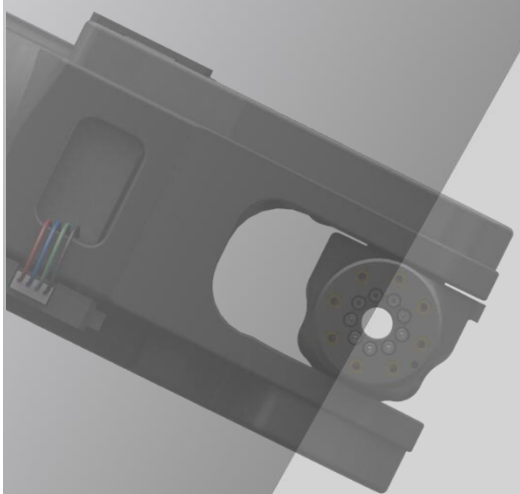
Kawasaki Robotics

ASTORINO

C# API Manual

astorino



Preface

This manual describes the C# API of the 6-axis robot “astorino"

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.

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 ☑.

Change log:

Date	Change Description
2024/09/06	Create a document

Content

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		±0.2 mm
Motion range	Axis 1 (JT1)	±158°
	Axis 2 (JT2)	-90°÷127°
	Axis 3 (JT3)	-168°÷0°
	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, TLL 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
	Conveyor tracking	Max. 2 Encoder

4 Safety instructions

4.1 General information on safety

	Astorino robot does not incorporate breaks on other joints than 2 and 3. During power failure robot might collapse. User safety and vigilance is necessary.

Always ensure the personal safety of users and others when operating the robot arm or starting the robot cell!

• In its basic version, the robot has no safety-related components for the robotic workstation. Such components may be required, depending on the target application. The basic version of the robot is provided with an emergency stop button.

• CE marking: The robot arm, when operating in factory applications, must undergo a risk assessment and comply with applicable safety regulations to ensure personal safety. Depending on the outcome of the assessment, further safety features should be integrated. These typically include safety relays and door switches. The person responsible here is the commissioning engineer. Educational applications do not require additional safety components.

• The robot controller includes a 24 V power supply that must be supplied with mains voltage (100/240 V). Please check the label on the power supply. Only qualified personnel can connect the power supply to the mains and put it into operation.

• Works carried out on the robot's electronic components should only be performed by qualified personnel. Check current guidelines for electrostatic discharges (ESD).

• Always disconnect the robot from the power supply (100/240 V) when working on the robot base (controller) or any electronic components connected to the robot controller.

• Hot-plugging is forbidden! It could lead to a permanent damage to motor modules. Do not install or remove any modules or plug/disconnect connectors (e.g. emergency stop button, DIO modules, motor connectors) while the power is on.

• The robot arm must be placed on a stable surface and bolted or otherwise secured.

• Use and store the robot only in a dry and clean place.

• Use the system only in a room temperature (15° to 32°C) — recommended.

5 C# API

5.1 API introduction

Astorino robot can be controlled by this API with a .NET applications achieving data transmission through certain communication protocols. The communication can be realized by USB-serial port(COM), Ethernet TCP/IP or USB to UART (TLL) converter.

API features:

- Realization of astorino communication protocol
- Long term support by utilizing .NET 6.0 version
- API contains synch and async methods

All motion commands can be execute only in **REPEAT MODE**.

For more information about communication protocol please refer to astorino Communication Protocol Manual.

6 Files description

Compiled version of astorino api contains of several dll files:

- astorino_lib.dll – all communication functions
- System.IO.Ports.dll – COM port library
- System.Managment.dll – COM port library – allows to look for astorino from COM port list.
- ZeroConf.dll – library for mDNS
- System.Reactive.dll – library required by ZeroConf
- System.CodeDom.dll – library required by ZeroConf
- astorino.cs – source code of the library

7 General Information

astorino robot is mainly programmed and operated in astorino software or Teach Pendant, not all functions and operations are available from API. API should be used in combination with astorino software or robot's Teach Pendant.

⚠ WARNING

All motion commands can be executed only in REPEAT MODE! Greater user attention is strongly recommended!

8 Common return values

All methods return one of listed below values:

- Byte ReturnCode;
- Struct retVal:
  - Double[] Values,
  - String Name,
  - List<string> Names,
  - Byte[] bValues,
  - Byte ReturnCode
- String[],
- Struct TransformationPoints:
  - Double[,] points,
  - Byte ReturnCode,
- Struct JointPoints:
  - Double[,] points,
  - Byte ReturnCode

RETURN CODE	DESCRIPTION
0X00	Instruction completed successfully
0X01	CRC error
0X02	Estop or error
0X03	Cycle is ON
0X04	SD save error
0X05	TeachMode – TP deadman switch is OFF
0X06	Cycle is OFF
0X07	Robot is not ready
0X08	Value out range
0X09	AS command failed
0X10	Unknown command ID
0X11	Data frame error
0X12	Motion out of range
0X13	JT command suddenly changed
0X14	Motion out of Working Space
0X15	Robot is already in motion
0X16	Zeroing is not done
0X17	Mastering data missing
0X18	Not allowed in TeachMode
0X19	Zeroing already done
0X20	Response timeout
0X21	Point does not exist
0X22	Wrong data
0X23	Program is not selected
0X24	Motion command exceeded maximum joint speed
0X25	RTC is OFF
0X26	HOLD is active
0X27	Motion disturbed
0XFE	Send instruction failed
0XFF	Communication is not established

In case of structs as returned values only if ReturnCode is 0x00 then other values are correctly returned.

For example retVal ret = ForwardKinematics(…) only returned correct values if ret.ReturnCode = 0x00, then ret.Values[] contains calculation results.

TransformationPoint and JointPoints are described like this [index, {values}] for example

index	val1	val2	val3	val4	val5	val6	val7	exist
0	90	100	-20	45	66	44	0	1
1	0	0	0	0	0	0	0	0
2	10	-10	45	1	12	11	0	1

If exist is equal to 1 then point is saved in robots memory.

8.1 Common methods

Those methods are used to set common parameters of API, search for astorino, connect etc.

8.1.1 Get API Version

Prototype	string getLibVersion()
Description	returns API version
Parameters	void
Return	string

8.1.2 Find COM port

Prototype	List<string> findUSB()
Description	returns List of COM ports associated with astorino robot
Parameters	void
Return	List<string> or null if not found

8.1.3 Find IP address

Prototype	async Task<List<string>> findEthernet()
Description	returns a list of IP addresses associated with astorino robot
Parameters	void
Return	List<string> or null if not found

8.2 Set TimeOut

Prototype	void setTimeOut(int timeOut)
Description	sets communication TimeOut for non-motion instructions (default 1000 ms)
Parameters	timeOut: value in ms
Return	void

8.3 Set Motion TimeOut

Prototype	void setMotionTimeOut(int timeOut)
Description	sets communication TimeOut for motion instructions (default 30000 ms)
Parameters	timeOut: value in ms
Return	void

8.3.1 Connect

Prototype	byte Connect(string path)
Description	starts connection with a robot
Parameters	path: COM port name or IP Adress in string format e.g “COM3”
Return	byte ReturnCode

8.3.2 ConnectViaUART

Prototype	byte ConnectViaUART(string portName)
Description	starts connection with a robot via USB to UART (TTL) converter
Parameters	portName: COM port name “COM3”
Return	byte ReturnCode

8.3.3 Disconnect

Prototype	byte Disconnect()
Description	closes connection with a robot
Parameters	void
Return	byte ReturnCode

8.3.4 Cancel Motion

Prototype	byte cancelMotion()
Description	cancels current motion – robot decelerates and stops – cycle is off afterwards
Parameters	void
Return	byte ReturnCode

8.3.5 Emergency Stop

Prototype	byte emergencyStop()
Description	stops current motion – robot stops immediately – cycle is off afterwards
Parameters	void
Return	byte ReturnCode

8.4 Kinematics and unit conversion methods

8.4.1 Forward Kinematics

Prototype	retVal ForwardKinematics(double jt1, double jt2, double jt3, double jt4, double jt5, double jt6, double jt7)
Description	return forward kinematics calculation result
Parameters	double jt1..jt7 – axis angles in degrees [°]
Return	retVal: - Double[] Values – calculation result (x,y,z,o,a,t,jt7)) in [mm] and [°] - ReturnCode

Prototype	retVal ForwardKinematics(double jt1, double jt2, double jt3, double jt4, double jt5, double jt6)
Description	return forward kinematics calculation result
Parameters	double jt1..jt6– axis angles in degrees [°]
Return	retVal: - Double[] Values – calculation result (x,y,z,o,a,t,jt7)) in [mm] and [°] - ReturnCode

