL –x	Deactivates signal x (1–8 or int. 2001–2016)
SPEED x	Robot speed in % for next motion command
SPEED x ALWAYS	Robot speed in %
SPEED x MM/S	Robot speed in mm/s (max. 250 mm/s) for next motion command
SPEED x MM/S ALWAYS	Robot speed in mm/s (max. 250 mm/s)
SWAIT x	Pauses the program until the high state of the x signal (1-8 or 2001-2016)
SWAIT -x	Pauses the program until the state of low signal x (1-8 or 2001-2016)
TDRAW x,y,z	Linear motion with respect to x,y,z according to TOOL
TOOL p	Selecting tool data from point transformations
TOOL x	Selecting one of the TOOL systems (x = 1,2,3)
TWAIT x	Pauses the program for x seconds
TYPE x	Print data/text on the Terminal
X= CVPOS	Read conveyor data 1
X= CVPOS2	Read conveyor data 2
X = RECEIVE	Read HOST data from buffer
Y = VAL(x)	Changes string value to a number

- Conditional expressions:
    - IF ... THEN … ELSE … END
    - IF … THEN … END
    - CASE … OF … VALUE … ANY… END

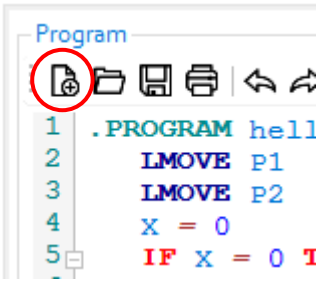
- Loops:
    - FOR … TO … END
    - DO … UNTIL
    - WHILE … END

- Mathematical expressions and functions:
    - +, -, *, /, ^, MOD
    - SIN, COS, ATAN, ABS

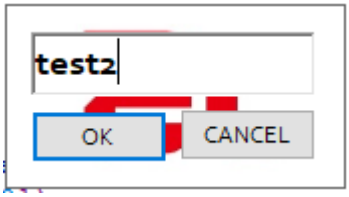
16.17 Programming

16.17.1 Creating a new program

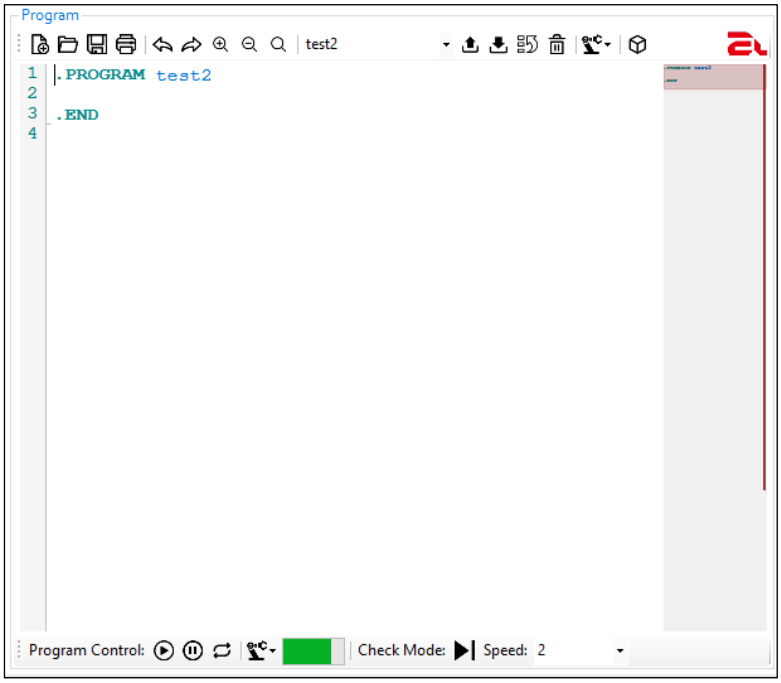
The Program area is located in the Programs tab.
The icon located on the far left creates a new program:



A window will appear to enter a name for the program.

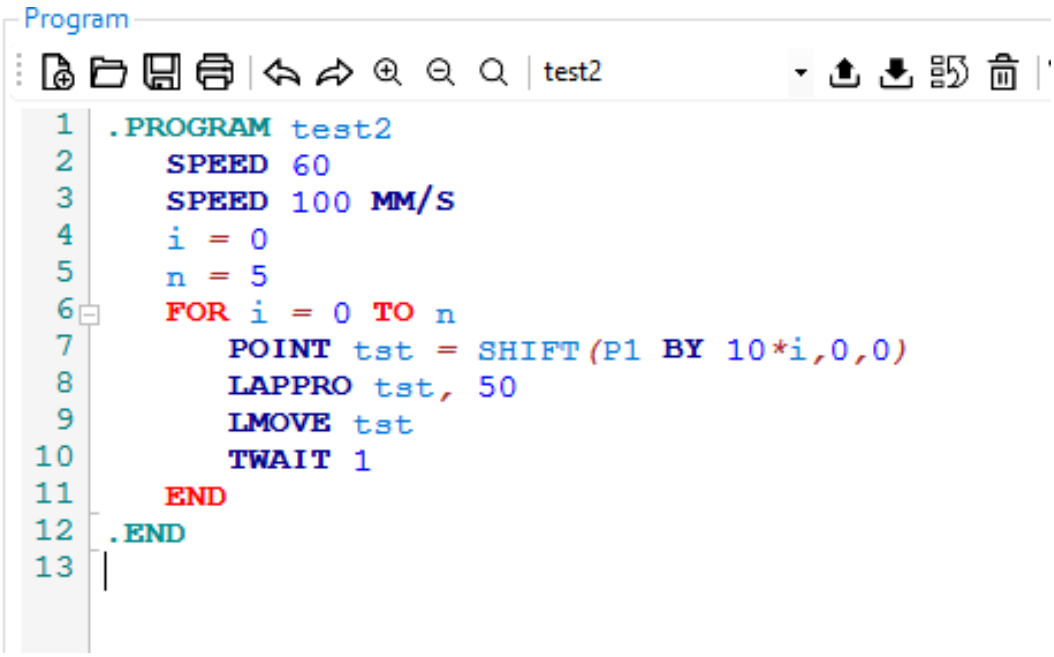


A new program template is generated:



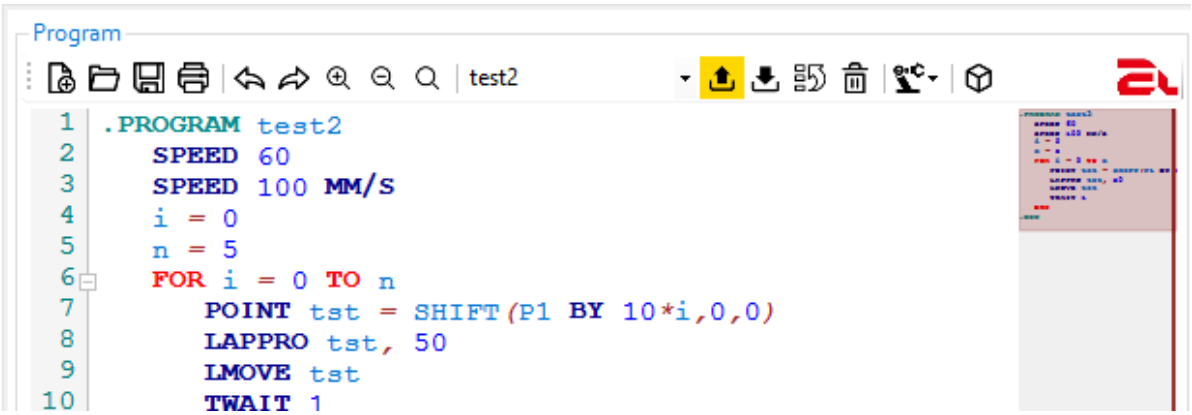
The [Upload] button starts flashing yellow. This is a sign that the program is modified on the PC but not uploaded to a robot.

16.17.2 Write a program



16.17.3 Loading a program onto the robot

To upload the program to the robot's memory press the [Upload] button:

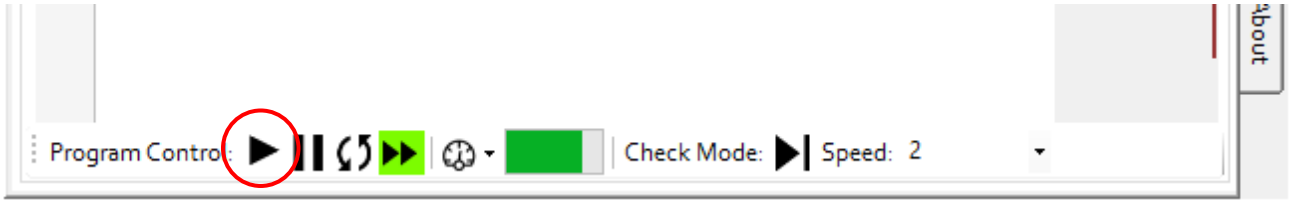


16.17.4 Running a program

⚠ WARNING

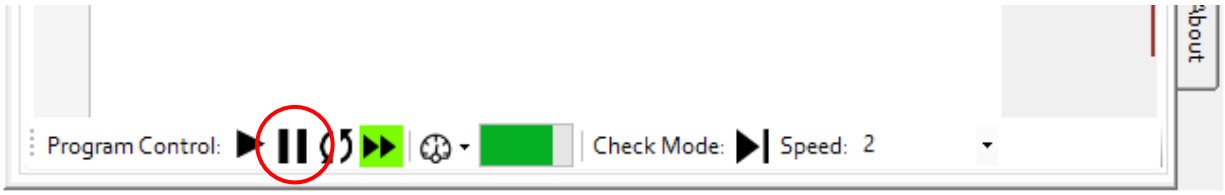
Before running the program make sure that robot will not hit anything. If you are not sure about written program, run it first in DryRun mode!

Click on the Play icon in the Program Control bar ▶ [Cycle Start] to run the program:

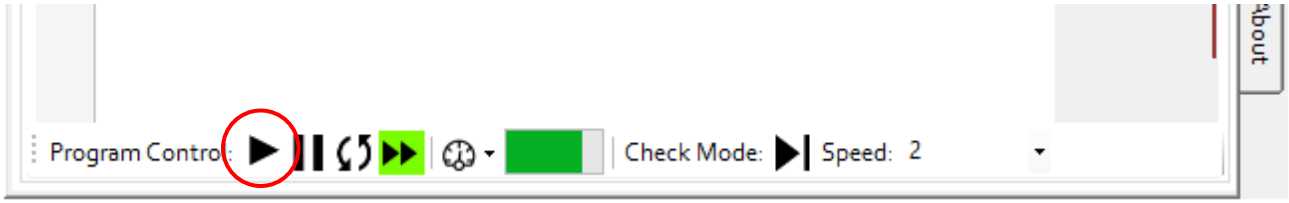


16.17.5 Stopping a program

To stop the execution of the program, you must first pause the robot by pressing the [HOLD]



And then after pausing the robot, click the Play icon on the [Cycle Start] program control bar ▶ to stop the program:



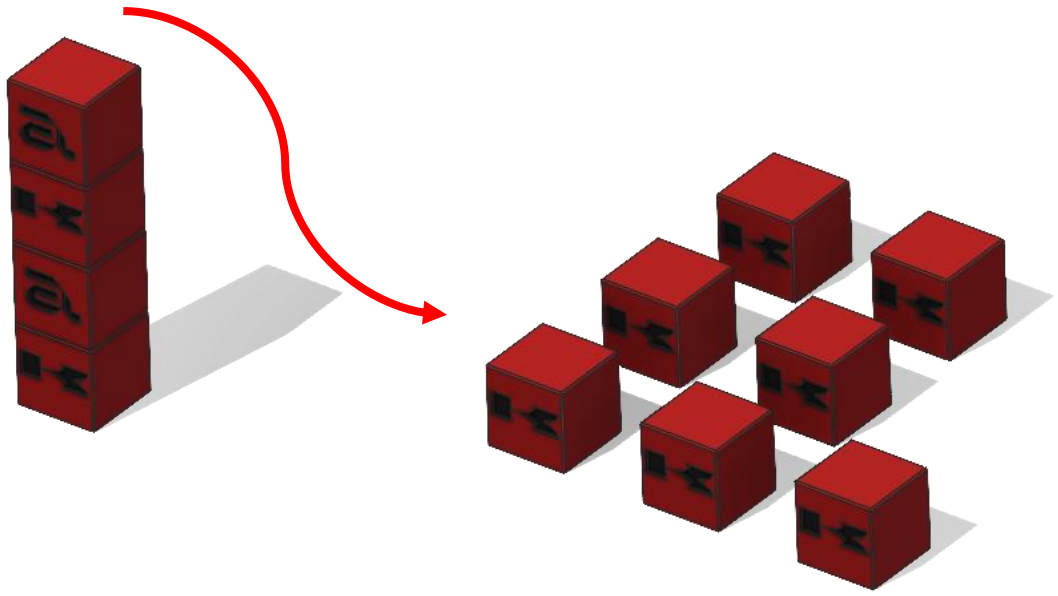
17 Example programs

17.1 Pick & Place – Palletization example

This program picks cubes from single tower and places them into specifiable numbers of: Rows, Columns and Layers.

