Kawasaki Robotics

ASTORINO

Communication protocol Manual

Preface

This manual describes the safety 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.

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.

Copyright © 2024 by KAWASAKI Robotics GmbH.

All rights reserved.

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 Communication protocol

5.1 Protocol Introduction

Astorino robot can be controlled by PC/Android/iOS/Arduino etc. achieving data transmission through certain communication protocols. The communication can be realized by USB-serial port(COM), 3.3V TLL level serial port, Ethernet TCP/IP.

Communication protocol features:

- CRC, please see 5.3 for checksum calculation
- Protocol command does not have fixed length,
- Protocol command consists of packet header, payload frame and check,
- Immediate commands. It will process the command once received and return back data, ACK or error code. If command cannot be executed when called, the appropriate error code is returned.
- All data is send in integer values,
- Float values are converted to int32_t,
- The parameters and data in the commands use Big-Edian mode.
- All units are [mm] and [deg]
- String data is transmitted with EOT char (0x03) that terminates string data.
- Communication must be started with a start communication command,
- Communication must be ended with end communication command,
- Send and received data is not encrypted,

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

5.2 Communication parameters

Type	Details	Description
USB to serial port (COM)	Baud rate	256000 bps
	Data bits	8 Bit
	Stop bit	1 Bit
	Parity bit	void
	DTR	Enabled
Serial port (3.3V TLL)	Baud rate	115200 bps
	Data bits	8 Bit
	Stop bit	1 Bit
	Parity bit	void
Ethernet TCP/IP	IP	192.168.0.1 (default)
	Port	23

⚠ WARNING

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

5.3 Checksum calculation

Check code (CRC) is CHECKSUM 8bit (uint8_t)

For example: command "01 02 27 CRC"

CRC = (0x01 + 0x02 + 0x27) & 0xFF = 0x2A & 0xFF = 0x2A

5.4 Float data conversion to 4 bytes

All float data is converted to int32_t in this method:

1. Float data is multiplied by 1000,
2. Multiplied float data is stored into int32_t integer,
3. Int32_t is divided into 4 bytes

For example pose x is -124.02. int32_t after multiplication is -124020 (HEX FF FE 1B 8C).

Algorithm for dividing int32_t into 4 bytes is as follow:

bytes0	bytes1	bytes2	bytes3
(value >> 24) & 0xFF	(value >> 16) & 0xFF	(value >> 8) & 0xFF	value & 0xFF

Result of that algorithm will be:

bytes0 = 0xFF

bytes1 = 0xFE

bytes2 = 1B

bytes3 = 8C

5.5 4 bytes conversion into float data

Float data that is transmitted as 4 bytes can be transformed using this method.

1. Create int32_t integer from 4 bytes,
2. Assign int32_t into float and divide by 1000;

For example pose x data is received as 4 bytes 0xFF 0xFE 0x1B 0x8C.

Algorithm for dividing int32_t into 4 bytes is as follow:

```
int32_t data = bytes0 << 24 | bytes1 << 16 | bytes2 << 8 | bytes3
```

Result of that algorithm will be:

int32_t data = FF FE 1B 8C

Float value after division by 1000 is -124.02.

5.6 Decode string data

All string data like program names, firmware version, serial number etc. is transmitted as bytes (ASCII) with EOT (0x03) terminator at the end of a text data.

For example program name “MAIN” is transmitted as [4E 41 49 4E 03] (frame head and CRC is skipped in this example)

To read the program name decode all the bytes until terminator (0x03) is found.

6 Commands

This section describes protocol commands frame, all data is displayed in HEX.

Used integer types:

- uint8_t – unsigned char 8-bit wide
- uint16_t – unsigned 16-bit integer/unsigned short
- uint32_t – unsigned 32-bit integer/unsigned long
- int32_t – signed 32-bit integer/signed long
- uint48_t – unsigned 48-bit integer

6.1 Common response commands

Those commands are widely used. Please refer to those commands in case of response commands from other functions.

6.1.1 Instruction completed successfully

Head	ID	CRC
01 02	06	

6.1.2 Instruction failed

Head	ID	Data	CRC
01 02	CC	Code	
		uint8_t	

CODE	DESCRIPTION
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
0X28	User already connected

6.1.3 Motion instruction completed

Head	ID	CRC
01 02	AA	

6.2 Communication start

This command must be send first after establishing the connection with astorino.

Command for TCP or COM port:

Head	ID	CRC
01 02	24	

Command for UART:

Head	ID	CRC
01 02	19	

Response if connected:

[instruction completed successfully]

Head	ID	CRC
01 02	06	

6.3 Communication end

This command must be send to end the communication with astorino.