Prototype	retVal ForwardKinematics(double[] jt)
Description	return forward kinematics calculation result
Parameters	double[] jt – axis angles in degrees [°]
Return	retVal: - Double[] Values – calculation result (x,y,z,o,a,t,jt7) in [mm] and [°] - ReturnCode

8.4.2 Inverse Kinematics

Prototype	retVal InverseKinematics(double x, double y, double z, double o, double a, double t, double jt7)
Description	return inverse kinematics calculation result
Parameters	double x,y,z, jt7 in [mm] o,a,t – in degrees [°]
Return	retVal: - Double[] Values – calculation result (jt1, jt2 , jt3, jt4, jt5, jt6, jt7) in degrees [°] - ReturnCode

Prototype	retVal InverseKinematics(double x, double y, double z, double o, double a, double t)
Description	return inverse kinematics calculation result
Parameters	double x,y,z in [mm] o,a,t – in degrees [°]
Return	retVal: - Double[] Values – calculation result (jt1, jt2 , jt3, jt4, jt5, jt6, jt7) in degrees [°] - ReturnCode

Prototype	retVal InverseKinematics(double[] pose)
Description	return inverse kinematics calculation result
Parameters	double[] pose (x,y,z,o,a,t,jt7) or (x,y,z,o,a,t) x,y,z,jt7 in [mm] o,a,t – in degrees [°]
Return	retVal: - Double[] Values – calculation result (jt1, jt2 , jt3, jt4, jt5, jt6, jt7) in degrees [°] - ReturnCode

8.4.3 RPY to OAT conversion

Prototype	retVal toOAT(double rx, double ry, double rz)
Description	return OAT angles calculated from RPY
Parameters	double rx, ry, rz in degrees [°]
Return	retVal: - Double[] Values – calculation result (o, a, t) in degrees [°] - ReturnCode

8.4.4 OAT to RPY conversion

Prototype	retVal toRPY(double o, double a, double t)
Description	return RPY angles calculated from OAT
Parameters	double o, a, t in degrees [°]
Return	retVal: - Double[] Values – calculation result (rx, ry, rz) in degrees [°] - ReturnCode

8.5 Read robot status and parameters methods

8.5.1 Read Status bytes

Prototype	retVal readStatusBytes()
Description	reads all status bytes described in Communication Protocol Manual
Parameters	void
Return	retVal: - byte[] bValues – status bytes [1..5] - ReturnCode

8.5.2 Read motors status

Prototype	retVal isMotorOn()
Description	read the status of motors
Parameters	void
Return	retVal: - int iVal: 1 – MotorsON, -1 - MotorsOff - ReturnCode

8.5.3 Read inHOME bit status

Prototype	retVal isInHome()
Description	read the status of in Home status
Parameters	void
Return	retVal: - int iVal: 1 – in Home, -1 – not in Home - ReturnCode

8.5.4 Read Repeat Mode bit status

Prototype	retVal isRepeatModeOn()
Description	read current selected robot mode (Repeat/Teach mode)
Parameters	void
Return	retVal: - int iVal: 1 – in Repeat Mode, -1 – Teach Mode - ReturnCode

8.5.5 Read HOLD bit status

Prototype	retVal isHoldOn()
Description	read HOLD bit status
Parameters	void
Return	retVal: - int iVal: 1 – in HOLD is ON, -1 – HOLD is OFF - ReturnCode

8.5.6 Read Cycle On bit status

Prototype	retVal isCycleOn()
Description	read Cycle On bit status
Parameters	void
Return	retVal: - int iVal: 1 – Cycle is ON, -1 – Cycle is OFF - ReturnCode

8.5.7 Read E-Stop bit status

Prototype	retVal isEstopOn()
Description	read Emergency stop bit status
Parameters	void
Return	retVal: - int iVal: 1 – E-Stop is ON, -1 – E-stop is OFF - ReturnCode

8.5.8 Read Error bit status

Prototype	retVal isErrorOn()
Description	read Error On bit status
Parameters	void
Return	retVal: - int iVal: 1 – Error is ON, -1 – Error is OFF - ReturnCode

8.5.9 Read Ready bit status

Prototype	retVal isReadyOn()
Description	read Ready bit status
Parameters	void
Return	retVal: - int iVal: 1 – Ready is ON, -1 – Ready is OFF - ReturnCode

8.5.10 Read External Hold On bit status

Prototype	retVal isExternalHoldOn()
Description	read Cycle On bit status
Parameters	void
Return	retVal: - int iVal: 1 – Cycle is ON, -1 – Cycle is OFF - ReturnCode

8.5.11 Read Safety Fence On bit status

Prototype	retVal isSafetyFenceOn()
Description	read Safety Fence On bit status
Parameters	void
Return	retVal: - int iVal: 1 – Safety Fence is ON, -1 – Safety Fence is OFF - ReturnCode

8.5.12 Read Repeat Cont On bit status

Prototype	retVal isRepeatContOn()
Description	read Repeat Cont On bit status
Parameters	void
Return	retVal: - int iVal: 1 – Repeat Cont is ON, -1 – Repeat Cont is OFF (Repeat Once is ON) - ReturnCode

8.5.13 Read Step Once On bit status

Prototype	retVal isStepOnceOn()
Description	read Step Once On bit status
Parameters	void
Return	retVal: - int iVal: 1 – Step Once is ON, -1 – Step Once is OFF (Step Cont is ON) - ReturnCode

8.5.14 Read Step Waiting On bit status

Prototype	retVal isStepWaitingOn()
Description	read Step waiting On bit status
Parameters	void
Return	retVal: - int iVal: 1 – Step is waiting, -1 – Step is not waiting - ReturnCode

8.5.15 Read DryRun status

Prototype	retVal isDryRunOn()
Description	read DryRun status
Parameters	void
Return	retVal: - int iVal: 1 – DryRun is ON, -1 – DryRun is OFF - ReturnCode

8.5.16 Read Zeroing Done bit status

Prototype	retVal isZeroingDone()
Description	read Zeroing done status
Parameters	void
Return	retVal: - int iVal: 1 – Zeroing is done, -1 – Zeroing is done - ReturnCode

8.5.17 Read inMotion bit status

Prototype	retVal isInMotion()
Description	read if robot is currently in motion
Parameters	void
Return	retVal: - int iVal: 1 – in Motion, -1 – robot is not moving - ReturnCode

8.5.18 Read I/O Active Module On bit status

Prototype	retVal isIOActive()
Description	read status of an IO module
Parameters	void
Return	retVal: - int iVal: 1 – IO module is ON, -1 – IO module is off - ReturnCode

8.5.19 Read ModbusConnected bit status

Prototype	retVal isModbusConnected()
Description	read Modbus connected status bit
Parameters	void
Return	retVal: - int iVal: 1 – is connected, -1 – is disconnected - ReturnCode

8.5.20 Read Collision detection module active bit status

Prototype	retVal isCollisionDetectionActive()
Description	read Collision detection module status
Parameters	void
Return	retVal: - int iVal: 1 – Collision detection is ON, -1 – Collison detection is OFF - ReturnCode

8.5.21 Read Zeroing is running bit status

Prototype	retVal isZeroingRunning()
Description	read zeroing is running bit status
Parameters	void
Return	retVal: - int iVal: 1 – zeroing is running, -1 – zeroing is not running - ReturnCode

8.5.22 Read Teach motion active bit status

Prototype	retVal isTeachMotionActive()
Description	read an information if user is moving a robot in Teach mode
Parameters	void
Return	retVal: - int iVal: 1 – Active, -1 – not Active - ReturnCode

8.5.23 Read in motion command bit status

Prototype	retVal isMotionCommand()
Description	read if robot is executing motion command in TeachMode
Parameters	void
Return	retVal: - int iVal: 1 – Active, -1 – not Active - ReturnCode

8.5.24 Read currently selected Teach Motion speed

Prototype	retVal readSelectedTeachSpeed()
Description	reads currently selected Teach motion speed
Parameters	void
Return	retVal: - int iVal: [1..5] - ReturnCode

8.5.25 Read currently selected TOOL number

Prototype	retVal readSelectedToolNumber()
Description	reads currently selected Tool number
Parameters	void
Return	retVal: - int iVal: [1..4] - ReturnCode

8.5.26 Read currently selected Teach motion mode

Prototype	retVal readSelectedTeachMotionMode()
Description	reads currently selected Teach motion mode
Parameters	void
Return	retVal: - int iVal: 1 – TOOL, 2 – JOINT, 3 – CONV, 4 – BASE, 5 - WORK - ReturnCode

8.5.27 Read endstop (Hall sensor) state

Prototype	retVal isHallSensorOn(int jt)
Description	reads current state of a joints zeroing-hall sensor
Parameters	jt – joint number
Return	retVal: - int iVal: 1 – sensor is ON, -1 – sensor is OFF - ReturnCode