The user can adjust:

- The size of the workpiece (cubes)
- Distance between the cubes
- The number of rows, columns and layers



```
.PROGRAM PAL1
;-------------- Init ---------
deltaX = 60 ;distance between workpieces X
deltaY = 60 ;distance between workpieces Y
deltaZ = 30 ;layer height
numLev = 2
numRow = 1
numCol = 2
numPcs = numLev*numCol*numRow  ;pieces count
height = 25   ;height of a workpiece (25 mm)
;-------------- variable init ---------
x = 0
y = 0
z = 1
SIGNAL -1
speed 100 mm/s always
POINT place = p2
POINT pick = P1
POINT pick = SHIFT(p1 BY 0,0,numPcs*height)
;P1 on the table, pick shifted by number of pieces in Z
HOME
LAPPRO pick, 40
;---------------------------------- Pal-----------------
FOR z = 0 TO (numLev-1)
  FOR y = 0 TO (numRow-1)       ; rows in Y
    FOR x = 0 TO (numCol-1)      ;col in X
      POINT pick = SHIFT(pick BY 0,0,-height);calc new pick pose
      JAPPRO pick,40
      speed 20 mm/s
      LMOVE pick
      TWAIT 0.5
      SIGNAL 1 ;close the gripper
      TWAIT 0.5
      LDEPART 50
      LMOVE P3
      POINT place = p2
      POINT place = SHIFT(p2 BY deltax*x,deltay*y,deltaz*z)
      LAPPRO place,30
      speed 20 mm/s
      LMOVE place
      TWAIT 0.5
      SIGNAL -1 ;open the gripper
      TWAIT 0.5
      LDEPART 30
      LMOVE P3
    END
  END
END
.END
```

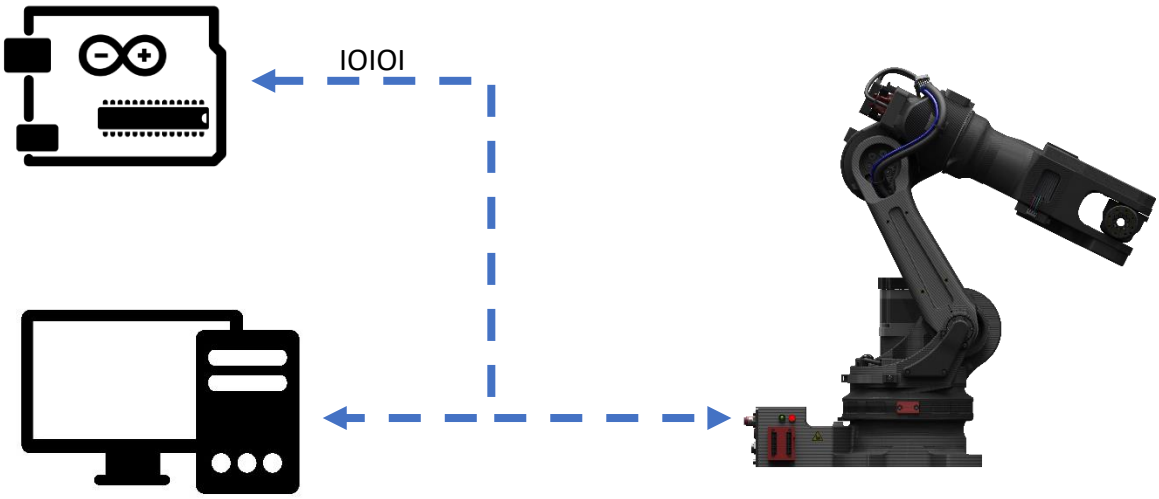
17.2 I/O example program

This example program shows how to use signals in multiple ways.

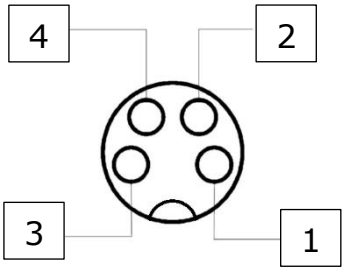
```
.PROGRAM IO
; ----- IO example program
; ----- Robot reads and sets IOs
  sensor = 1002 ;sets variable
  SWAIT 2001 ;wait until internal 1 signal is on
  SIGNAL 8 ;sets 8 output HIGH
  IF SIG(sensor) == TRUE THEN
    ;checks if sensor(2 input) is high
    SIGNAL 2002 ; sets 2 internal HIGH
  ELSE
    IF SIG(1001) == FALSE THEN
      SIGNAL -8 ;sets 1 output LOW
    END
  END
  BITS 1,4 = 12
  ;changes 12 to 4bit binary and sets that on out    puts from 1
  data = BITS(1004,4) ;read binary data from inputs
  ;4 bit from   4th output and changes that to decimal
  PRINT data
.END
```

17.3 Serial communication example program

This example shows how to use serial communication. Programs can exchange data between the astorino robot and PC (for example Matlab or Serial Terminal) or microcontroller (for example an Arduino).



M8 connector (Pins: 1 – GND, 2 - 5V, 3 – TX, 4 – RX)



[NOTE]

The parameters of serial communication are:

- Baudrate: 115200
- Date size: 8
- Parity: None
- Handshake: OFF

```
.PROGRAM SERIAL
; ----- Serial communication example program
; ----- Robot command frame form Serial Port
; ----- frames: P/ or L/x/y/z/
; ----- From X,Y,Z point is created
; ----- Sends current location if frame is P/
  SPEED 150 MM/S ALWAYS
  $S_FRAME = "XYZ"
  $S_FRAME2 = "JT"
  WHILE EXISTCOM == FALSE DO
    TWAIT 0.1
  END
  $TEMP = RECEIVE
  $COMMAND = $DECODE($TEMP,"/")
  PRINT $COMMAND
  ;RECEIVE DATA FROM SERIAL AND CREATE A POINT
  IF $COMMAND == "L" THEN
    $VAL1 = $DECODE($TEMP,"/")
    $VAL2 = $DECODE($TEMP,"/")
    $VAL3 = $DECODE($TEMP,"/")
    DATAX = VAL($VAL1)
    DATAY = VAL($VAL2)
    DATAZ = VAL($VAL3)
    POINT TEST = TRANS(DATAX,DATAY,DATAZ,0,0,0)
    POINT/OAT TEST = P0
    LMOVE TEST
    SEND "OK"
  END
  ;SEND CURRENT LOCATION TO SERIAL PORT
  IF $COMMAND == "P" THEN
    HERE TEMP
    HERE #TEMP
    DECOMPOSE TAB[0] = TEMP
    DECOMPOSE TAB2[0] = #TEMP
    FOR I = 0 TO 5
      TAB2[I] = TAB2[I]*180/PI
      $S_FRAME = $S_FRAME + $ENCODE(TAB[I]) + "/"
      $S_FRAME2 = $S_FRAME2 + $ENCODE(TAB2[I]) + "/"
    END
    SEND $S_FRAME
    SEND $S_FRAME2
  END
.END
```

⚠ WARNING

Serial communication is working with 3.3V, use electronics compatible with 3.3V or use level shifters.

Voltage level of 5V will damage the CPU unit!

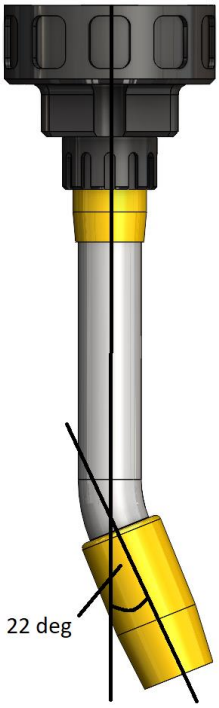
18 Tool Data

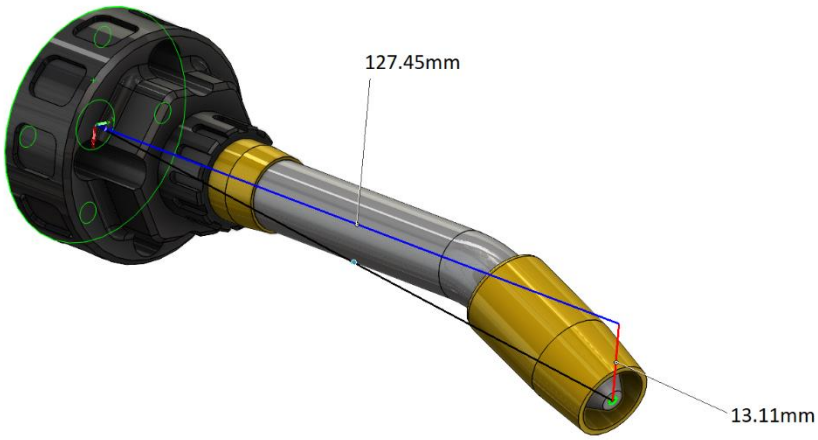
18.1 Tool data from known dimensions

Enter the dimensions of the tool (TOOL DATA) in the base coordinate system (BASE) of the robot.

Obtain the tool dimensions from existing CAD data or physicaly measure them .

The following is an example of how to obtain tool data for an arc welding torch.





The burner is angled at 22 degree.

The length of the torch from the flange surface (axis 6) to the tip of the nozzle is 127.45mm in Z-direction.

The angled torch results in an offset in the Y-direction of 13.11mm.

The measurements for the TCP are always from the center of the robots flange.

This tool example has the following TCP coordinates.

X[mm]	0.0	
Y[mm]	13.11	
Z[mm]	127.45	
Rx: -22,0	[O: 90.0]	Rx is the rotation around the X axis
Ry: 0.0	[A: 22.0]	Ry is the rotation around the Y axis
Rz: 0,0	[T: -90.0]	Rz is the rotation around the Z axis

18.2 Automatic Tool (Coordinates Data) Registration

This chapter describes operation procedures for automatic registration of tool coordinates values.

[ATTENTION]

Automatic tool registration is a kind of teaching. Its usage is limited to personnel who have completed special training and are qualified for teaching or supervising robot operations.

18.2.1 Overview of Automatic Tool Registration Function

A variety of different shaped tools (gripper, hand, etc.) can be mounted on the robot flange. If the tool data is not measured correctly, the robot motion trajectory will deviate from the taught path.

The tool data is essential for operating the robot correctly. In general, the tool data should be input by numeric values and registered, but in some cases measurement of the position and orientation of the tool coordinates may not be accurate, or take a long time.

This function makes it possible to automatically calculate the tool transformation values by teaching several points in space without having to enter values. For this procedure a measuring jig with sharp point is required, for example a big screw and the cone tool attachment (not delivered with a robot).

18.2.2 Required Data for Automatic Tool Coordinates Registration

When using the automatic tool registration function, the set of pose data is stored (A1,A2,A3,A4,B,C). The pose data measurement is taken by aiming at one teaching point from 4 or 6 different tool poses, as described below.