Command:

Head	ID	CRC
01 02	25	

Response:

[instruction completed successfully]

Head	ID	CRC
01 02	06	

OR

[instruction failed]

Head	ID	Data	CRC
01 02	CC	Code	
		uint8_t	

6.4 Device information

Those commands are used to read data from astorino robot.

6.4.1 Read Status

Command:

Head	ID	CRC
01 02	27	

Response:

Head	ID	Data					CRC
01 02	27	Status1	Status2	Status3	Status4	Status5	
		uint8_t	uint8_t	uint8_t	uint8_t	uint8_t	

Bytes explanation:

7	6	5	4	3	2	1	0
Status1							
inHome	MotorOn	Repeat Mode	Hold	CycleOn	E-stop	Error	Ready
Status2							
EXT_IT	Safety-Fence	RepeatCont	StepOnce	Step Wait-ing	DryRun On	Zeroing done	inMotion
Status3							
IO-module-Active	H7	H6	H5	H4	H3	H2	H1
Status4							
Tool bits2	Tool bits1	Tool bits 0	Teach-Speed bit2	Teach-Speed bit1	Teach-Speed bit0	Modbus connected	Collision detection active
Status5							
inWork	Zeroing Running	Motion in Teach mode ac-tive	motion-Command active	inBase	inConv	inJoint	inTool

ASTORINO Communication protocol

Bits explanation:

- InHome – robot is in a HOME position,
- MotorOn – robot's motors are enabled,
- RepeatMode – robot is in Repeat Mode, if this is low then robot is in Teach Mode,
- Hold – robot in in a hold state,
- CycleOn – program is running,
- E-stop – emergency stop is active,
- Error – error is on,
- Ready – robot is ready for work, zeroing is done, no error state, emergency is off,
- EXT_IT – external hold – from dedicated signals,
- Safet-fence – safety fence stop is active,
- RepeatCont – program looping is active,
- StepOnce – program step by step execution is active,
- Step_Waiting – program is waiting for user next step command,
- DryRunOn – dry run mode is active,
- ZeroingDone – zeroing procedure was performed,
- inMotion – robot is executing motion command,
- IO-moduleActive – IO module is active,
- H1-H7 – end stops (zeroing sensors) state,
- Tool_bits2-Tool_bits0 – selected Tool number in binary (4 bit)

N/A	Tool_bits2	Tool_bits1	Tool_bits0
0	1/0	1/0	1/0

For example TOOL is 3 when:

N/A	Tool_bits2	Tool_bits1	Tool_bits0
0	0	1	1

- TeachSpeed bit2 - TeachSpeed bit0 – selected Teach speed, number in binary (4 bit)

N/A	TeachSpeed bit2	TeachSpeed bit1	TeachSpeed bit0
0	1/0	1/0	1/0

For example Teach speed is 5 when:

N/A	TeachSpeed bit2	TeachSpeed bit1	TeachSpeed bit0
0	1	0	1

- Modbus Connected – communication via modbus TCP is active,
- Collision detection active – collision detection module is active,
- Zeroing running – zeroing procedure is active,
- Motion in Teach mode active – robot is moving in Teach Mode,
- Motion command active – robot executes motion command in Teach Mode,
- inWork – Teach mode motion operates in WORK coordinate system,
- inBase - Teach mode motion operates in BASE coordinate system,
- inJoint - Teach mode motion operates in JOINT coordinate system,
- inTool - Teach mode motion operates in TOOL coordinate system,

6.4.2 Read IO

Command:

Head	ID	CRC
01 02	26	

Response:

Head	ID	Data							CRC
01 02	26	Inputs	Modbus Inputs	Arm In-put	Outputs	Modbus Outputs	Arm Outputs	Internal	
		uint8_t	uint48_t	Uint8_t	uint8_t	uint48_t	uint8_t	uint16_t	

or [Instruction failed]

Modbus Input data is divided into 3x 16bits registers,

Modbus Inputs					
uint48_t					
uint16_t		uint16_t		uint16_t	
High byte	Low byte	High byte	Low byte	High byte	Low byte
uint8_t	uint8_t	uint8_t	uint8_t	uint8_t	uint8_t

Modbus Output data is divided into 3x 16bits registers,

Modbus Outputs					
uint48_t					
uint16_t		uint16_t		uint16_t	
High byte	Low byte	High byte	Low byte	High byte	Low byte
uint8_t	uint8_t	uint8_t	uint8_t	uint8_t	uint8_t

Internal data is one16bits register,

Internal	
uint16_t	
High byte	Low byte
uint8_t	uint8_t

6.4.3 Read current Position – JT1 - JT7

Command:

Head	ID	CRC
01 02	28	

Response:

Head	ID	Data							CRC
01 02	28	JT1	JT2	JT3	JT4	JT5	JT6	JT7	
		int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

or [Instruction failed]

Please refer to 5.5 for data conversion. Units in [deg]