8.5.28 Read current joint position

Prototype	retVal JT()
Description	reads current robot position – joint angles in degrees
Parameters	void
Return	retVal: - Double[] Values – (jt1, jt2, jt3, jt4, jt5, jt6, jt7) in degrees [°] - ReturnCode

8.5.29 Read current joint velocity

Prototype	retVal JT_Velocity()
Description	reads current robot speed – joint velocity in degrees/s
Parameters	void
Return	retVal: - Double[] Values – (jt1, jt2, jt3, jt4, jt5, jt6, jt7) in degrees/s [°/s] - ReturnCode

8.5.30 Read current Pose

Prototype	retVal Pose()
Description	reads current robot position – pose
Parameters	void
Return	retVal: - Double[] Values – (x, y, z, o, a, t, jt7) in [mm] and degrees [°] - ReturnCode

8.5.31 Read current TCP speed

Prototype	retVal TCP_Velocity()
Description	reads current robot speed – TCP velocity
Parameters	void
Return	retVal: - Double[] Values – (dx, dy, dz, drx, dry, drz, djt7) in [mm/s] and degrees/s [°/s] - ReturnCode

8.5.32 Read current CPU temperature

Prototype	retVal cpuTemp()
Description	reads current CPU temperature
Parameters	void
Return	retVal: - int iVal – temperature in [°C] - ReturnCode

8.5.33 Read current Monitor speed

Prototype	retVal readMonitorSpeed()
Description	reads current Monitor speed
Parameters	void
Return	retVal: - int iVal – monitor speed [%] - ReturnCode

8.5.34 Read error code

Prototype	retVal readErrorCode()
Description	reads error code
Parameters	void
Return	retVal: - int iVal – error code - ReturnCode

8.5.35 Read serial number

Prototype	retVal serialNumber()
Description	reads robots serial number
Parameters	void
Return	retVal: - string Name – serial number - ReturnCode

8.5.36 Read firmware version

Prototype	retVal firmwareVersion()
Description	reads robots firmware version
Parameters	void
Return	retVal: - string Name – firmware version - ReturnCode

8.5.37 Read all Transformation Points

Prototype	TransformationPoints readTransPointsData()
Description	reads all transformation points data (Pxx)
Parameters	void
Return	TransformationPoints: - Double[,] points – array of positions [index,[x,y,z,o,a,t,jt7, exist]] - ReturnCode

8.5.38 Read one Transformation Point data

Prototype	retVal readTransformationPoint(int index)
Description	reads selected transformation point data (Pxx)
Parameters	void
Return	retVal: - Double[] Values – array [x,y,z,o,a,t,jt7] - ReturnCode

8.5.39 Read all Joint Points

Prototype	JointPoints readJointPointsData()
Description	reads all joint points data (#Pxx)
Parameters	void
Return	JointPoints: - Double[,] points – array of positions [index,[jt1,jt2,jt3,jt4,jt5,jt6,jt7, exist]] - ReturnCode

8.5.40 Read one Joint Point data

Prototype	retVal readJointPoint(int index)
Description	reads selected joint point data (#Pxx)
Parameters	void
Return	retVal: - Double[] Values – array [jt1,jt2,jt3,jt4,jt5,jt6,jt7] - ReturnCode

8.5.41 Read TOOL transformation

Prototype	retVal readToolData(int index)
Description	reads selected TOOL transformation data
Parameters	void
Return	retVal: - Double[] points – array [x,y,z,o,a,t] - ReturnCode

8.5.42 Read all stored programs names

Prototype	retVal readProgramsName()
Description	reads all saved programs names in robots memory
Parameters	void
Return	retVal: - string[,] Names – array of programs names - ReturnCode

8.5.43 Read main (start-up) program name

Prototype	retVal mainProgram()
Description	reads the name of a start-up program
Parameters	void
Return	retVal: - string Name - ReturnCode

8.5.44 Read selected program name

Prototype	retVal selectedProgram()
Description	reads the name of a selected program
Parameters	void
Return	retVal: - string Name - ReturnCode

8.5.45 Read accelerometer data

Prototype	retVal AccelerometerData()
Description	reads accelerometer data (valid only for B-version)
Parameters	void
Return	retVal: - Double[] Values – array [x,y,z] - ReturnCode Accelerometer data contains data of 3 axes. Range is from -2G to 2G.

8.6 Set robot status and parameters methods

8.6.1 Set Motors ON

Prototype	byte setMotorOn()
Description	turns motors on
Parameters	void
Return	byte ReturnCode

8.6.2 Set Motors OFF

Prototype	byte setMotorOn()
Description	turns motors off
Parameters	void
Return	byte ReturnCode

8.6.3 Reset

Prototype	byte reset()
Description	resets error and if possible sets Ready status
Parameters	void
Return	byte ReturnCode

8.6.4 Hold

Prototype	byte Hold()
Description	Holds the robot task execution
Parameters	void
Return	byte ReturnCode

8.6.5 Run (hold off)

Prototype	byte Run()
Description	Turns off Hold switch – resumes the robot task execution
Parameters	void
Return	byte ReturnCode

8.6.6 Turn DryRun On

Prototype	byte turnDryRunOn()
Description	turns on the DryRun mode
Parameters	void
Return	byte ReturnCode

8.6.7 Turn DryRun Off

Prototype	byte turnDryRunOff()
Description	turns off the DryRun mode
Parameters	void
Return	byte ReturnCode

8.6.8 Set Repeat Cont On

Prototype	byte setRepeatCont()
Description	turns on looping of an AS written program
Parameters	void
Return	byte ReturnCode

8.6.9 Set Repeat Cont Off (Repeat Once)

Prototype	byte setRepeatOnce()
Description	turns off looping of an AS written program
Parameters	void
Return	byte ReturnCode

8.6.10 Set Step Once

Prototype	byte setStepOnce()
Description	turns on step mode execution of an AS written program
Parameters	void
Return	byte ReturnCode

8.6.11 Set Step Cont

Prototype	byte setStepCont()
Description	turns off step mode execution of an AS written program
Parameters	void
Return	byte ReturnCode

8.6.12 Next step

Prototype	byte nextStep()
Description	executes a next step of currently running program in step Once mode
Parameters	void
Return	byte ReturnCode

8.6.13 Cycle start

Prototype	byte cycleStart()
Description	turns on the execution of a selected AS written program
Parameters	void
Return	byte ReturnCode

8.6.14 Cycle stop

Prototype	byte cycleStop()
Description	turns off the execution of a selected AS written program – robot decelerates and stops
Parameters	void
Return	byte ReturnCode

8.6.15 Set currently selected TOOL number

Prototype	byte setTool(int index)
Description	selects currently tool number used by robot
Parameters	index – tool number [1, 2, 3]
Return	byte ReturnCode

8.6.16 Set currently selected WORK number

Prototype	byte setWork(int index)
Description	selects currently work number used by robot
Parameters	index – tool number [1, 2]
Return	byte ReturnCode

8.6.17 Set HOME data to current position of a robot

Prototype	byte setHomeHere()
Description	sets HOME position to current position of a robot
Parameters	void
Return	byte ReturnCode

8.6.18 Set monitor speed

Prototype	byte setMonitorSpeed(int speed)
Description	sets monitor speed [1-100%]
Parameters	speed – 1 – 100 [%]
Return	byte ReturnCode

8.6.19 Set HOME position to specific data

Prototype	byte setHome(double jt1, double jt2, double jt3, double jt4, double jt5, double jt6, double jt7)
Description	sets robots HOME position to a specific data
Parameters	double jt1..jt7 – in degrees [°]
Return	byte ReturnCode

Prototype	byte setHome(double jt1, double jt2, double jt3, double jt4, double jt5, double jt6)
Description	sets robots HOME position to a specific data
Parameters	double jt1..jt6 – in degrees [°]
Return	byte ReturnCode

Prototype	byte setHome(double[] jt)
Description	sets robots HOME position to a specific data
Parameters	double[] jt – in degrees [°]
Return	byte ReturnCode

8.6.20 Save current position to a Joint point

Prototype	byte saveCurrentJointsAsPoint(int point)
Description	saves current position (joint angles) of a robot to a selected point (#Px)
Parameters	point – point number from #P list [0-99]
Return	byte ReturnCode

8.6.21 Save current position to a Transformation point

Prototype	byte saveCurrentPoseAsPoint(int point)
Description	saves current position (pose) of a robot to a selected point (Px)
Parameters	point – point number from P list [0-99]
Return	byte ReturnCode

8.6.22 Set Main program (Start-up program)

Prototype	byte setMainProgram (string name)
Description	sets selected program as a start-up program
Parameters	name – program name
Return	byte ReturnCode

8.6.23 Select program (PRIME)

Prototype	byte selectProgram (string name)
Description	selects a program for execution
Parameters	name – program name
Return	byte ReturnCode