Tool coordinates data

- When registering only tool coordinates position (X,Y,Z)

  Teaching pose consists of four base poses, A1, A2, A3,

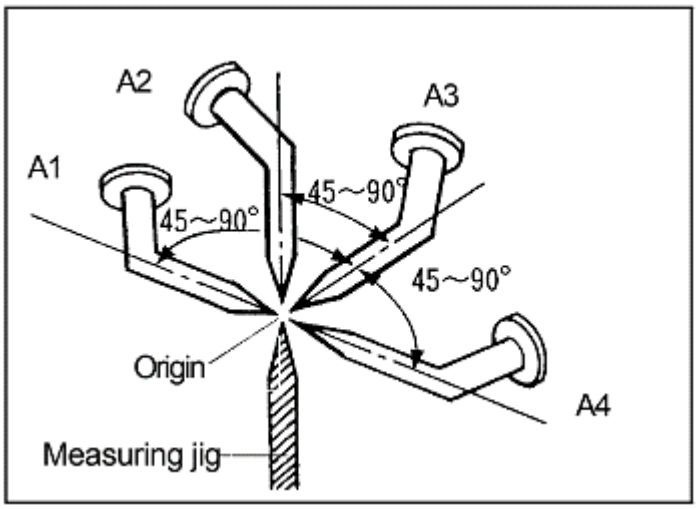
- When registering tool coordinates position (X,Y,Z) and orientation of the tool coordinates (O,A,T/Rx,Ry,Rz)

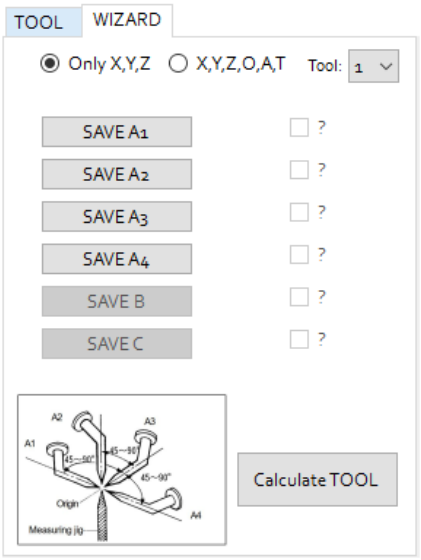
  Teaching pose consists of six base poses, A1, A2, A3, A4, B and C.

18.2.3 Teaching the Four Base Poses

The four position method of TCP calculation determines the offsets of the TOOL data in X,Y,Z dimensions.

Teach the 4 Poses (A1, A2, A3, A4) at the same position data but with different approaches, ensure the Tool tip is at the same point on the measuring jig. Ensure that the angles between each orientation are as large as possible within the preferred range 45° - 90°. If the range is lower than 45° the accuracy of the calculation might be less precise. The wrist flange face should have a different plane for each base orientation. Teach each base pose so that the tool coordinates and measuring jig origins are in contact with each other.



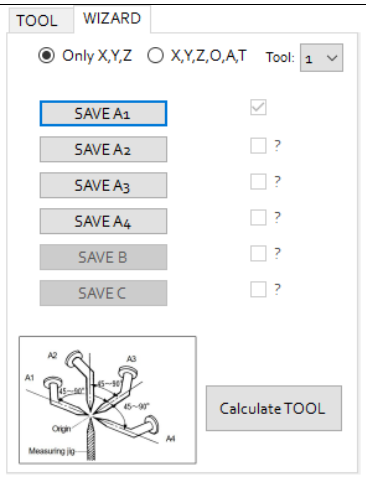


Select [Only X,Y,Z] in HOME/Tool Tab

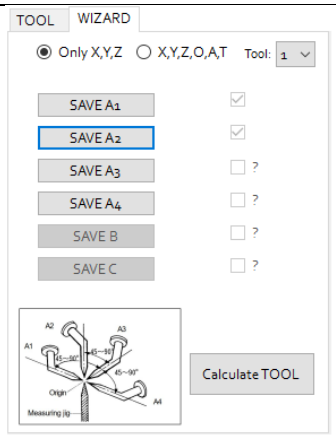
Choose the Tool number to Teach from the list. 1,2 or 3 or be selected.

Switch the robot to Teach Mode and move to positions as below (this is an example, real positions might be different). After each position is reached, press SAVE Ax, where x is 1,2,3 or 4.

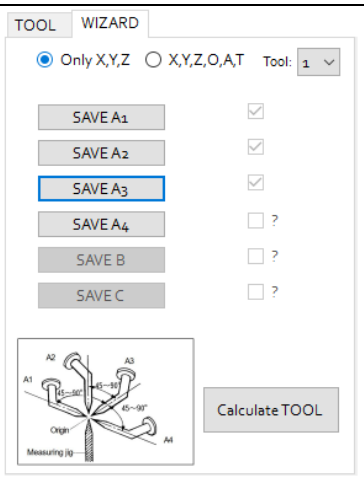
Teach A1.



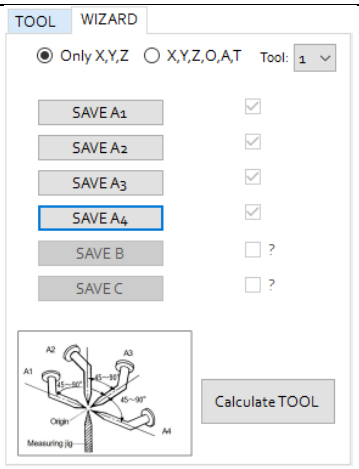
Teach A2.



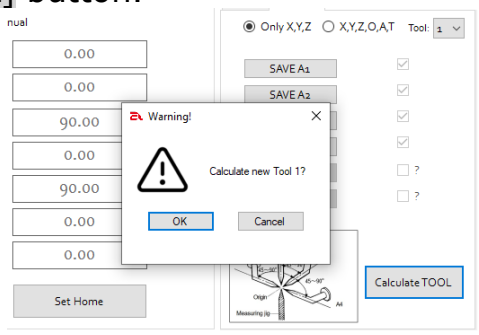
Teach A3.



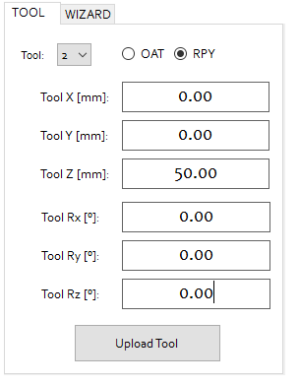
Teach A4.



Press [Calculate TOOL] button.



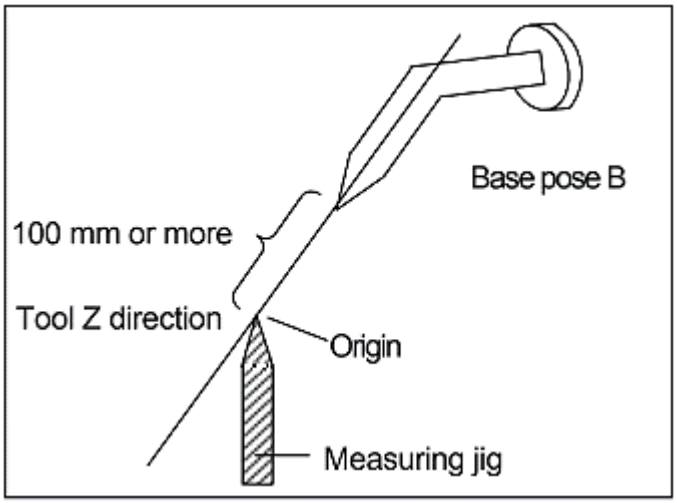
New calculated tool data will be saved on the SD card and displayed on the TOOL tab.



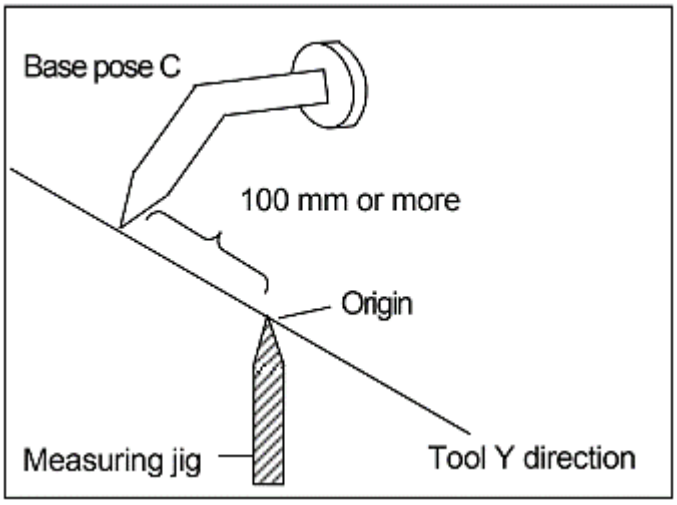
18.2.4 Teaching the Six Base Poses

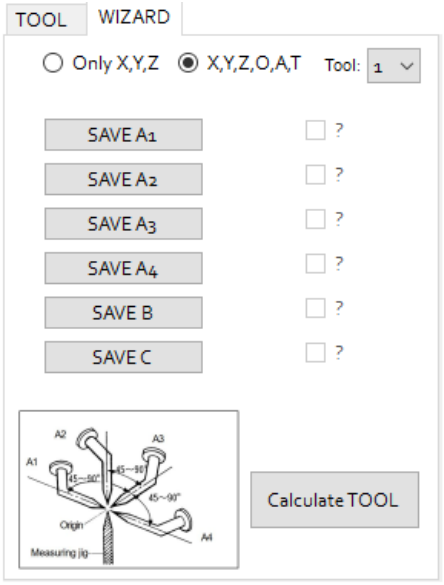
The initial points A1,2,3 & 4 should be taught as referenced in the four poses method.

For Base Pose B, teach so that contact is made between the Measuring Jig's Origin and a position 100 mm or more away from the TCP (tool center point) in the desired -Z direction of the tool.



For Base Pose C, teach so that contact is made between the Measuring Jig's Origin and a position 100 mm or more away from the TCP in the desired +Y direction of the tool.



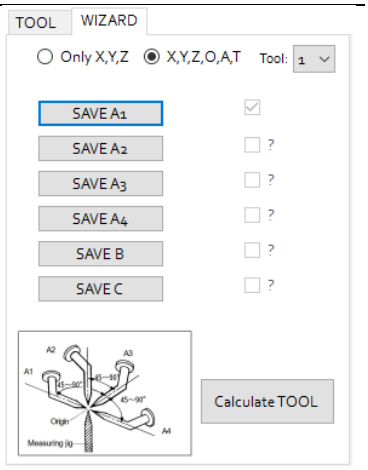


Select [X,Y,Z,O,A,T] in HOME/Tool Tab.

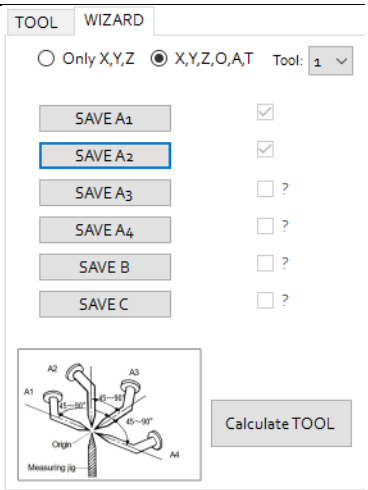
Choose the Tool number to Teach from the list. 1,2 or 3 or be selected.

Switch robot to the Teach Mode and move to positions as below (this is an example, real positions might be different). After position is reached, press SAVE Ax, where x is 1,2,3 or 4.

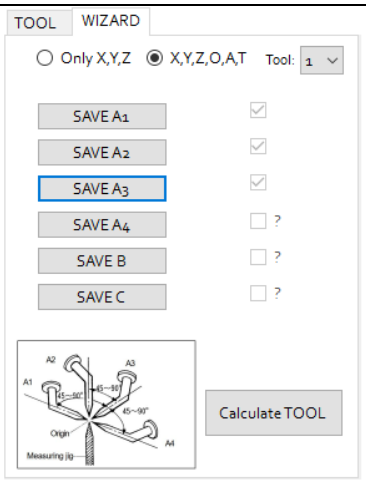
Teach A1.



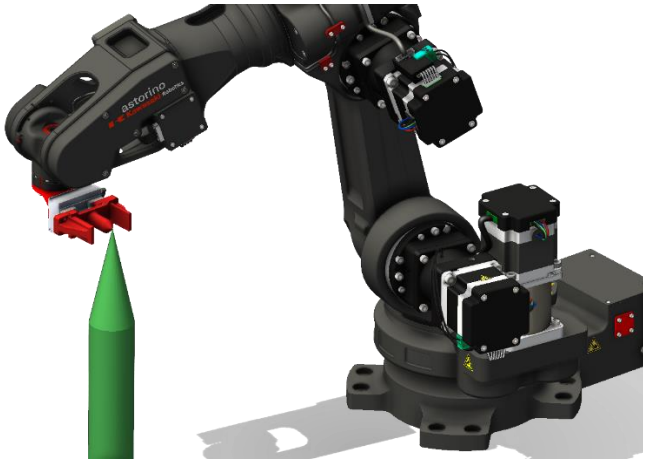
Teach A2.