6.4.4 Read current JT velocity

Command:

Head	ID	CRC
01 02	67	

Response:

Head	ID	Data							CRC
01 02	67	dJT1	dJT2	dJT3	dJT4	dJT5	dJT6	dJT7	
		int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

or [Instruction failed]

Please refer to 5.5 for data conversion. Units in [deg/s]

6.4.5 Read current Position – XYZ OAT JT7

Command:

Head	ID	CRC
01 02	29	

Response:

Head	ID	Data							CRC
01 02	29	X	Y	Z	O	A	T	JT7	
		int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

or [Instruction failed]

Please refer to 5.5 for data conversion. Units in [mm] and [deg]

6.4.6 Read current TCP velocity

Command:

Head	ID	CRC
01 02	66	

Response:

Head	ID	Data							CRC
01 02	66	dX	dY	dZ	dRx	dRy	dRz	dJT7	
		int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

or [Instruction failed]

Please refer to 5.5 for data conversion. Units in [mm/s] and [deg/s]

6.4.7 Read serial number

Command:

Head	ID	CRC
01 02	2A	

Response:

Head	ID	Data	Data Terminator	CRC
01 02	2A	uint_8…	03	

or [Instruction failed]

Please refer to 5.6 for data conversion.

6.4.8 Read one position – Transformations

This function transfer all stored Transformation points.

Command:

Head	ID	CRC
01 02	5F	

Response per point:

Head	ID	Data								CRC
01 02	5F	index	X	Y	Z	O	A	T	JT7	
		uint8_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

or [Instruction failed]

Index is the number of a point P[0..99].

6.4.9 Read all positions – Transformations

This function transfer all stored Transformation points.

Command:

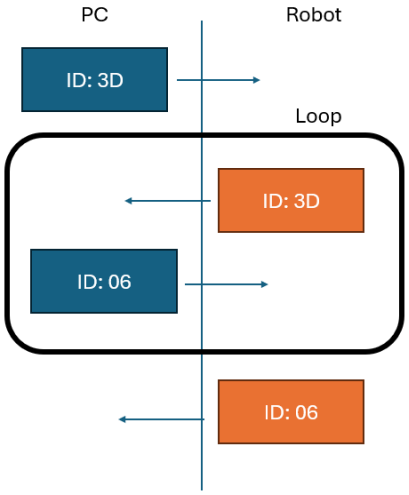
Head	ID	CRC
01 02	3D	

Response per point:

Head	ID	Data								CRC
01 02	3D	index	X	Y	Z	O	A	T	JT7	
		uint8_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

or [Instruction failed]

After each position is transmitted robot awaits for [instruction completed success-fully]. After all stored programs are transmitted [Instruction completed successfully] or [Command Failed] is send.



Index is the number of a point #P[0..99].

6.4.10 Read one position – Joints

This function transfer all stored Transformation points.

Command:

Head	ID	Data	CRC
01 02	60	Point number	
		uint8_t	

Response per point:

Head	ID	Data								CRC
01 02	60	index	JT1	JT2	JT3	JT4	JT5	JT6	JT7	
		int8_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

or [Instruction failed]

Index is the number of a point P[0..99].

6.4.11 Read accelerometer data

Command:

Head	ID	CRC
01 02	6E	

Response:

Head	ID	Data			CRC
01 02	6E	X	Y	Z	
		int32_t	int32_t	int32_t	

or [Instruction failed]

Accelerometer data contains data of 3 axes. Range is from -2G to 2G.

6.4.12 Read all positions – Joints

Command:

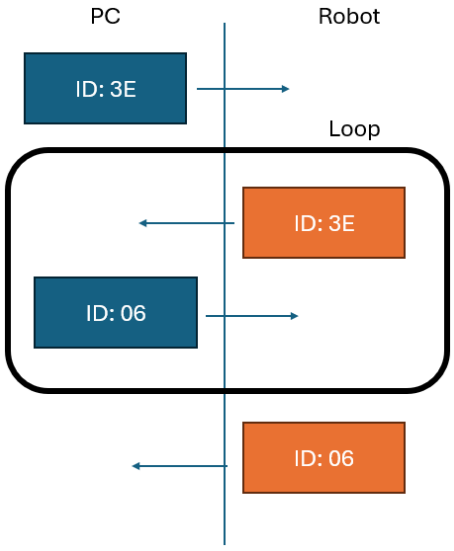
Head	ID	CRC
01 02	3E	

Response per point:

Head	ID	Data								CRC
01 02	3E	index	JT1	JT2	JT3	JT4	JT5	JT6	JT7	
		int8_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

or [Instruction failed]

After each position is transmitted robot awaits for [instruction completed successfully]. After all stored programs are transmitted [Instruction completed successfully] or [Command Failed] is send.