8.6.24 Remove point

Prototype	byte removePoint (int number, int type)
Description	saves current position (pose) of a robot to a selected point (Px)
Parameters	numer – point number [0-99], type – point type: 1 – Transformation (Px), 2 – Joint (#Px)
Return	byte ReturnCode

8.6.25 Set TOOL data

Prototype	byte mainProgram(int index, double[] data)
Description	sets TOOL data for a specific TOOL number
Parameters	int index – [1..3] double[] – {x,y,z,o,a,t}
Return	byte ReturnCode

8.7 I/O methods

8.7.1 Read input state

Prototype	retVal readInput(int ind)
Description	reads selected input state
Parameters	ind – IO index [1..58]
Return	retVal: - int iVal: 1 – Input is ON, -1 – Input is OFF - ReturnCode

8.7.2 Read output state

Prototype	retVal readOutput(int ind)
Description	reads selected output state
Parameters	ind – IO index [1..58]
Return	retVal: - int iVal: 1 – Output is ON, -1 – Output is OFF - ReturnCode

8.7.3 Read internal signal state

Prototype	retVal readInternal(int ind)
Description	reads selected internal signal state
Parameters	ind – internal signal index [1..16]
Return	retVal: - int iVal: 1 – Internal signal is ON, -1 – Internal signal is OFF - ReturnCode

8.7.4 Set output state

Prototype	byte setOutput(int ind, int state)
Description	sets selected output state
Parameters	ind – IO index [1..58] state – signal state: 1 – ON, -1 - OFF
Return	byte ReturnCode

8.7.5 Set internal signal state

Prototype	byte setInternal(int ind, int state)
Description	sets selected internal signal state
Parameters	ind – internal signal index [1..16] state – signal state: 1 – ON, -1 - OFF
Return	byte ReturnCode

8.8 Synchronious motion methods (blocking)

8.8.1 Start Zeroing

Prototype	byte Zero()
Description	starts Zeroing procedure
Parameters	void
Return	byte ReturnCode

8.8.2 HOME

Prototype	byte HOME(byte spd, byte acc, byte dec)
Description	starts motion to HOME position in JOINT interpolation
Parameters	spd: [1..100%] acc: [0..100%] – if acc = 0 then initial speed of motion is set to spd dec: [0..100%] – if dec = 0 then final speed of motion is set to spd
Return	byte ReturnCode

8.8.3 JMOVE to saved position

Prototype	byte JMOVE(byte pointType, byte pointIndex, byte spd, byte acc, byte dec)
Description	starts motion to selected point in JOINT interpolation
Parameters	pointType: 1 – Transformation point, 2 – Joint point pointIndex: [0..99] point index spd: [1..100%] acc: [0..100%] dec: [0..100%]
Return	byte ReturnCode

8.8.4 JMOVE to point data

WARNING

This command might cause rapid movement if sent data is incorrect! Greater user attention is strongly recommended!

Prototype	byte JMOVE(byte pointType, byte spd, byte acc, byte dec, double[] target)
Description	starts motion to target as a data array in JOINT interpolation
Parameters	pointType: 1 – Transformation point, 2 – Joint point spd: [1..100%] acc: [0..100%] dec: [0..100%] target: [val1, val2, val3, val4, val5, val6] or [val1, val2, val3, val4, val5, val6, val7]
Return	byte ReturnCode

8.8.5 JAPPRO to saved position

Prototype	byte JAPPRO(byte pointType, byte pointIndex, byte spd, byte acc, byte dec, double offset)
Description	starts approach motion to selected point in JOINT interpolation. Offset sets the distance from a destination point in [mm] in TOOL coordinate system according to Z direction
Parameters	pointType: 1 – Transformation point, 2 – Joint point pointIndex: [0..99] point index spd: [1..100%] acc: [0..100%] dec: [0..100%] offset: [mm]
Return	byte ReturnCode

8.8.6 JAPPRO to point data

WARNING

This command might cause rapid movement if sent data is incorrect! Greater user attention is strongly recommended!

Prototype	byte JAPPRO(byte pointType, byte spd, byte acc, byte dec, double[] target, double offset)
Description	starts approach motion to target as a data array in JOINT interpolation. Offset sets the distance from a destination point in [mm] in TOOL coordinate system according to Z direction
Parameters	pointType: 1 – Transformation point, 2 – Joint point spd: [1..100%] acc: [0..100%] dec: [0..100%] target: [val1, val2, val3, val4, val5, val6] or [val1, val2, val3, val4, val5, val6, val7] offset: [mm]
Return	byte ReturnCode

8.8.7 LMOVE to saved position

Prototype	byte LMOVE(byte pointType, byte pointIndex, byte spd, byte acc, byte dec)
Description	starts motion to selected point in LINEAR interpolation
Parameters	pointType: 1 – Transformation point, 2 – Joint point pointIndex: [0..99] point index spd: [1..250 mm/s] acc: [0..100%] – if acc = 0 then initial speed of motion is set to spd dec: [0..100%] – if dec = 0 then final speed of motion is set to spd
Return	byte ReturnCode

8.8.8 LMOVE to point data

WARNING

This command might cause rapid movement if sent data is incorrect! Greater user attention is strongly recommended!

Prototype	byte LMOVE(byte pointType, byte spd, byte acc, byte dec, double[] target)
Description	starts motion to target as a data array in LINEAR interpolation
Parameters	pointType: 1 – Transformation point, 2 – Joint point spd: [1..250 mm/s] acc: [0..100%] – if acc = 0 then initial speed of motion is set to spd dec: [0..100%] – if dec = 0 then final speed of motion is set to spd target: [val1, val2, val3, val4, val5, val6] or [val1, val2, val3, val4, val5, val6, val7]
Return	byte ReturnCode

8.8.9 LAPPRO to saved position

Prototype	byte LAPPRO(byte pointType, byte pointIndex, byte spd, byte acc, byte dec, double offset)
Description	starts approach motion to selected point in LINEAR interpolation. Offset sets the distance from a destination point in [mm] in TOOL coordinate system according to Z direction
Parameters	pointType: 1 – Transformation point, 2 – Joint point pointIndex: [0..99] point index spd: [1..250 mm/s] acc: [0..100%] – if acc = 0 then initial speed of motion is set to spd dec: [0..100%] – if dec = 0 then final speed of motion is set to spd offset: [mm]
Return	byte ReturnCode

8.8.10 LAPPRO to point data

WARNING

This command might cause rapid movement if send data is incorrect! Greater user attention is strongly recommended!

Prototype	byte LAPPRO(byte pointType, byte spd, byte acc, byte dec, double[] target, double offset)
Description	starts approach motion to target as a data array in LINEAR interpolation. Offset sets the distance from a destination point in [mm] in TOOL coordinate system according to Z direction
Parameters	pointType: 1 – Transformation point, 2 – Joint point spd: [1..250 mm/s] acc: [0..100%] – if acc = 0 then initial speed of motion is set to spd dec: [0..100%] – if dec = 0 then final speed of motion is set to spd target: [val1, val2, val3, val4, val5, val6] or [val1, val2, val3, val4, val5, val6, val7] offset: [mm]
Return	byte ReturnCode

8.8.11 CMOVE to saved position

Prototype	byte CMOVE(byte pointType, byte pointIndex1, byte pointIndex2, byte spd, byte acc, byte dec)
Description	starts motion to selected points in CIRCULAR interpolation, motion start in current position, via pointIndex1 and ends in pointIndex2
Parameters	pointType: 1 – Transformation point, 2 – Joint point pointIndex1: [0..99] point index pointIndex2: [0..99] point index spd: [1..250 mm/s] acc: [1..100%] dec: [1..100%]
Return	byte ReturnCode

8.8.12 CMOVE to point data

WARNING

This command might cause rapid movement if sent data is incorrect! Greater user attention is strongly recommended!

Prototype	byte CMOVE(byte pointType, byte spd, byte acc, byte dec, double[] middle, double[] target)
Description	starts approach motion to target as a data array, via middle in CIRCULAR interpolation.
Parameters	pointType: 1 – Transformation point, 2 – Joint point spd: [1..250 mm/s] acc: [1..100%] dec: [1..100%] middle: [val1, val2, val3, val4, val5, val6] or [val1, val2, val3, val4, val5, val6, val7] target: [val1, val2, val3, val4, val5, val6] or [val1, val2, val3, val4, val5, val6, val7]
Return	byte ReturnCode

8.9 Asynchronious motion methods (no blocking)

8.9.1 Start Zeroing

Prototype	async Task<byte> ZeroAsync()
Description	starts Zeroing procedure
Parameters	void
Return	byte ReturnCode

8.9.2 HOME

Prototype	async Task<byte> HOMEAsync(byte spd, byte acc, byte dec)
Description	starts motion to HOME position in JOINT interpolation
Parameters	spd: [1..100%] acc: [0..100%] – if acc = 0 then initial speed of motion is set to spd dec: [0..100%] – if dec = 0 then final speed of motion is set to spd
Return	byte ReturnCode

8.9.3 JMOVE to saved position

Prototype	async Task<byte> JMOVEAsync(byte pointType, byte pointIndex, byte spd, byte acc, byte dec)
Description	starts motion to selected point in JOINT interpolation
Parameters	pointType: 1 – Transformation point, 2 – Joint point pointIndex: [0..99] point index spd: [1..100%] acc: [0..100%] dec: [0..100%]
Return	byte ReturnCode

8.9.4 JMOVE to point data

WARNING

This command might cause rapid movement if sent data is incorrect! Greater user attention is strongly recommended!