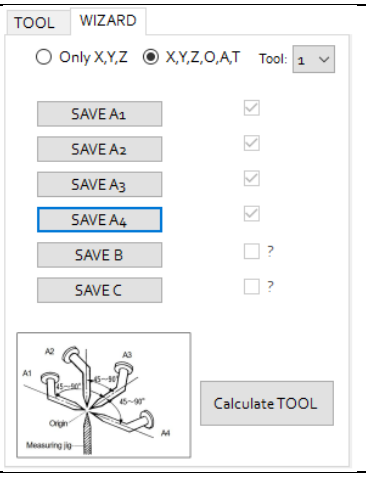


Teach A3.

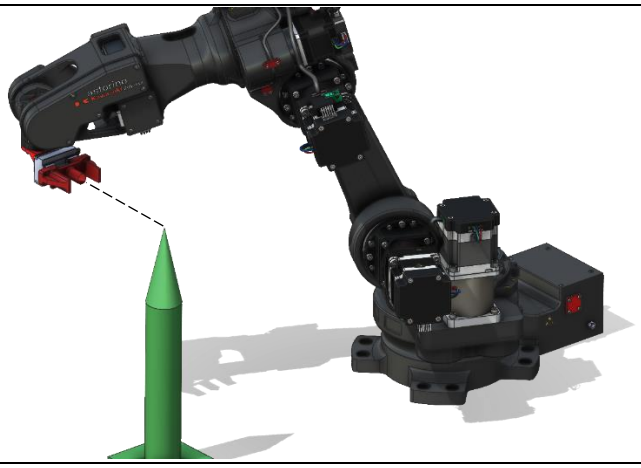


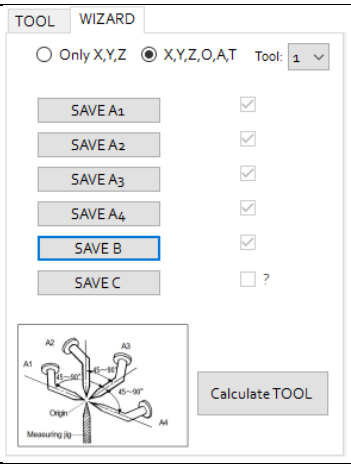
Teach A4.



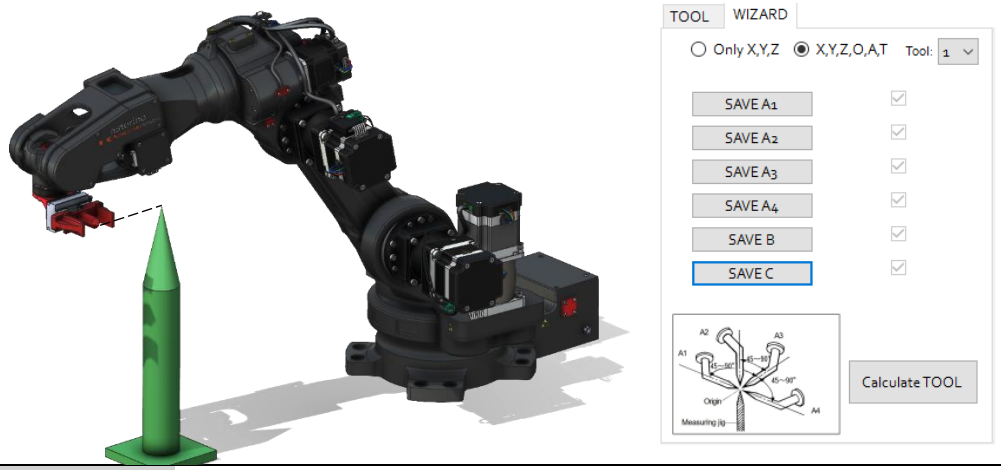


Teach B.

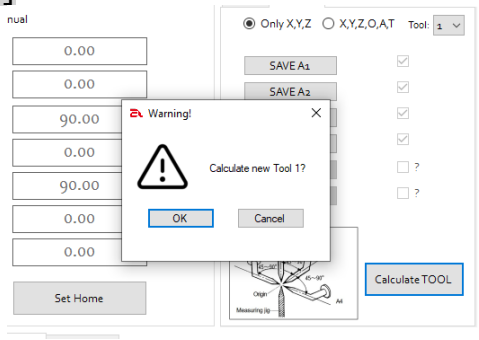




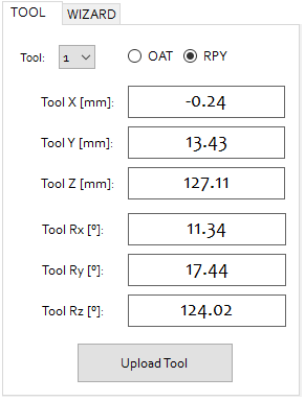
Teach C.



Press [Calculate TOOL] button.



New calculated tool data will be saved on the SD card and displayed on the TOOL tab.



19 Auto-calibration of collision detection

[**ATTENTION**]

Calibration is done for a specific program, changing the program might require to repeat this procedure again!

To auto calibrate the collision detection thresholds go to the Collision change user level to 3.

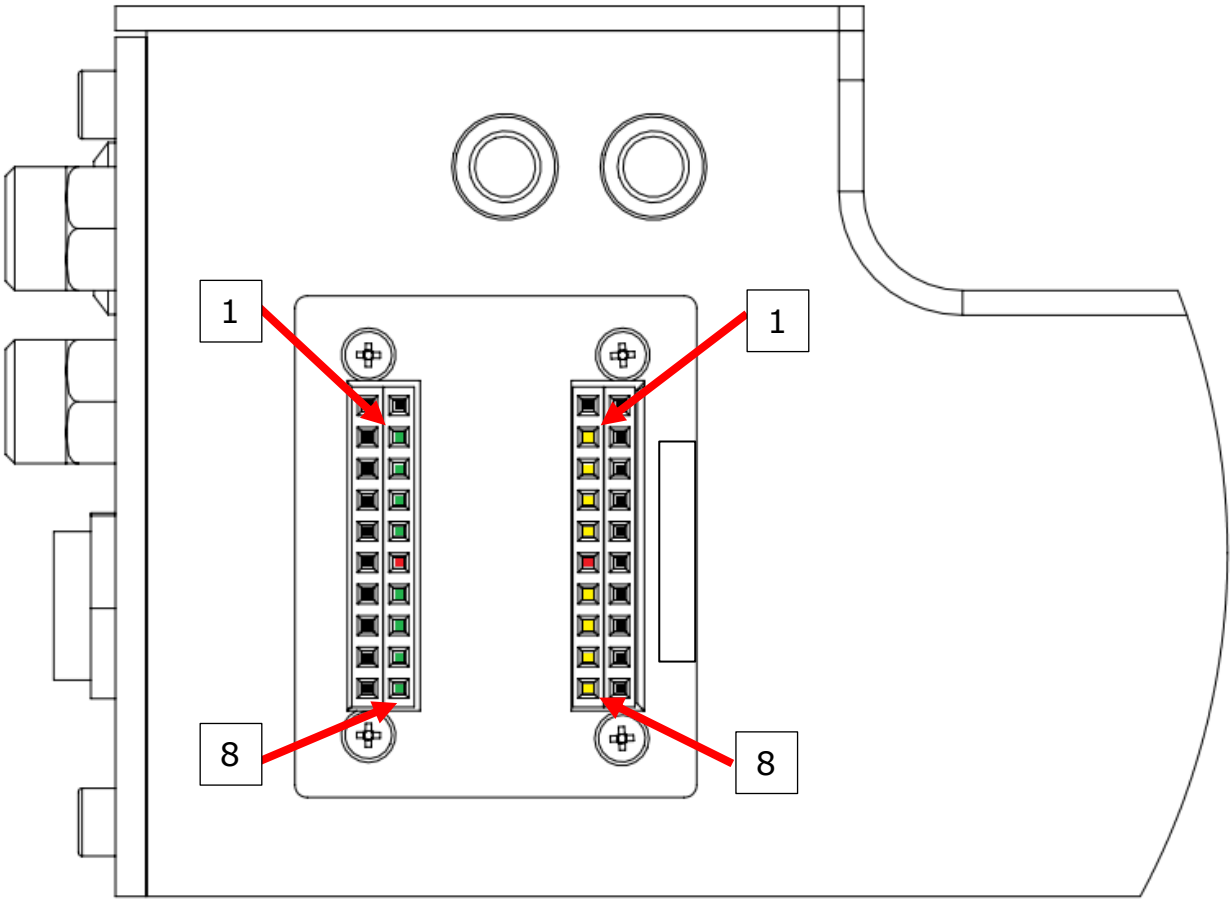
Follow listed below steps:

- Switch robot to Teach or Repeat mode (calibration will be done for recently selected mode),
- Select a program for which you would like to calibrate the sensor,
- Go to Collision detection tab and click [Calibration ON/OFF]
- Run the selected program for a few cycles,
- Stop the program (Cycle off),
- Go to Collision detection tab and click [Calibration ON/OFF],
- New data will be saved to robots memory.

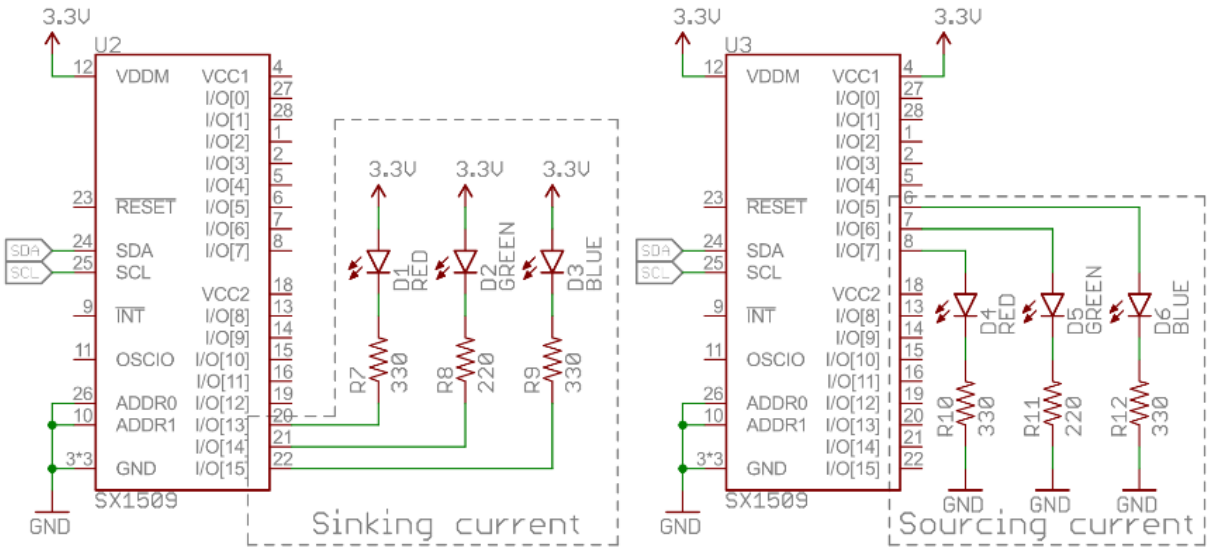
20 I/O – 3,3V

The Astorino has 8 inputs and 8 outputs at 3.3V DC.

Colours	Function
	Output
	Input
	3,3 V/DC
	Ground (GND)



The system normally operates in PNP switching mode (sourcing current). PNP means positive switching (mainly used in Europe and North America). A module therefore switches positive potential to its output.



The operation can be changed to NPN by using the following commands in the terminal:

- Z_OUTSOURCE 1 – SOURCE OUTPUT
- Z_OUTSOURCE 0 – SINK OUTPUT
- Z_INPULL 1 - activates pulling the inputs to 3,3V
- Z_INPULL 0 - deactivates pulling the inputs to 3,3V

WARNING!

Each OUTPUT provides 8mA of current. Please do not exceed the limit as this may damage the motherboard.