Index is the number of a point #P[0..99].

6.4.13 Read CPU temperature

Command:

Head	ID	CRC
01 02	5B	

Response:

Head	ID	Data	CRC
01 02	5B	uint_8	

or [Instruction failed]

Temperature is in degrees Celsius.

6.4.14 Read Firmware version

Command:

Head	ID	CRC
01 02	38	

Response:

Head	ID	Data	Data Terminator	CRC
01 02	38	uint_8…	03	

or [Instruction failed]

Please refer to 5.6 for data conversion.

6.4.15 Read all Programs names

Command:

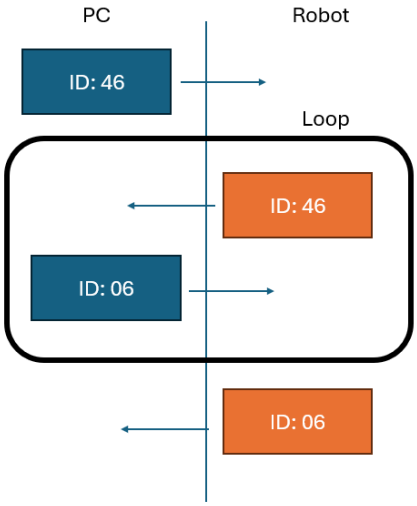
Head	ID	CRC
01 02	46	

Response:

Head	ID	Data	Data Terminator	CRC
01 02	46	uint_8…	03	

or [Instruction failed]

After each program name is transmitted robot awaits for [instruction completed successfully]. After all stored programs are transmitted [Instruction completed successfully] or [Command Failed] is send.



Please refer to 5.6 for data conversion.

6.4.16 Read Selected Program name

Command:

Head	ID	CRC
01 02	57	

Response:

Head	ID	Data	Data Terminator	CRC
01 02	57	uint_8…	03	

or [Instruction failed]

Please refer to 5.6 for data conversion.

6.4.17 Read Main Program name

Command:

Head	Data	CRC
01 02	47	

Response:

Head	ID	Data	Data Terminator	CRC
01 02	47	uint_8…	03	

or [Instruction failed]

Please refer to 5.6 for data conversion.

6.4.18 Read Tool data

Command:

Head	ID	Data	CRC
01 02	56	Tool number	
		uint8_t	

Response:

Head	ID	Data							CRC
01 02	56	T. num	X	Y	Z	O	A	T	
		uint8_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

or [Instruction failed]

Please refer to 5.5 for data conversion. Units in [mm] and [deg].

6.4.19 Read HOME position – JT1 - JT7

Command:

Head	ID	CRC
01 02	4A	

Response:

Head	ID	Data							CRC
01 02	4A	JT1	JT2	JT3	JT4	JT5	JT6	JT7	
		int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

or [Instruction failed]

Please refer to 5.5 for data conversion. Units in [deg]

6.4.20 Read Monitor Speed

Command:

Head	ID	CRC
01 02	5E	

Response:

Head	ID	Data	CRC
01 02	5E	uint_8	

or [Instruction failed]

Speed is in [%]

6.4.21 Read Error code

Command:

Head	ID	CRC
01 02	62	

Response:

Head	ID	Data	CRC
01 02	62	uint_8	

or [Instruction failed]

6.5 Set parameters instructions

All set parameters instructions responds with:

[instruction completed successfully]

Head	ID	CRC
01 02	06	

OR

[instruction failed]

Head	ID	Data	CRC
01 02	CC	Code	
		uint8_t	

6.5.1 Set Repeat Speed

Command:

Head	ID	Data	CRC
01 02	39	uint_8	

6.5.2 Set output

Command:

Head	ID	Data		CRC
01 02	3A	IO	State	
		uint8_t	uint8_t	

6.5.3 Set internal

Command:

Head	ID	Data		CRC
01 02	4C	Number	State	
		uint8_t	uint8_t	

6.5.4 Set HOME to current position

Command:

Head	ID	CRC
01 02	5D	

6.5.5 Set UART communication TimeOut

Command:

Head	ID	Data	CRC
01 02	6F	uint_8	

TimeOut is set as Data * 100ms, so if Data value is 50 (0x32) then TimeOut is 5000ms

TimeOut is used only in UART communication.

6.6 Write data instructions

All write data instructions respond with:

[instruction completed successfully]

Head	ID	CRC
01 02	06	

OR

[instruction failed]

Head	ID	Data	CRC
01 02	CC	Code	
		uint8_t	

6.6.1 Write HOME po data

Command:

Head	ID	Data							CRC
01 02	23	JT1	JT2	JT3	JT4	JT5	JT6	JT7	
		int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

Please refer to 5.4 for data conversion. Units in [deg].