Prototype	async Task<byte> JMOVEAsync(byte pointType, byte spd, byte acc, byte dec, double[] target)
Description	starts motion to target as a data array in JOINT interpolation
Parameters	pointType: 1 – Transformation point, 2 – Joint point spd: [1..100%] acc: [0..100%] dec: [0..100%] target: [val1, val2, val3, val4, val5, val6] or [val1, val2, val3, val4, val5, val6, val7]
Return	byte ReturnCode

8.9.5 JAPPRO to saved position

Prototype	async Task<byte> JAPPROAsync(byte pointType, byte pointIndex, byte spd, byte acc, byte dec, double offset)
Description	starts approach motion to selected point in JOINT interpolation. Offset sets the distance from a destination point in [mm] in TOOL coordinate system according to Z direction
Parameters	pointType: 1 – Transformation point, 2 – Joint point pointIndex: [0..99] point index spd: [1..100%] acc: [0..100%] dec: [0..100%] offset: [mm]
Return	byte ReturnCode

8.9.6 JAPPRO to point data

WARNING

This command might cause rapid movement if sent data is incorrect! Greater user attention is strongly recommended!

Prototype	async Task<byte> JAPPROAsync(byte pointType, byte spd, byte acc, byte dec, double[] target, double offset)
Description	starts approach motion to target as a data array in JOINT interpolation. Offset sets the distance from a destination point in [mm] in TOOL coordinate system according to Z direction
Parameters	pointType: 1 – Transformation point, 2 – Joint point spd: [1..100%] acc: [0..100%] dec: [0..100%] target: [val1, val2, val3, val4, val5, val6] or [val1, val2, val3, val4, val5, val6, val7] offset: [mm]
Return	byte ReturnCode

8.9.7 LMOVE to saved position

Prototype	async Task<byte> LMOVEAsync(byte pointType, byte pointIndex, byte spd, byte acc, byte dec)
Description	starts motion to selected point in LINEAR interpolation
Parameters	pointType: 1 – Transformation point, 2 – Joint point pointIndex: [0..99] point index spd: [1..250 mm/s] acc: [0..100%] – if acc = 0 then initial speed of motion is set to spd dec: [0..100%] – if dec = 0 then final speed of motion is set to spd
Return	byte ReturnCode

8.9.8 LMOVE to point data

WARNING

This command might cause rapid movement if sent data is incorrect! Greater user attention is strongly recommended!

Prototype	async Task<byte> LMOVEAsync(byte pointType, byte spd, byte acc, byte dec, double[] target)
Description	starts motion to target as a data array in LINEAR interpolation
Parameters	pointType: 1 – Transformation point, 2 – Joint point spd: [1..250 mm/s] acc: [0..100%] – if acc = 0 then initial speed of motion is set to spd dec: [0..100%] – if dec = 0 then final speed of motion is set to spd target: [val1, val2, val3, val4, val5, val6] or [val1, val2, val3, val4, val5, val6, val7]
Return	byte ReturnCode

8.9.9 LAPPRO to saved position

Prototype	async Task<byte> LAPPROAsync(byte pointType, byte pointIndex, byte spd, byte acc, byte dec, double offset)
Description	starts approach motion to selected point in LINEAR interpolation. Offset sets the distance from a destination point in [mm] in TOOL coordinate system according to Z direction
Parameters	pointType: 1 – Transformation point, 2 – Joint point pointIndex: [0..99] point index spd: [1..250 mm/s] acc: [0..100%] – if acc = 0 then initial speed of motion is set to spd dec: [0..100%] – if dec = 0 then final speed of motion is set to spd offset: [mm]
Return	byte ReturnCode

8.9.10 LAPPRO to point data

⚠ WARNING

This command might cause rapid movement if sent data is incorrect! Greater user attention is strongly recommended!

Prototype	async Task<byte> LAPPROAsync(byte pointType, byte spd, byte acc, byte dec, double[] target, double offset)
Description	starts approach motion to target as a data array in LINEAR interpolation. Offset sets the distance from a destination point in [mm] in TOOL coordinate system according to Z direction
Parameters	pointType: 1 – Transformation point, 2 – Joint point spd: [1..250 mm/s] acc: [0..100%] – if acc = 0 then initial speed of motion is set to spd dec: [0..100%] – if dec = 0 then final speed of motion is set to spd target: [val1, val2, val3, val4, val5, val6] or [val1, val2, val3, val4, val5, val6, val7] offset: [mm]
Return	ReturnCode

8.9.11 CMOVE to saved position

Prototype	async Task<byte> CMOVEAsync(byte pointType, byte pointIndex1, byte pointIndex2, byte spd, byte acc, byte dec)
Description	starts motion to selected points in CIRCULAR interpolation, motion start in current position, via pointIndex1 and ends in pointIndex2
Parameters	pointType: 1 – Transformation point, 2 – Joint point pointIndex1: [0..99] point index pointIndex2: [0..99] point index spd: [1..250 mm/s] acc: [1..100%] dec: [1..100%]
Return	ReturnCode

8.9.12 CMOVE to point data

WARNING

This command might cause rapid movement if sent data is incorrect! Greater user attention is strongly recommended!

Prototype	async Task<byte> CMOVEAsync(byte pointType, byte spd, byte acc, byte dec, double[] middle, double[] target)
Description	starts approach motion to target as a data array, via middle in CIRCULAR interpolation.
Parameters	pointType: 1 – Transformation point, 2 – Joint point spd: [1..250 mm/s] acc: [1..100%] dec: [1..100%] middle: [val1, val2, val3, val4, val5, val6] or [val1, val2, val3, val4, val5, val6, val7] target: [val1, val2, val3, val4, val5, val6] or [val1, val2, val3, val4, val5, val6, val7]
Return	ReturnCode

8.10 RTC commands

RTC (Real Time Control) commands can control astorino in real time. User can inject motion offsets to cartesian motions or control the robot via external trajectory generator. Robot must be in REPEAT Mode. RTC is set to OFF automatically after disconnect or switching to Teach Mode.

WARNING

Those commands might cause rapid movement if sent data is incorrect! Greater user attention is strongly recommended!

8.11 RTC commands

RTC (Real Time Control) commands can control astorino in real time. User can inject motion offsets to cartesian motions or control the robot via external trajectory generator. Robot must be in REPEAT Mode. RTC is set to OFF automatically after disconnect or switching to Teach Mode.

WARNING

This commands might cause rapid movement if send data is incorrect! Greater user attention is strongly recommended!

8.12 Turn on RTC

Prototype	byte RTC_ON()
Description	Turns on Real Time Control
Parameters	void
Return	ReturnCode

8.13 Turn off RTC

Prototype	byte RTC_OFF()
Description	Turns off Real Time Control
Parameters	Void
Return	ReturnCode

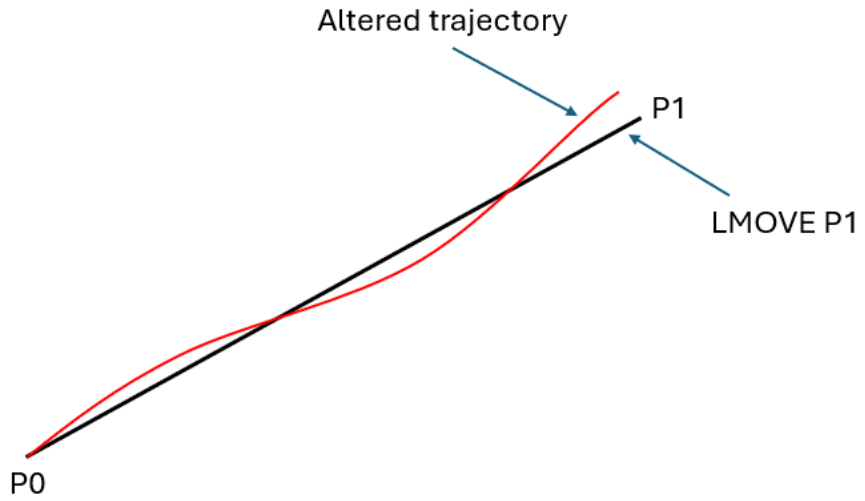
8.14 Set RTC offsets

To use this command RTC switch must be ON, Values are in [mm] and [deg].