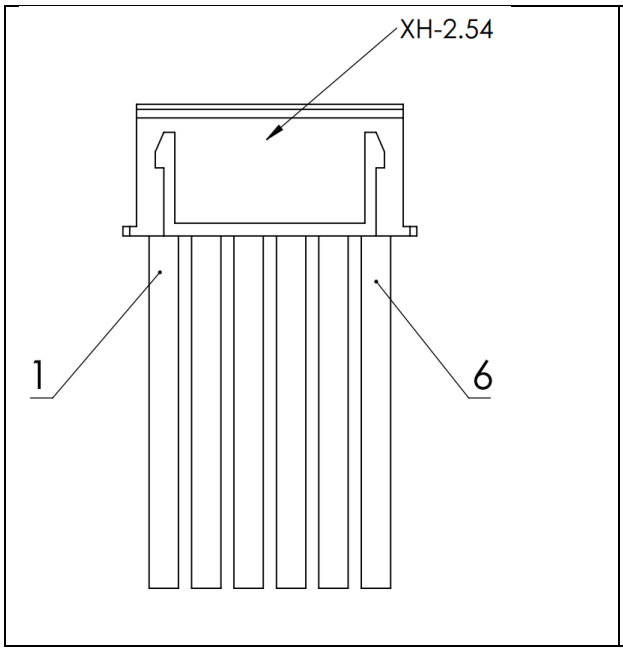
21 ARM INPUTS/OUTPUTS

The B version of the astorino robot is equipped with two 24V inputs and two 24V outputs (PNP) on JT3 of the arm.

In the astorino software and AS system signal numbers 57 and 58 for outputs, 1057 and 1058 for INPUTS.

The connector used is XH-2.54 6 pin female.

Refer to this table for connection the INPUTS/OUTPUTS



1. INPUT 57
2. INPUT 58
3. OUTPUT 57
4. OUTPUT 58
5. 24V
6. GND

⚠ WARNING

Each OUTPUT provides 300mA of current. Do not exceed the limit, it might damage the Main Board.

22 MODBUS TCP

Modbus is a data communication protocol originally published by Modicon (now Schneider Electric) in 1979 for use with its programmable logic controllers (PLCs). Modbus has become a de facto standard communication protocol and is a commonly available means of connecting industrial electronic devices.

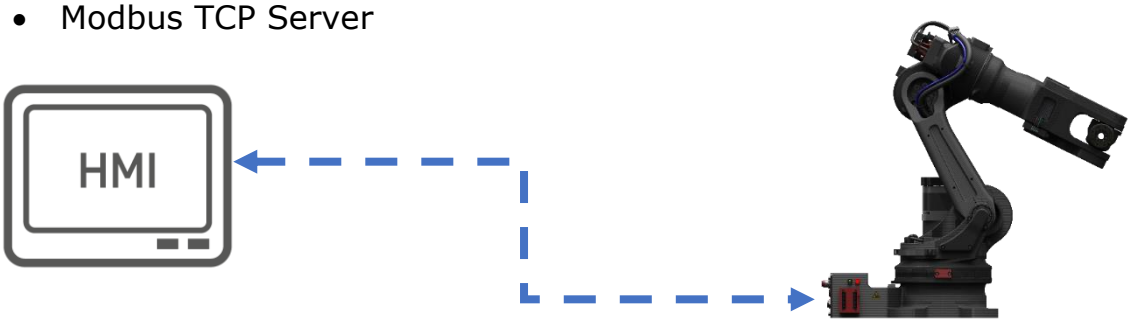
⚠ WARNING

Astorino in Modbus Client settings does not update the register during movement!

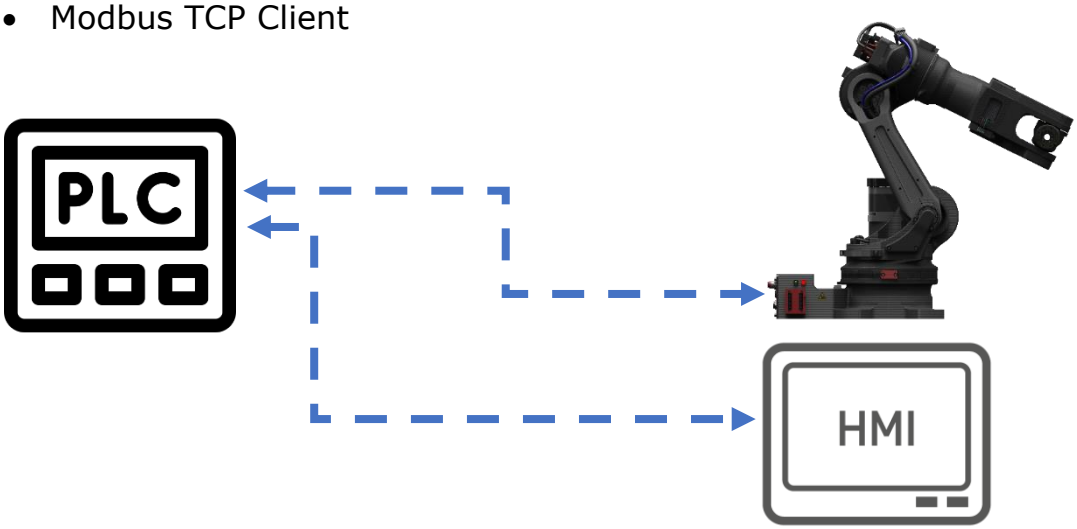
22.1 Modbus network operating modes

Astorino can operate in two network modes. The robot can be set as a server or a client. The signal field remains the same.

- Modbus TCP Server



- Modbus TCP Client



22.2 Modbus object types in astorino robot

The following object types may be provided by a Modbus server to a Modbus client device. The addresses are representative of the original Modicon specification. Under the current standard the address can be 0 - 65535 with the object type identified by the command used to read or write the coil or register. The Astorino robot can read and write 3x Input Registers and 3x Holding registers, that gives additional 56 inputs and 56 outputs.

astorino robot uses standard PORT: 502

astorino as Modbus Server

Object type	Astorino function	Size	Address Space
Input register	outputs	16 bits	30001 – 30003
Holding register	inputs	16 bits	40001 – 40003

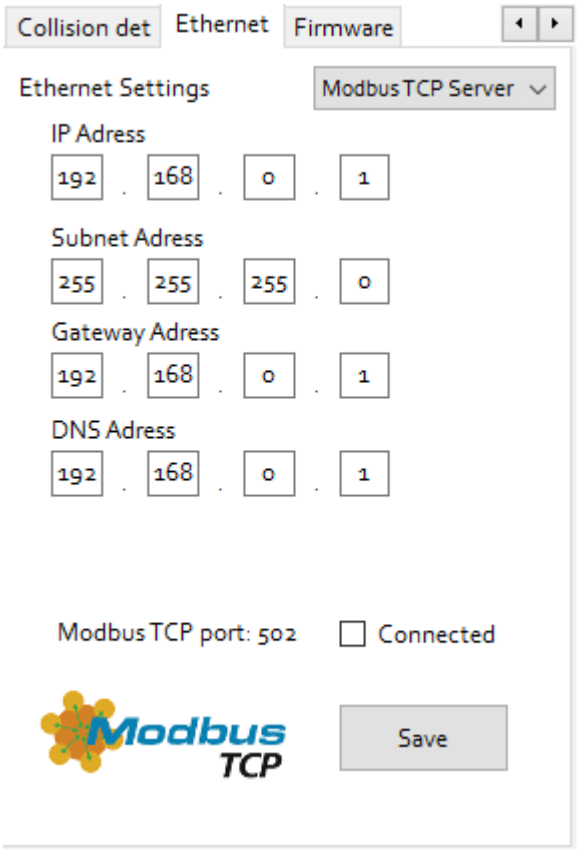
astorino as Modbus Client

Object type	Astorino function	Size	Address Space
Input register	inputs	16 bits	30001 – 30003
Holding register	outputs	16 bits	40001 – 40003

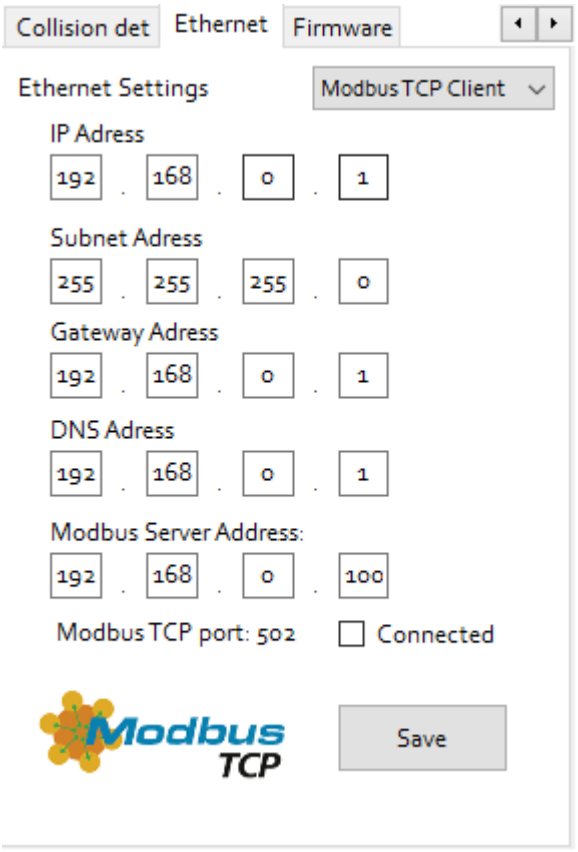
22.3 Configuration of the Ethernet port

Set the network addresses according to your PLC/HMI configuration and set Ethernet Settings to Modbus TCP.

Astorino as Modbus TCP Server



Astorino as Modbus TCP KClient

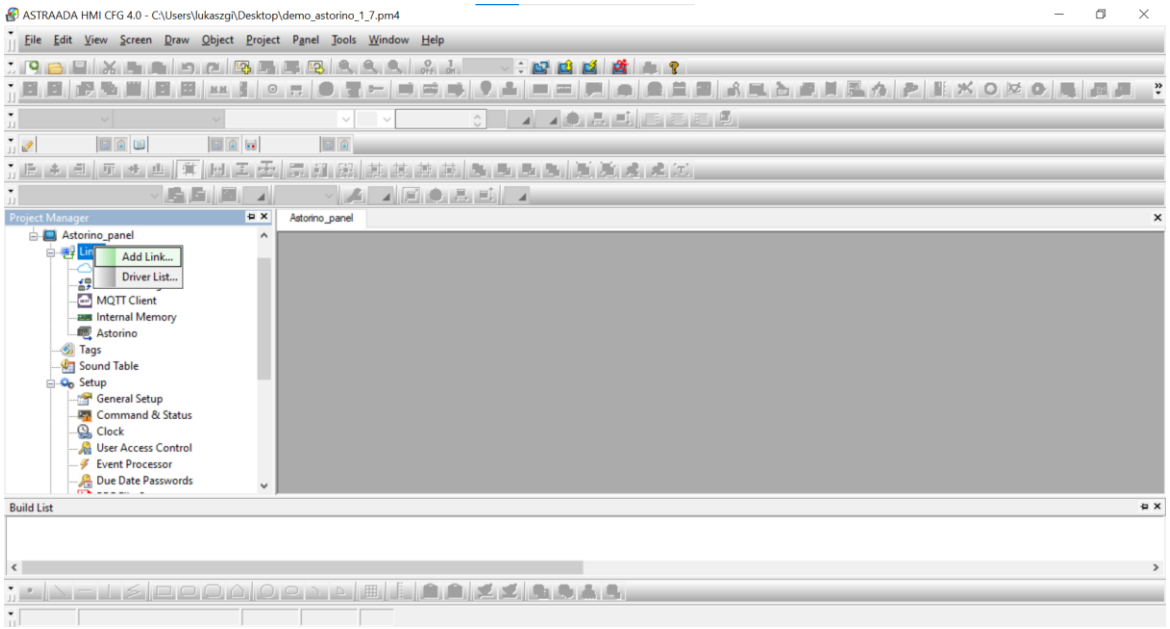


22.4 ASTRAADA HMI panel – example

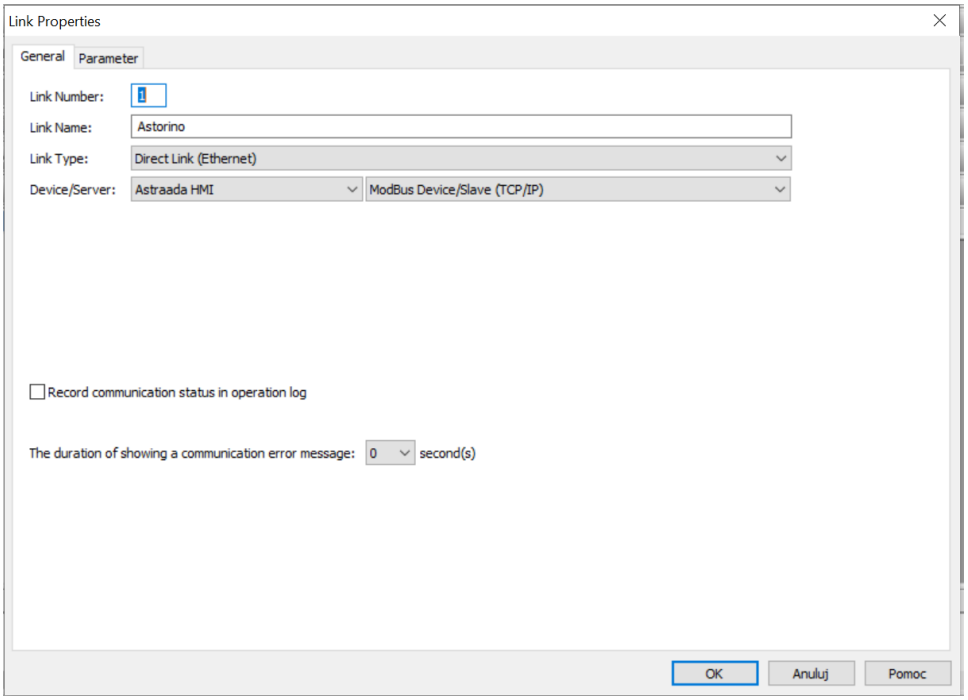
This example shows only the configuration of the Modbus TCP communication protocol on the ASTRAADA HMI panels. For more information refer to the ASTRAADA HMI manuals.