6.6.2 Write Joint point data

Command:

Head	ID	Data								CRC
01 02	40	index	JT1	JT2	JT3	JT4	JT5	JT6	JT7	
		uint8_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

Index is the number of a point #P[0..99].

Please refer to 5.4 for data conversion. Units in [deg].

6.6.3 Write Transformation point data

Command:

Head	ID	Data								CRC
01 02	42	index	X	Y	Z	O	A	T	JT7	
		uint8_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

Index is the number of a point P[0..99].

Please refer to 5.4 for data conversion. Units in [mm] and [deg].

6.6.4 Write main program name

Command:

Head	ID	Data	Data Terminator	CRC
01 02	43	uint_8…	03	

Please refer to 5.6 for data conversion.

6.6.5 Write selected program name

Command:

Head	ID	Data	Data Terminator	CRC
01 02	44	uint_8…	03	

Please refer to 5.6 for data conversion.

6.6.6 Write Tool data

Command:

Head	ID	Data							CRC
01 02	48	T. num	X	Y	Z	O	A	T	
		uint8_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

Please refer to 5.4 for data conversion. Units in [mm] and [deg].

6.7 Actions

All action instructions respond with:

[instruction completed successfully]

Head	ID	CRC
01 02	06	

OR

[instruction failed]

Head	ID	Data	CRC
01 02	CC	Code uint8_t	

6.7.1 Motor On

Head	ID	CRC
01 02	20	

6.7.2 Motor Off

Head	ID	CRC
01 02	21	

6.7.3 Error reset

Head	ID	CRC
01 02	22	

6.7.4 Cycle On

Head	ID	CRC
01 02	2B	

6.7.5 Set TOOL 1

Head	ID	CRC
01 02	2C	

6.7.6 Set TOOL 2

Head	ID	CRC
01 02	2D	

6.7.7 Set TOOL 3

Head	ID	CRC
01 02	2E	

6.7.8 Hold

Head	ID	CRC
01 02	30	

6.7.9 Run (Hold off)

Head	ID	CRC
01 02	31	

6.7.10 DryRun On

Head	ID	CRC
01 02	32	

6.7.11 DryRun Off

Head	ID	CRC
01 02	33	

6.7.12 Repeat continues On

Head	ID	CRC
01 02	34	

6.7.13 Repeat continues Off

Head	ID	CRC
01 02	35	

6.7.14 Step once On

Head	ID	CRC
01 02	36	

6.7.15 Step once Off

Head	ID	CRC
01 02	37	

6.7.16 Cycle off

Head	ID	CRC
01 02	3C	

6.7.17 Set WORK 1

Head	ID	CRC
01 02	63	

6.7.18 Set WORK 1

Head	ID	CRC
01 02	64	

6.7.19 Next Step in Step Once mode

Head	ID	CRC
01 02	6C	

6.7.20 Skip Wait for a condition when Cycle is ON

Head	ID	CRC
01 02	6D	

6.7.21 Save current position as JOINT point

Command:

Head	ID	Data	CRC
01 02	3F	uint_8	

Data – point number 0..99

6.7.22 Save current position as Transformation point

Command:

Head	ID	Data	CRC
01 02	41	uint_8	

Data – point number 0..99

6.7.23 Remove point

Command:

Head	ID	Data		CRC
01 02	49	Index	Type	
		uint8_t	uint8_t	

Type:

0x01 – Transformation point

0x02 – Joints angle point

6.7.24 Cancel motion

Head	ID	CRC
01 02	45	

6.7.25 Emergency Stop

Head	ID	CRC
01 02	5A	

6.8 Motion instructions

All action instructions respond with:

[Motion instruction completed]

Head	ID	CRC
01 02	AA	

OR

[instruction failed]

Head	ID	Data	CRC
01 02	CC	Code	
		uint8_t	

To execute motion instruction astorino must be in REPEAT MODE, MOTORS must enabled and there cannot be any error.

6.8.1 Start zeroing

Command:

Head	ID	CRC
01 02	3B	

6.8.2 Go to Home

Head	ID	Data			CRC
01 02	2F	Speed	Acc	Dec	
		uint8_t	uint8_t	uint8_t	

Speed – [1-100] unit [%] of maximum joint speed,

Acc – acceleration [0-100] units [%]

Dec - deceleration [0-100] units [%]

6.8.3 Execute linear motion to a point

Command:

Head	ID	Data					CRC
01 02	4D	Type	Index	Speed	Acc	Dec	
		uint8_t	uint8_t	uint8_t	uint8_t	uint8_t	

Type:

- 0x01 – Transformation point
- 0x02 – Joints angle point

Index is the number of a point [0..99].

Speed – [1-250] unit [mm/s]

Acc – acceleration [0-100] units [%]

Dec - deceleration [0-100] units [%]

If Acc is 0 then initial motion speed is set to Speed,

If Dec is 0 then final motion speed is set to Speed.