WARNING

This command might cause rapid movement if send data is incorrect! Greater user attention is strongly recommended!

RTC offset are added to currently used Linear and Circular Motion commands every 10 ms. That allows to alter current trajectory in real time. Maximum values are 2mm and 2 deg. Offset are cumulative, if not set to 0 will add up with every loop. Cumulative offsets are reset after each motion instruction has ended.



Offsets data are in cartesian units [x,y,z] and Euler RPY angles [rx, ry, rz]

ASTORINO C# API

Prototype	byte setRTC_OffsetData(double x, double y, double z, double rx, double ry, double rz)
Description	Sets RTC offset for path modulation
Parameters	x,y,z in [mm] rx,ry,rz – in degrees [°]
Return	ReturnCode

Prototype	byte setRTC_OffsetData(double[] data)
Description	Sets RTC offset for path modulation
Parameters	x,y,z in [mm] rx,ry,rz – in degrees [°]
Return	ReturnCode

8.15 Execute RTC data motion

To use this command RTC switch must be ON. Values are in absolute coordinates [xzyoat jt7 or joints angle deg 1..7]

⚠ WARNING

This command might cause rapid movement if send data is incorrect! Greater user attention is strongly recommended!

Prototype	byte RTC_move(byte type, byte time, double z, double[] target)
Description	Executes motion to transmitted data. Motion is done in required time.
Parameters	Type: 0x01- JT angles, 0x02- Transformation data Time: unit [ms] target: [x, y, z, o, a, t] – units [mm] and degrees [°]
Return	ReturnCode

⚠ WARNING

This command do not implement acceleration and deceleration. It is up to user commands to realize that!

8.16 AS commands

8.16.1 Execute AS command

Prototype	byte executeASCommand (string command)
Description	Executes AS Language command
Parameters	Command – AS language command
Return	ReturnCode

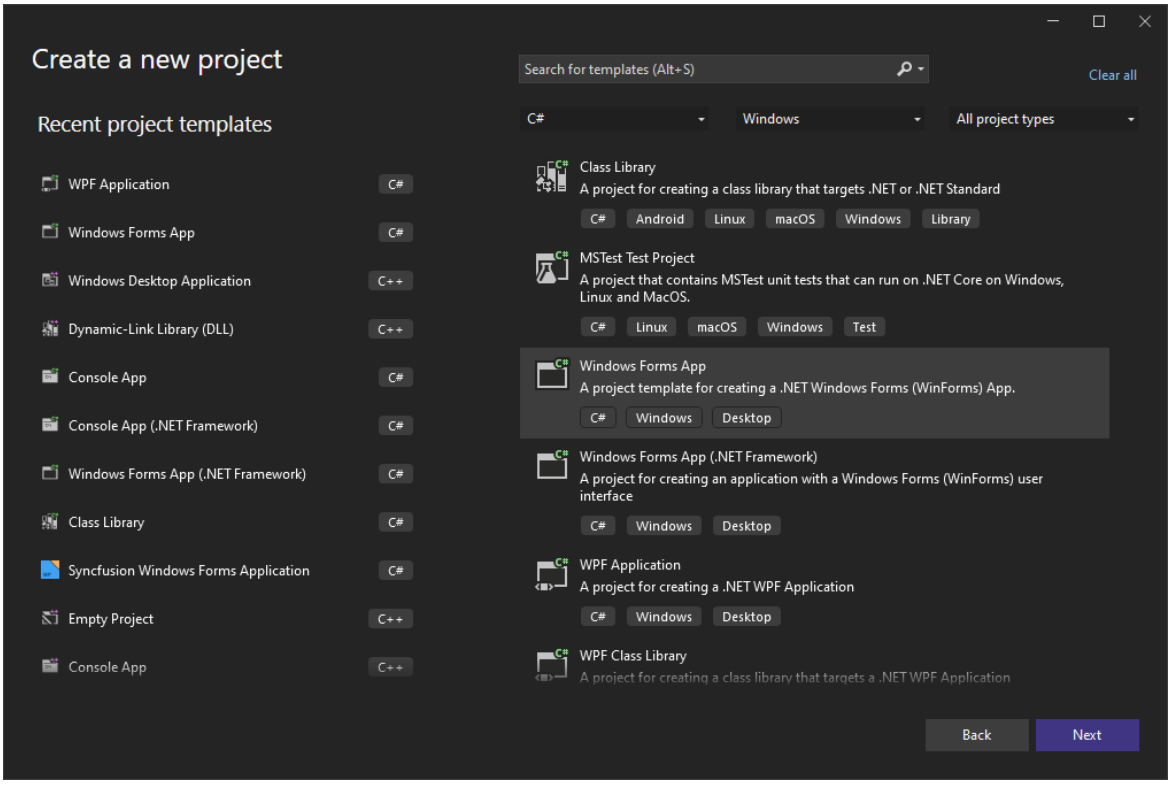
8.16.2 Execute AS command Asynchronous

Prototype	async Task<byte> executeASCommandAsync (string command)
Description	Executes AS Language command
Parameters	Command – AS language command
Return	ReturnCode

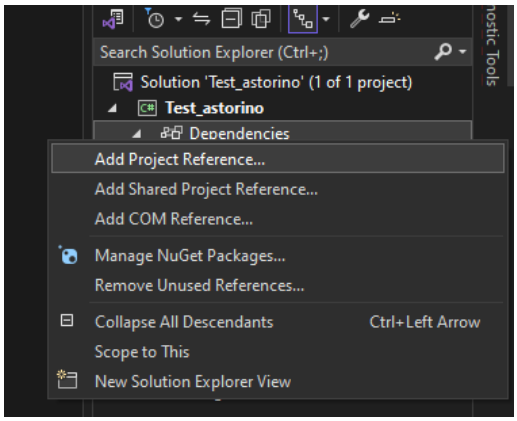
9 Example

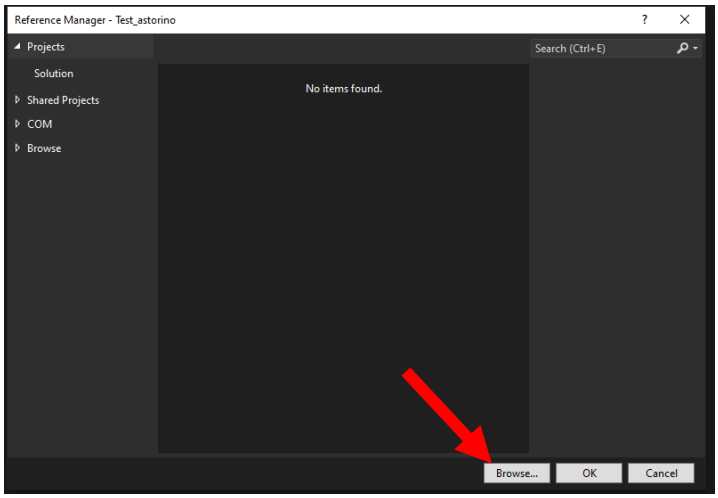
9.1 Visual Studio – WinForms example

Create a new WinForms project based on .NET 6.0



Add Project Reference (right click on project Dependencies)

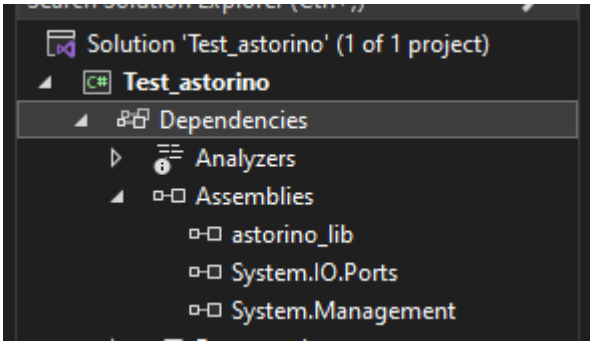




ASTORINO C# API

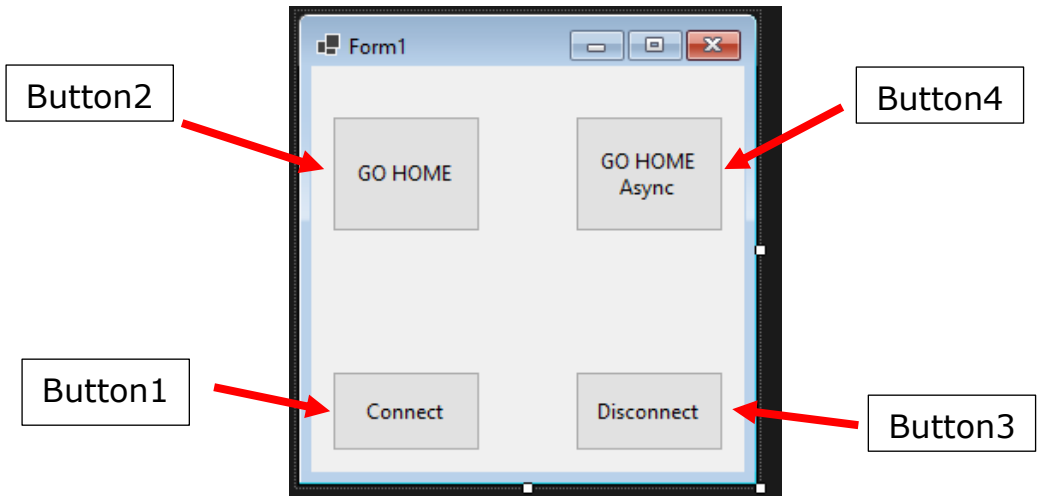
Select:

- astorino_lib.dll
- System.IO.Ports.dll
- System.Managment.dll



It should be loaded to current project:

Now create your GUI:



Example code:

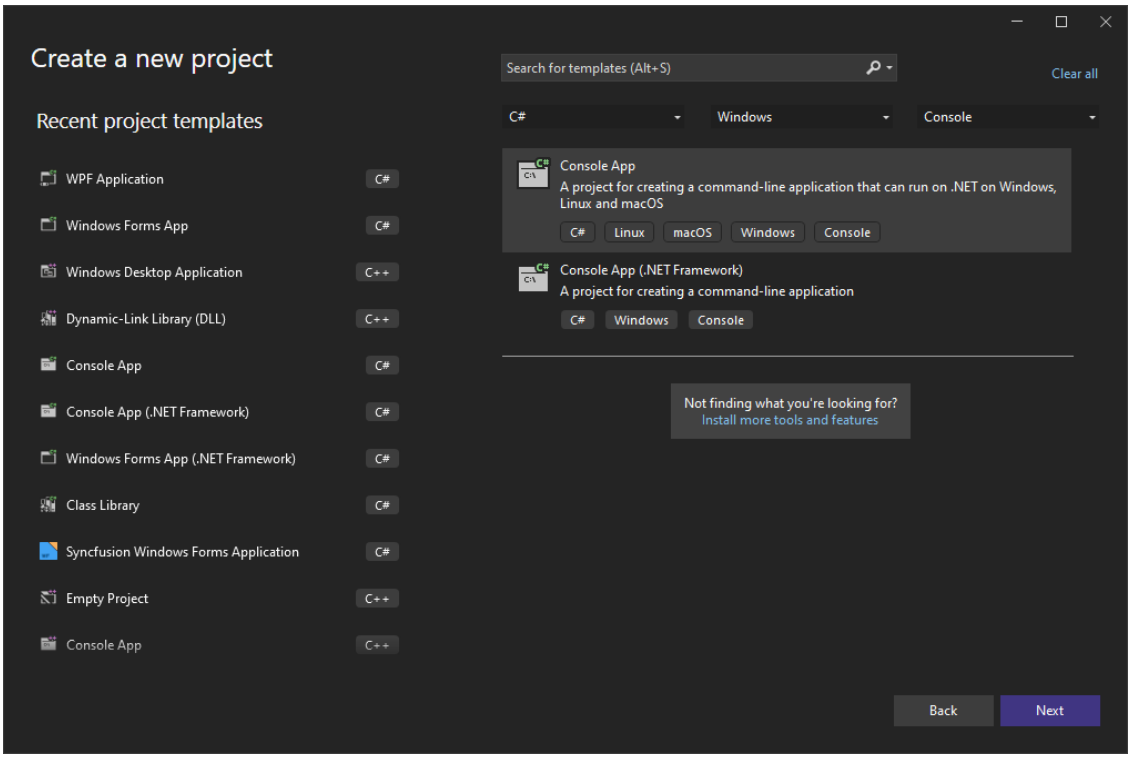
```
using astorino_lib;

namespace Test_astorino
{
    public partial class Form1 : Form
    {
        astorino r = new astorino();
        astorino.retVal ret = new astorino.retVal();
        astorino.JointPoints j_points = new astorino.JointPoints();
        astorino.TransformationPoints t_points = new astorino.TransformationPoints();
        byte status = 0xFF;

        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            List<string> ports = r.findUSB();
            if (ports != null)
            {
                status = r.Connect(ports.FirstOrDefault());
            }
            else
                throw new Exception("Robot is not found");
        }
```

ASTORINO C# API

```
        private void button3_Click(object sender, EventArgs e)
        {
            if (status == 0)
            {
                byte success = r.HOME(90, 90, 90);
                if (success != 0)
                {
                    throw new Exception("Robot motion failed!");
                }
            }
        }
        private async void button4_Click(object sender, EventArgs e)
        {
            if (status == 0)
            {
                byte success = await r.HOMEAsync(90, 90, 90);
                if (success != 0)
                {
                    throw new Exception("Robot motion failed!");
                }
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            r.Disconnect();
        }
    }
}
```

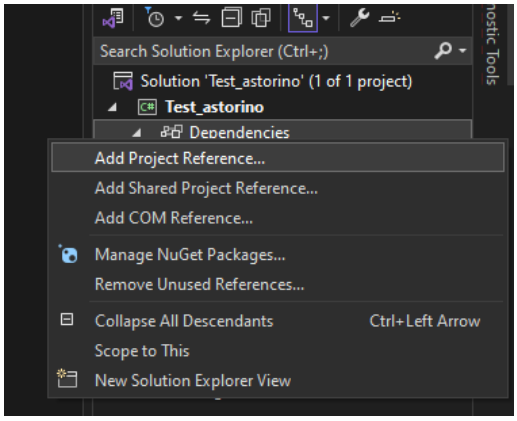
9.2 Visual Studio – Console Application

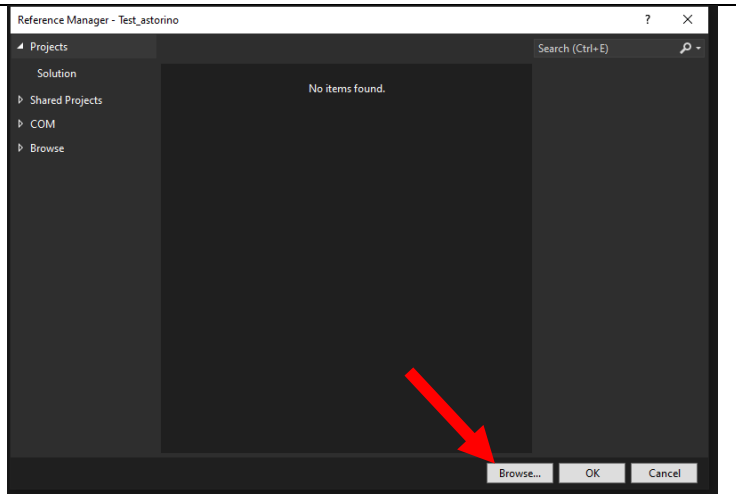
Create a new Console application:



Add Project Reference (right click on project Dependencies)

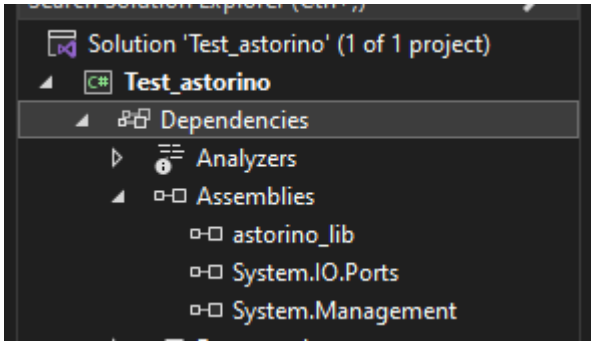
ASTORINO C# API





Select:

- astorino_lib.dll
- System.IO.Ports.dll
- System.Managment.dll

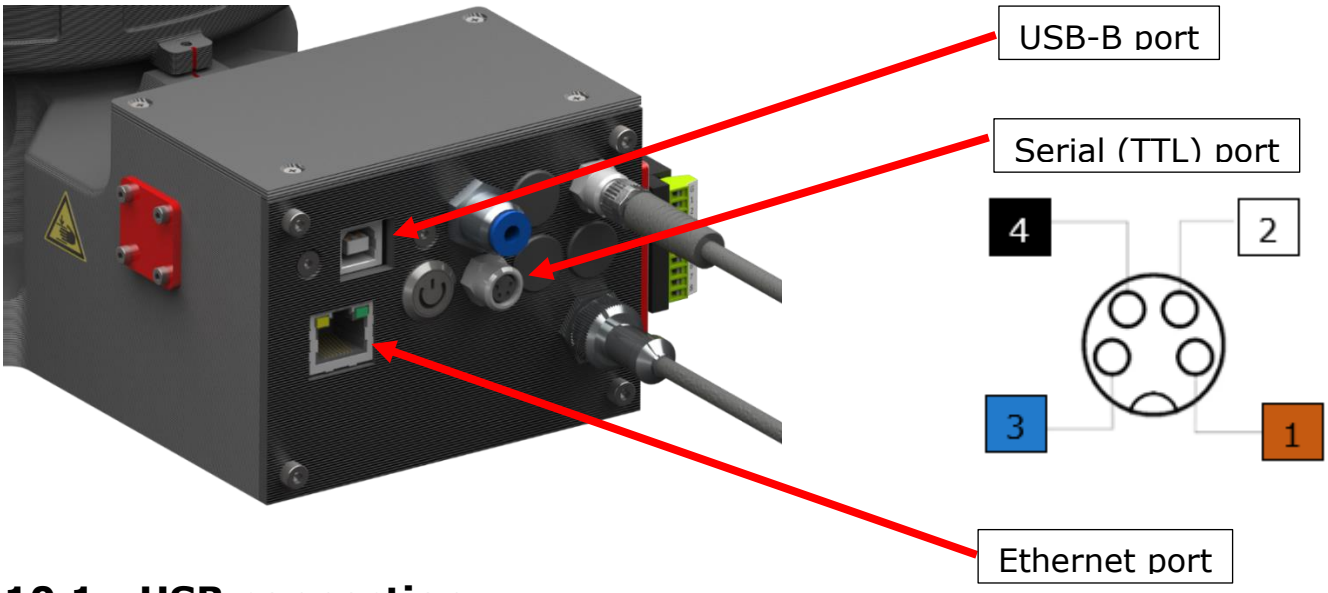


It should be loaded to current project:

Write your code, for example:

```
using astorino_lib;

astorino robot = new astorino();
astorino.retVal  retVal = new astorino.retVal();

Console.WriteLine("Hello, World!");


List<string> net = robot.findEthernet().GetAwaiter().GetResult();

if (net == null)
    throw new Exception("Robot not found");

if (robot.Connect(net.FirstOrDefault()) == 0)
{
    Console.WriteLine("Connected. Press enter key to go HOME");

    Console.ReadLine();

    if (robot.HOME(100, 90, 90) == 0)
        Console.WriteLine("Motion completed!");
    else
        Console.WriteLine("Motion failed!");

    Console.WriteLine("Finished. Disconnecting...");

    if (robot.Disconnect() == 0)
        Console.WriteLine("Disconnected");
    else
        Console.WriteLine("Failed");
}
```

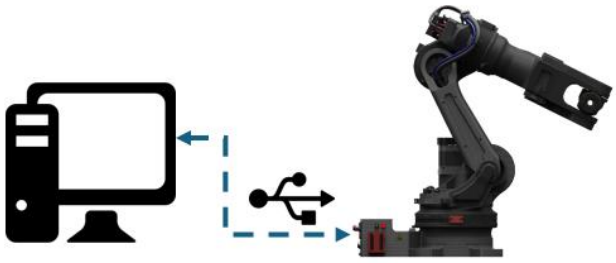
10 Hardware connection



10.1 USB connection

For USB connection use USB-B port in the robot base.

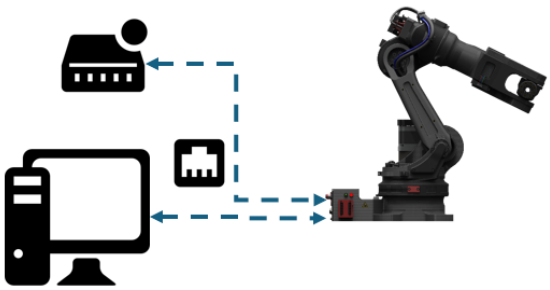
To use the USB as a communication port, no settings need to be configured in the Astorino software.



10.2 Ethernet connection

For Ethernet connection use Ethernet port in the robot base. Connect directly to the PC or use ethernet switch

To use the Ethernet as a communication port, the Ethernet settings need to be configured as connection in the Astorino software.



10.3 USB to UART (TTL) converter

For USB to UART (TTL) converter connection use Serial port in the robot base.

To use the Serial as a communication port, no settings need to be configured in the Astorino software.

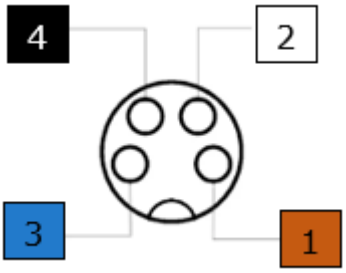
M8 connector pins:

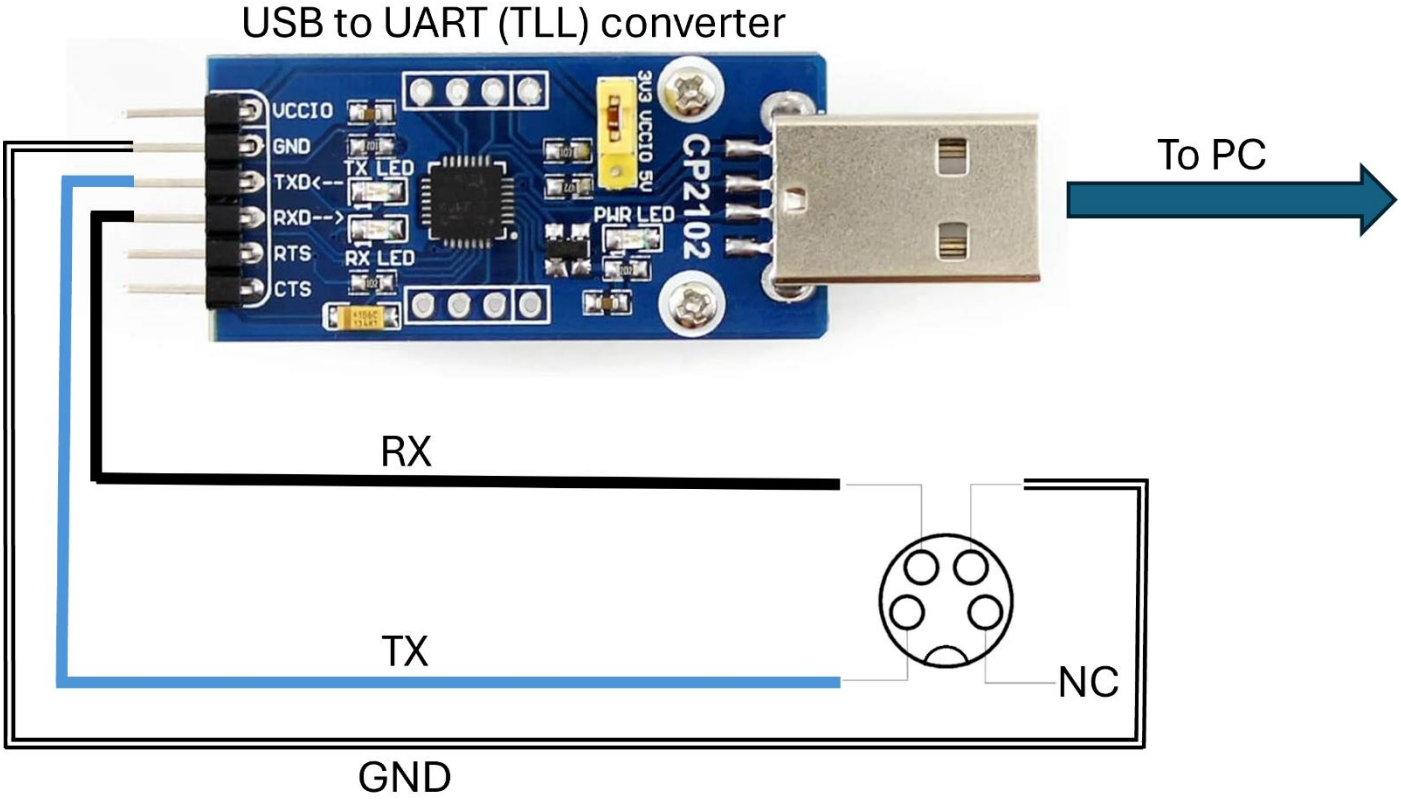
1 – 5V,

2 - GND,

3 – TX,

4 – RX





WARNING

Serial port (TLL) operates at 3.3V – connecting 5V might damage the CPU!

11 Manufacturer information

For further questions, contact Kawasaki Robotics support.

Contact:

Kawasaki Robotics GmbH

tech-support@kawasakirobot.de

+49 (0) 2131 – 3426 – 1310

Kawasaki Robot
C# API Manual

2024-09: 1st Edition

Publication: KAWASAKI Robotics GmbH