Open ASTRAADA HMI CFG program and set the correct HMI panel in the options.

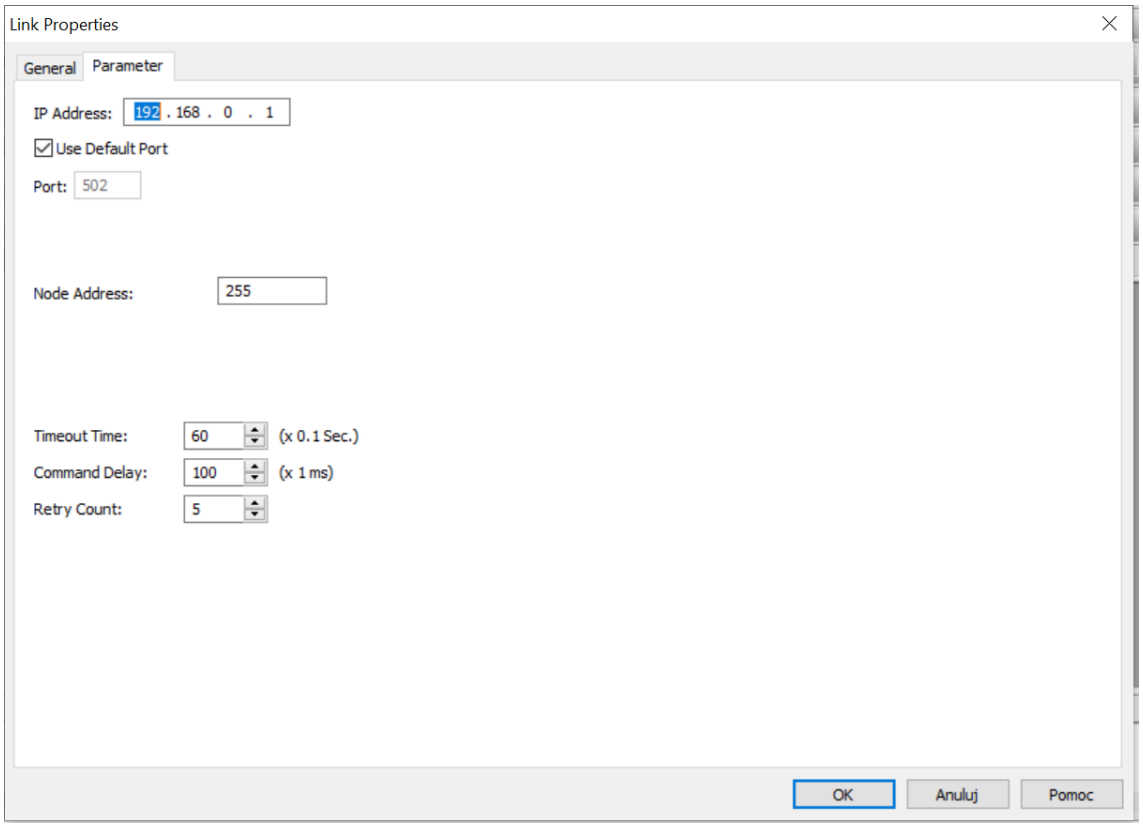
1. Add link.



2. Configure the link add a ModBus (Device/Slave TCP/IP).

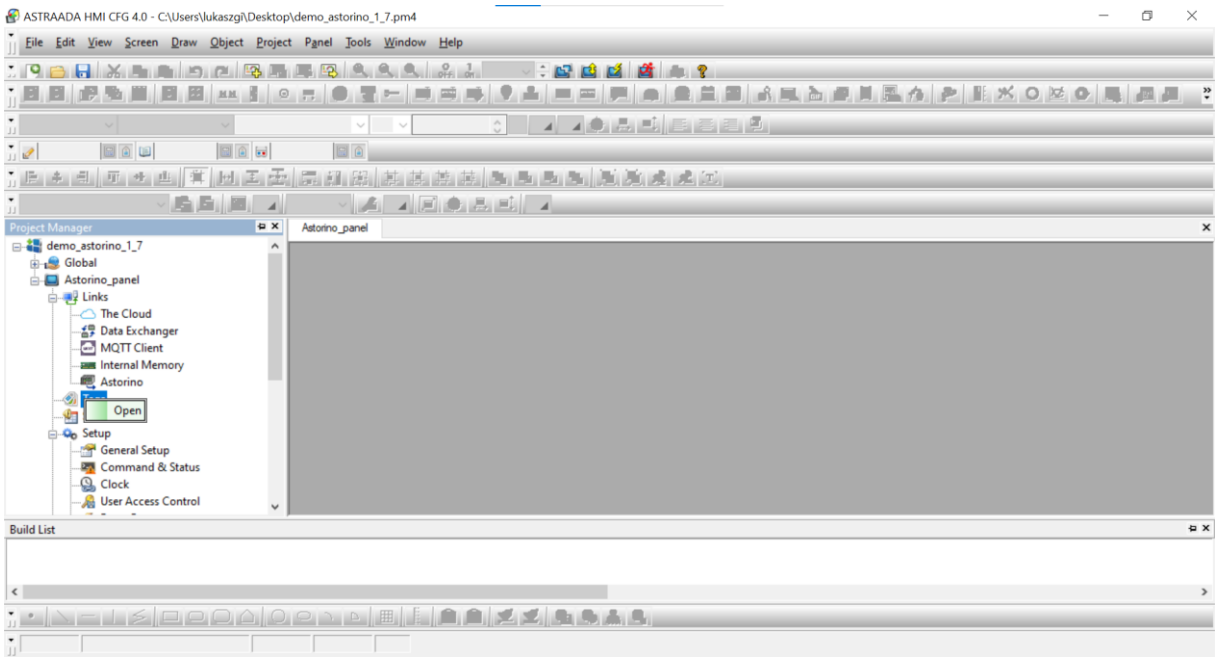


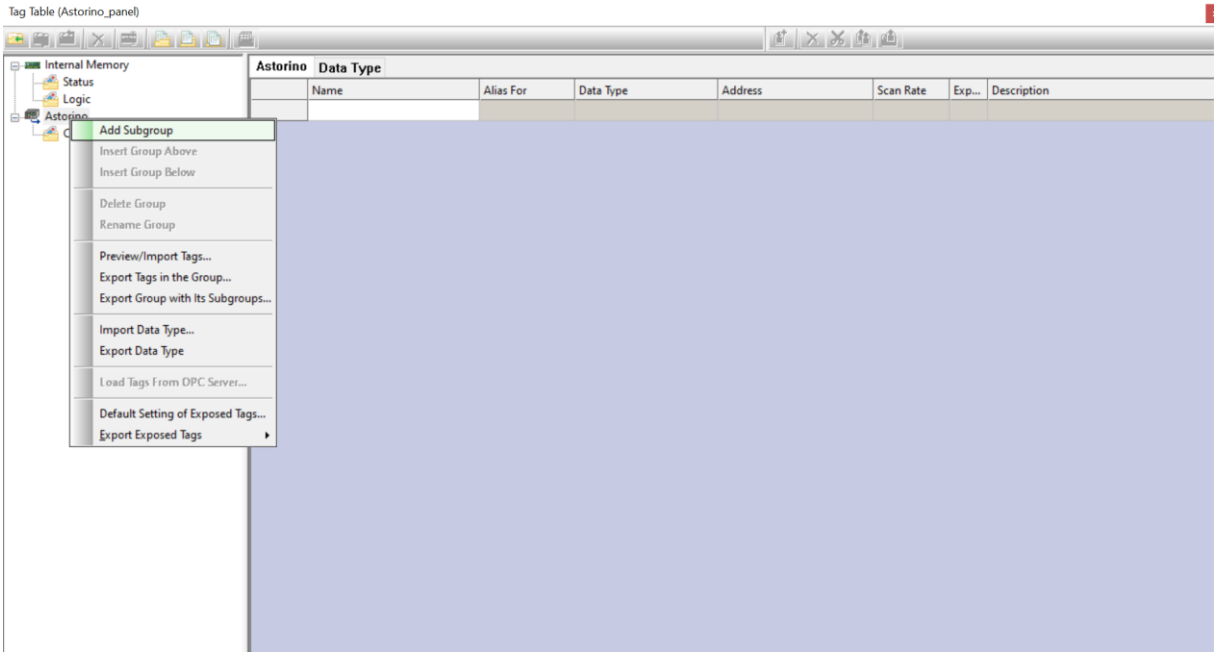
Set IP address, timeout, retry count etc. It is suggested to set retry count to minimum of 3, and timeout to at least 3s.



3. Set data types

In this example Inputs and Outputs will be set as normal and dedicated Inputs/Outputs