6.8.4 Execute linear approach motion to a point (LAPPRO)

Command:

Head	ID	Data						CRC
01 02	69	Type	Index	Speed	Acc	Dec	Offset	
		uint8_t	uint8_t	uint8_t	uint8_t	uint8_t	Int32_t	

Type:

- 0x01 – Transformation point
- 0x02 – Joints angle point

Index is the number of a point [0..99].

Speed – [1-250] unit [mm/s]

Acc – acceleration [0-100] units [%]

Dec - deceleration [0-100] units [%]

If Acc is 0 then initial motion speed is set to Speed,

If Dec is 0 then final motion speed is set to Speed.

6.8.5 Execute PTP motion to a point

Command:

Head	ID	Data					CRC
01 02	4F	Type	Index	Speed	Acc	Dec	
		uint8_t	uint8_t	uint8_t	uint8_t	uint8_t	

Type:

- 0x01 – Transformation point
- 0x02 – Joints angle point

Index is the number of a point [0..99].

Speed – [1-250] unit [mm/s]

Acc – acceleration [0-100] units [%]

Dec - deceleration [0-100] units [%]

If Acc is 0 then initial motion speed is set to Speed,

If Dec is 0 then final motion speed is set to Speed.

6.8.6 Execute joint approach motion to a point (JAPPRO)

Command:

Head	ID	Data						CRC
01 02	68	Type	Index	Speed	Acc	Dec	Offset	
		uint8_t	uint8_t	uint8_t	uint8_t	uint8_t	Int32_t	

Type:

- 0x01 – Transformation point
- 0x02 – Joints angle point

Index is the number of a point [0..99].

Speed – [1-250] unit [mm/s]

Acc – acceleration [0-100] units [%]

Dec - deceleration [0-100] units [%]

If Acc is 0 then initial motion speed is set to Speed,

If Dec is 0 then final motion speed is set to Speed.

6.8.7 Execute linear motion to user data

Command:

Head	ID	Data			
01 02	4E	Type	Speed	Acc	Dec
		uint8_t	uint8_t	uint8_t	uint8_t

Data							CRC
val1	val2	val3	val4	val5	val6	val7	
int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

Type:

- 0x01 – Transformation point
- 0x02 – Joints angle point
- 0x03 – relative Transformation data in BASE coordinate system
- 0x04 – relative Transformation data in TOOL coordinate system
- 0x05 – relative Transformation data in WORK coordinate system

Speed – [1-250] unit [mm/s]

Acc – acceleration [0-100] units [%]

Dec - deceleration [0-100] units [%]

If Acc is 0 then initial motion speed is set to Speed,

If Dec is 0 then final motion speed is set to Speed.

6.8.8 Execute linear approach motion to user data (LAPPRO)

Command:

Head	ID	Data			
01 02	6A	Type	Speed	Acc	Dec
		uint8_t	uint8_t	uint8_t	uint8_t

Data								CRC
val1	val2	val3	val4	val5	val6	val7	Offset	
int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

Type:

- 0x01 – Transformation point
- 0x02 – Joints angle point

Speed – [1-250] unit [mm/s]

Acc – acceleration [0-100] units [%]

Dec - deceleration [0-100] units [%]

If Acc is 0 then initial motion speed is set to Speed,

If Dec is 0 then final motion speed is set to Speed.

6.8.9 Execute PTP motion to user data

Command:

Head	ID	Data			
01 02	50	Type	Speed	Acc	Dec
		uint8_t	uint8_t	uint8_t	uint8_t

Data							CRC
val1	val2	val3	val4	val5	val6	val7	
int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

Type:

- 0x01 – Transformation point
- 0x02 – Joints angle point
- 0x03 – relative Joint data

Speed – [1-100] unit [%]

Acc – acceleration [0-100] units [%]

Dec - deceleration [0-100] units [%]

If Acc is 0 then initial motion speed is set to Speed,

If Dec is 0 then final motion speed is set to Speed.

6.8.10 Execute joint approach motion to user data (JAPPRO)

Command:

Head	ID	Data			
01 02	6B	Type	Speed	Acc	Dec
		uint8_t	uint8_t	uint8_t	uint8_t

Data								CRC
val1	val2	val3	val4	val5	val6	val7	Offset	
int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

Type:

- 0x01 – Transformation point
- 0x02 – Joints angle point

Speed – [1-100] unit [%]

Acc – acceleration [0-100] units [%]

Dec - deceleration [0-100] units [%]

If Acc is 0 then initial motion speed is set to Speed,

If Dec is 0 then final motion speed is set to Speed.

6.8.11 Execute circular motion to user data

Circular motion requires two points to be executed. Middle (via) point and final (destination) point.