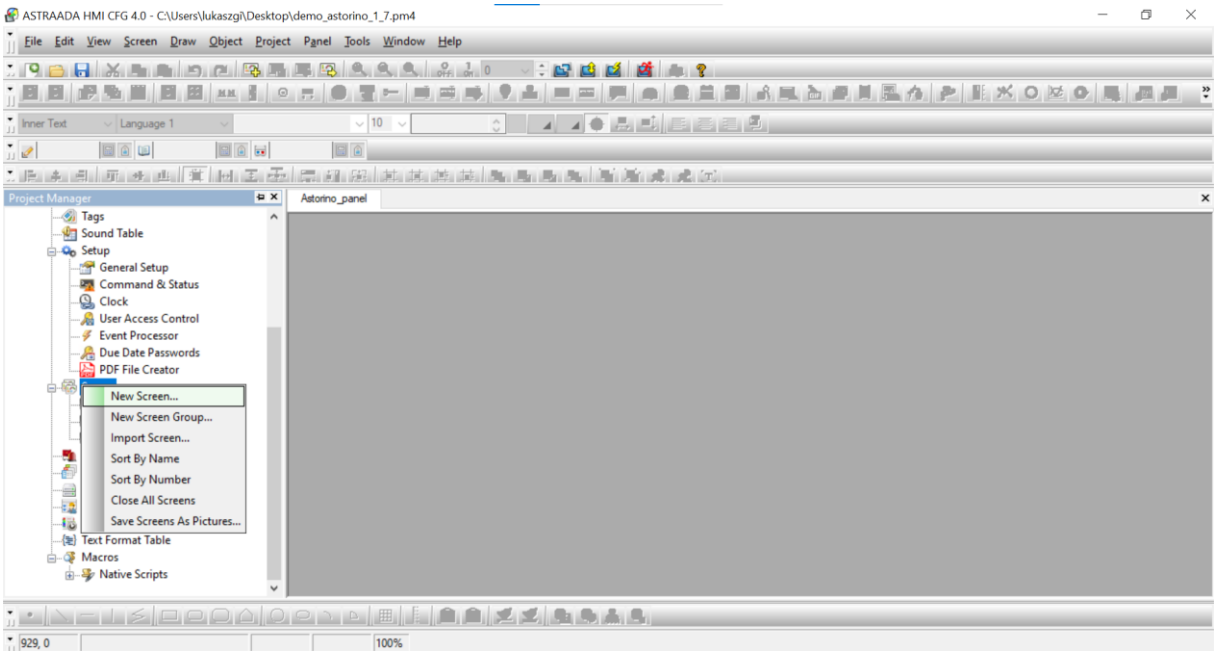


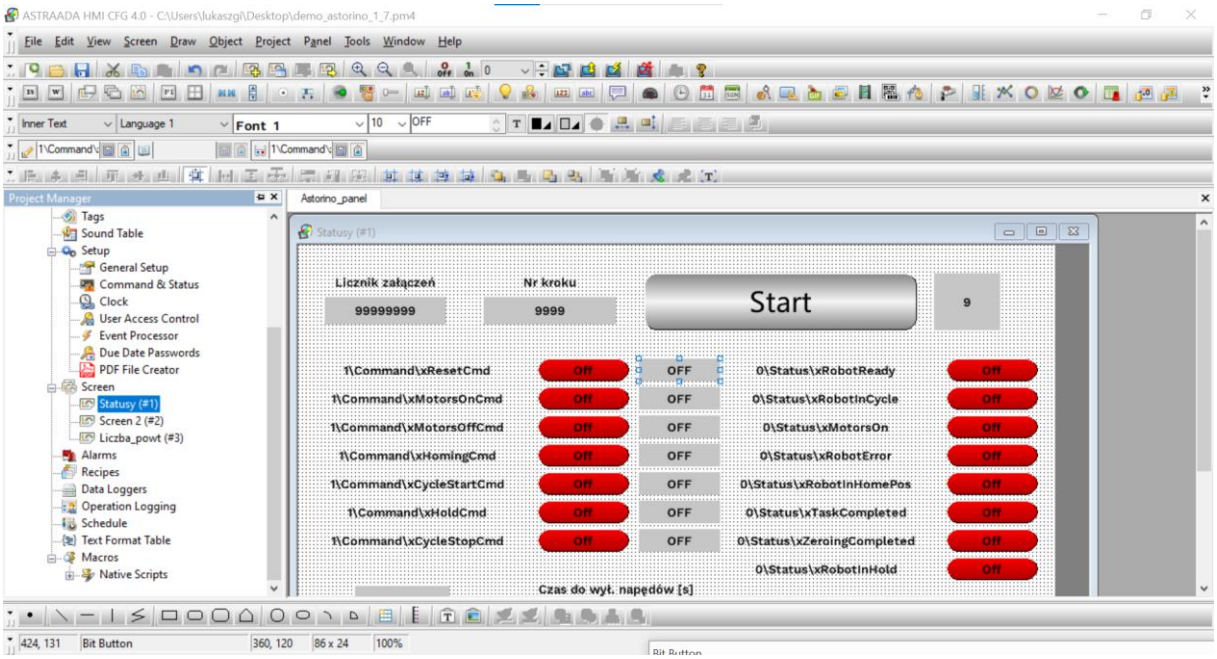


Command Data Type

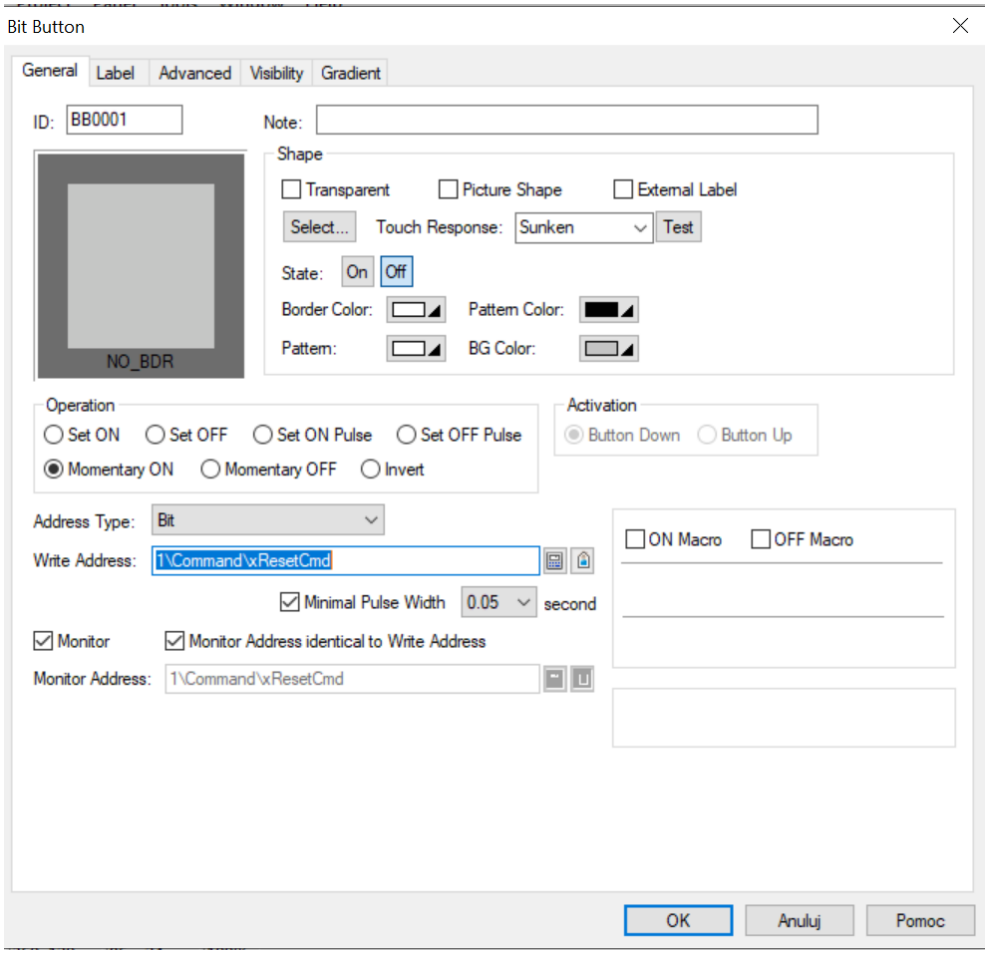
	Name	Alias For	Data Type	Address	Scan Rate	Exp...	Description
1	xResetCmd		Bit	40001.0	Normal	No	Error reset command on the robot
2	xMotorsOnCmd		Bit	40001.1	Normal	No	Motors on command on the robot
3	xMotorsOffCmd		Bit	40001.2	Normal	No	Motors off command on the robot
4	xHomingCmd		Bit	40001.3	Normal	No	Robot position homing command
5	xCycleStartCmd		Bit	40001.4	Normal	No	Start cycle command on the robot
6	xHoldCmd		Bit	40001.5	Normal	No	Command to put the robot into hold mode
7	xCycleStopCmd		Bit	40001.6	Normal	No	Command to stop the robot's work cycle

4. Add a new screen and configure its content.

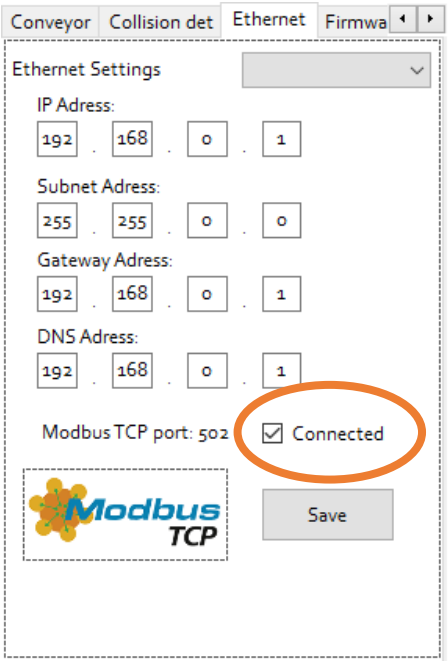




5. Configure the buttons to read/write particular addresses.



6. Save and write data to the HMI.
7. If the communication is working correctly, the status will be displayed on this Menu.



8. Status of the registers is displayed on this tab. If the register Bit is on/true state then the Buttons lights up Yellow.

22.5 Using Modbus registers to read/write numeric data

For reading and writing numeric data through Modbus registers the **BITS** function can be used.

```
.PROGRAM BIT
  BITS 9,16 = 12082 ;sets a number as bits
  x = BITS(1009,16) ;reads a number from bits
.END
```

23 Calibration

Perform calibration after assembling the robot. After calibration, the zeroing data is stored on the microSD card located on the main CPU board inside the robot base. This means that the robot does not have to be recalibrated each time the power supply is switched off.

The calibration procedure is described in the calibration manual.

24 Manufacturer information

For further questions, contact Kawasaki Robotics support.

Contact:

Kawasaki Robotics GmbH

tech-support@kawasakirobot.de

+49 (0) 2131 – 3426 – 1310

Kawasaki Robot
ASTORINO
OPERATION MANUAL

2024-01: 7th Edition

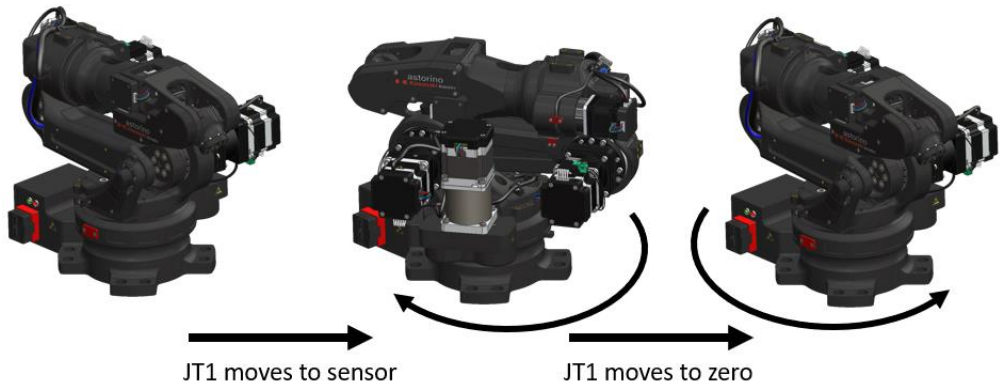
Publication: KAWASAKI Robotics GmbH

Appendix 1 – Default zeroing procedure

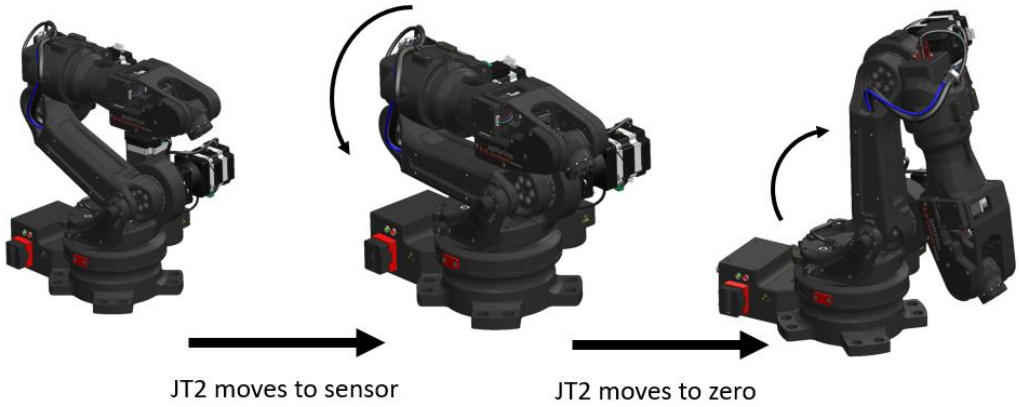
Astorino robot is equiped with incremental encoders, therefore after powering it up all axes must be zeroed.

This procedure is automatic and in its default configuration is decribed below.

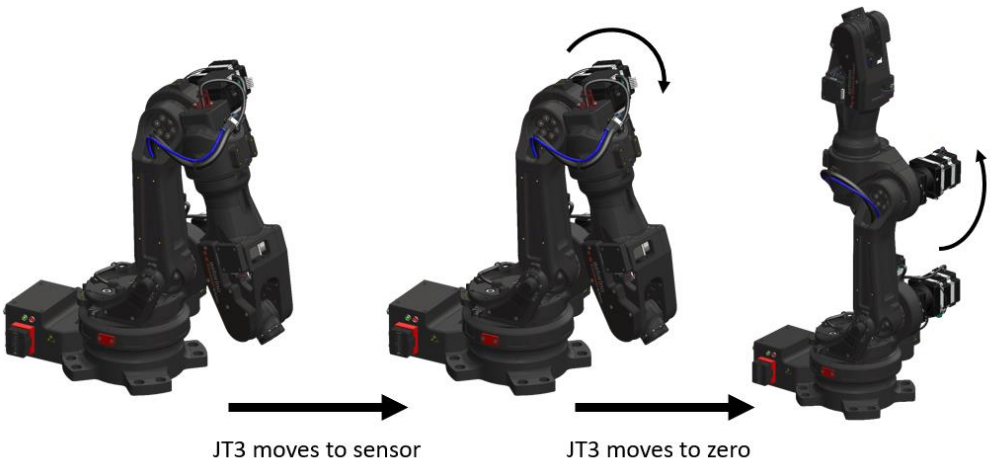
JT1:



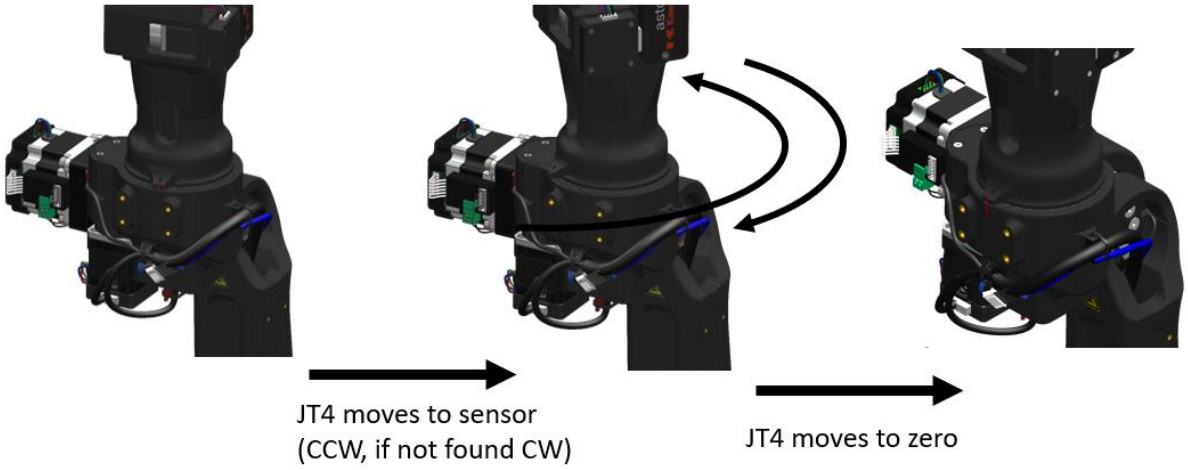
JT2:



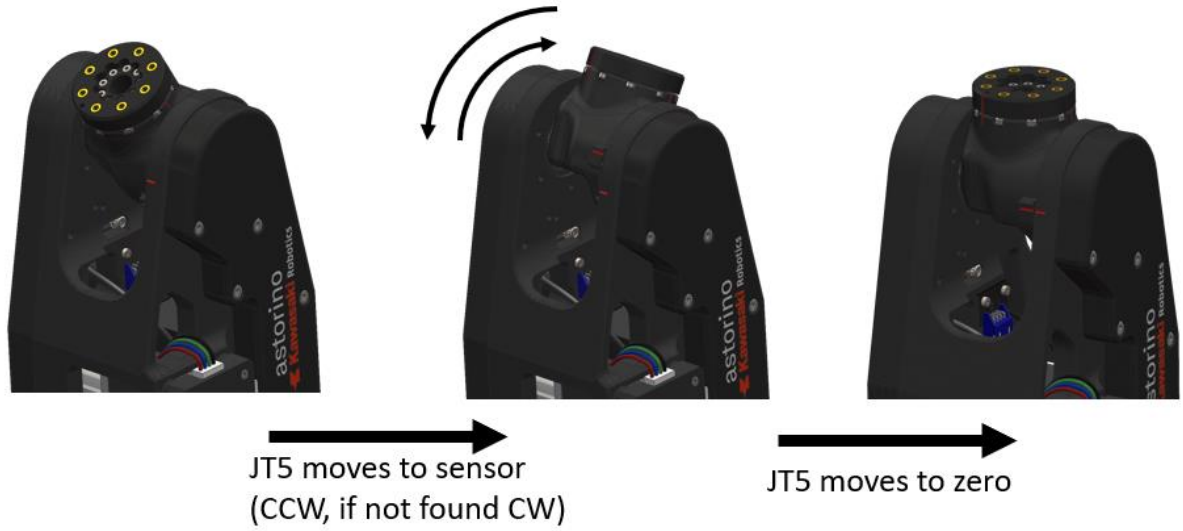
JT3:



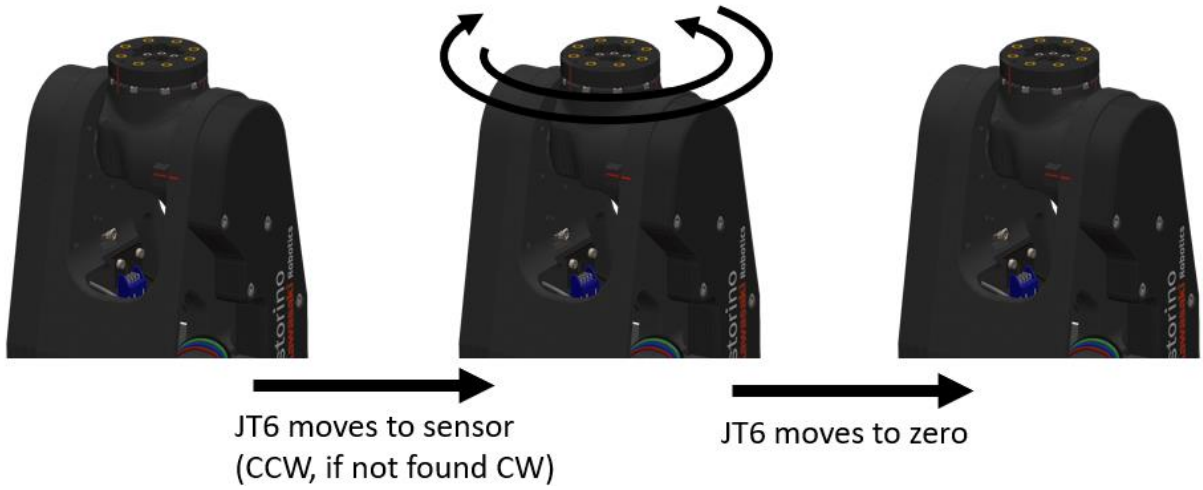
JT4:


JT4 moves to sensor
(CCW, if not found CW)
JT4 moves to zero

JT5:


JT5 moves to sensor
(CCW, if not found CW)
JT5 moves to zero

JT6:


JT6 moves to sensor
(CCW, if not found CW)
JT6 moves to zero

Appendix 2 – PET-G material

PETG is one of the most versatile 3D printing materials, strong and easy to print. Its popularity has increased over the last years as an alternative to PLA. PETG is the PET variant used in 3D printed. The G at the end means Glycol-modified. This change affects the chemical structure, making the material more transparent, less fragile and easier to process.

PETG has very interesting properties, and its closest competitors are PLA and ABS. The main properties you should consider are the following:

- **Rigidity:** Difficulty of the material to be deformed, including stretching and bending. PLA is more rigid than PETG, being PETG and ABS almost as rigid.
- **Resistance:** PETG is generally more difficult to break than PLA and ABS. Based on technical properties, PETG is not only more resistant than ABS, but the adhesion between layers is higher, giving an overall better resistance.
- **Heat resistance:** PETG softens at 80ºC, while PLA can start softening at 50ºC. However, ABS has the highest heat resistance, softening at 105ºC.
- **Odourless printing:** Unlike ABS, PETG does not produce an odour when printed.
- **Recyclable:** Due to its popularity, most cities have the required infrastructure to recycle PETG.

How to print PETG

Hotend temperature: PETG is usually printed at 220-250ºC, and it can be printed with almost any 3D printer, including all-metal hotends or those that use an inner PTFE tube.

- **Surface temperature:** In order to print PETG, it's necessary to use a heated bed at 60-90ºC. It's also recommended to add an adhesive such as paper glue to the print surface.
- **Enclosed 3D printers:** Even though it's not necessary to use enclosed 3D printers, we recommend to avoid room temperature variations.
- **Layer fan:** It's recommended to use a layer fan when printing PETG.
- **Warping:** PETG has a reduced thermal contraction, so it is not prone to warping and results in parts with good dimensional tolerances

Appendix 3 – PNP wiring

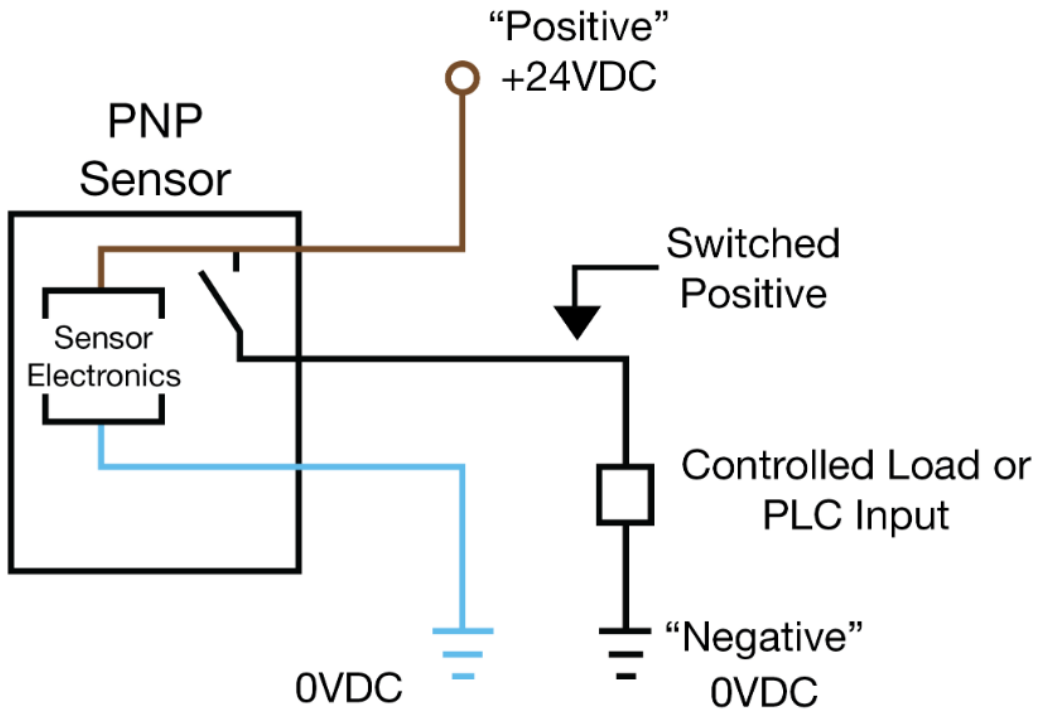
PNP stands for Positive, Negative, Positive. Also known as sourcing. On an IO Module, a PNP input, when undriven is pulled up to a high state e.g. +24V.

Most common in Europe is the 'sinking' type of input/output, these will be used with the PNP sensor or actuator. Less common nowadays are input cards that 'source', these were popular in Asia and require the NPN type of sensor in order to operate correctly.

Here's a simple way remember how to wire up a 3-wire DC PNP:

PNP = Switched Positive

"Switched" refers to which side of the controlled load (relay, small indicator, PLC input) is being switched electrically. Either the load is connected to Negative and the Positive is switched (PNP).



Appendix 4 – Teensy 4.1

Teensy 4.1 is the most powerful Arduino compatible microcontroller available today. Based on the NXP i.MX RT1062 ARM Cortex-M7 running at 600MHz with the ability to be overclocked. It is formatted into a very compact 'teensy' board outline for easy embedding into projects or for use with solderless breadboards. Perhaps best of all, it is compatible with the popular Arduino IDE programming environment as well as many of the existing Arduino libraries, so it is very easy to get up and running unlike many other advanced microcontrollers that are available.

The heart of the i.MX RT1060 microcontroller is an ARM Cortex-M7 CPU core that brings many powerful features to a true real-time microcontroller platform.

The Cortex-M7 is a dual-issue superscalar processor, meaning the M7 can execute two instructions per clock cycle, at 600MHz! Of course, executing two simultaneously depends upon the compiler ordering instructions and registers. Initial benchmarks have shown C++ code compiled by Arduino IDE tends to achieve two instructions per cycle about 40% to 50% of the time while performing numerically intensive work using integers and pointers.

For more information please visit PJRC webpage.

https://www.pjrc.com/store/teensy41.html