Command:

Head	ID	Data			
01 02	61	Type	Speed	Acc	Dec
		uint8_t	uint8_t	uint8_t	uint8_t

First point Data						
val1	val2	val3	val4	val5	val6	val7
int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t

Second point Data							CRC
val1	val2	val3	val4	val5	val6	val7	
int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

Type:

- 0x01 – Transformation point
- 0x02 – Joints angle point

Speed – [1-250] unit [mm/s]

Acc – acceleration [1-100] units [%]

Dec - deceleration [1-100] units [%]

6.8.12 Execute Circular motion through points

Command:

Head	ID	Data						CRC
01 02	65	Type	Index1	Index2	Speed	Acc	Dec	
		uint8_t	uint8_t	uint8_t	uint8_t	uint8_t	uint8_t	

Type:

- 0x01 – Transformation point
- 0x02 – Joints angle point

Index1 is the number of a middle point [0..99].

Index2 is the number of a final point [0..99].

Speed – [1-250] unit [mm/s]

Acc – acceleration [1-100] units [%]

Dec - deceleration [1-100] units [%]

6.9 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!

6.10 Turn on RTC

Command:

Head	ID	CRC
01 02	51	

Response:

[instruction completed successfully]

OR

[instruction failed]

6.11 Turn off RTC

Command:

Head	ID	CRC
01 02	52	

Response:

[instruction completed successfully]

OR

[instruction failed]

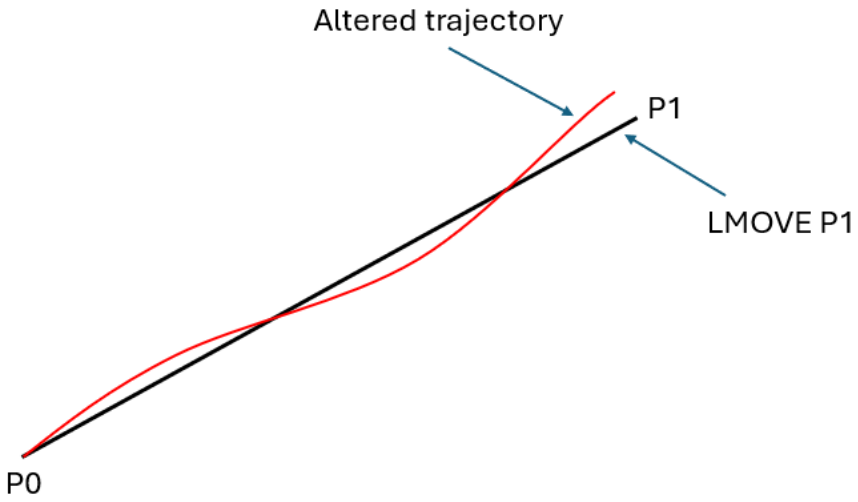
6.12 Set RTC offsets

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]

Command:

Head	ID	Data						CRC
01 02	53	x	y	z	rx	ry	rz	
		int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

Response:

[instruction completed successfully]

OR

[instruction failed]

6.13 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!

Command:

Head	ID	Data	
01 02	5C	Type	Duration
		uint8_t[0x01/0x02]	uint8_t

Data							CRC
val1	val2	val3	val4	val5	val6	val7	
int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

Duration is time in [ms] that the move must be executed in.

Type:

- 0x01 – Transformation point
- 0x02 – Joints angle point

WARNING

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

Response:

[Motion instruction completed]

OR

[instruction failed]

6.14 Auxiliary commands

6.15 Calculate forward kinematics

Command:

Head	ID	Data							CRC
01 02	54	JT1	JT2	JT3	JT4	JT5	JT6	JT7	
		int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

Response:

Head	ID	Data							CRC
01 02	54	X	Y	Z	O	A	T	JT7	
		int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

or [Instruction failed]

6.15.1 Calculate inverse kinematics

Command:

Head	ID	Data							CRC
01 02	55	X	Y	Z	O	A	T	JT7	
		int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

Response:

Head	ID	Data							CRC
01 02	55	JT1	JT2	JT3	JT4	JT5	JT6	JT7	
		int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	int32_t	

or [Instruction failed]

6.15.2 Roll Pitch Yaw (Rx,Ry,Rz) to OAT

Command:

Head	ID	Data			CRC
01 02	58	RX	RY	RZ	
		int32_t	int32_t	int32_t	

Response:

Head	ID	Data			CRC
01 02	58	O	A	T	
		int32_t	int32_t	int32_t	

or [Instruction failed]

6.15.3 OAT to Roll Pitch Yaw (Rx, Ry, Rz)

Command:

Head	ID	Data			CRC
01 02	59	O	A	T	
		int32_t	int32_t	int32_t	

Response:

Head	ID	Data			CRC
01 02	59	RX	RY	RZ	
		int32_t	int32_t	int32_t	

or [Instruction failed]

6.15.4 Execute AS command

Command:

Head	ID	Data	Data Terminator	CRC
01 02	4B	uint_8…	03	

Response:

If motion command is used:

[Motion instruction completed]

If non motion command is used:

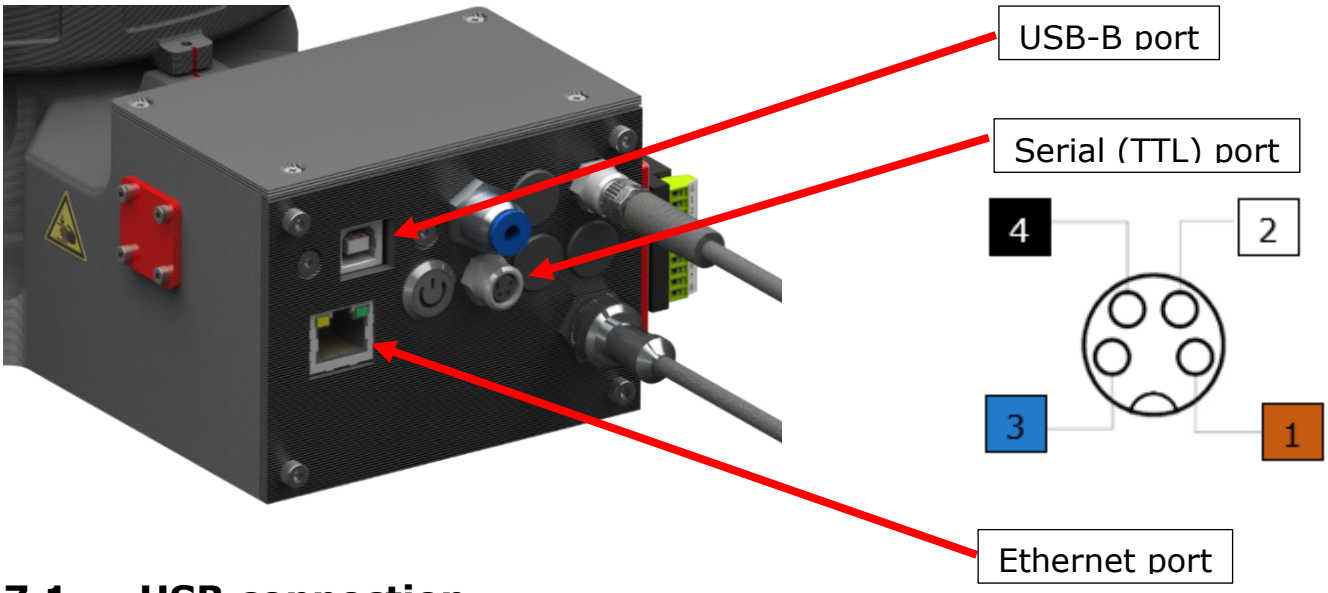
[instruction completed successfully]

Is command failed:

[Instruction failed]

Please refer to 5.6 for data conversion

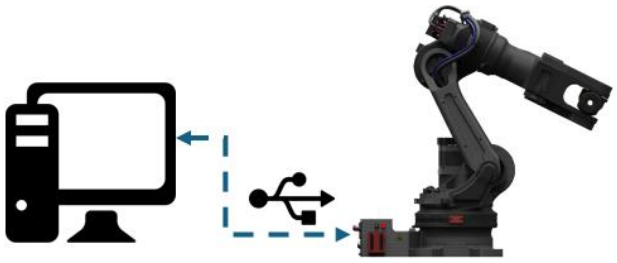
7 Hardware connection



7.1 USB connection

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

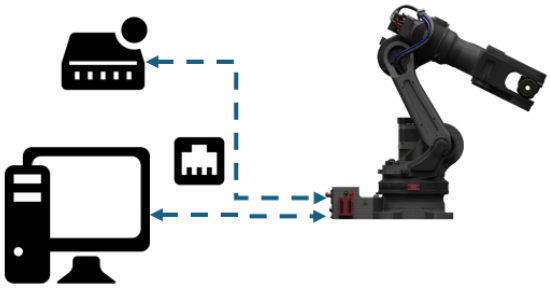
To use the USB as a communication port, no settings need to be configured in the Astorino software.



7.2 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.



7.3 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.

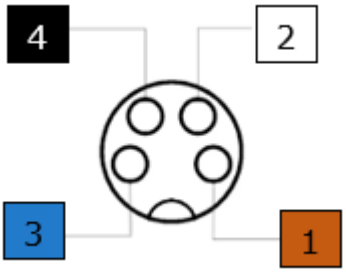
M8 connector pins:

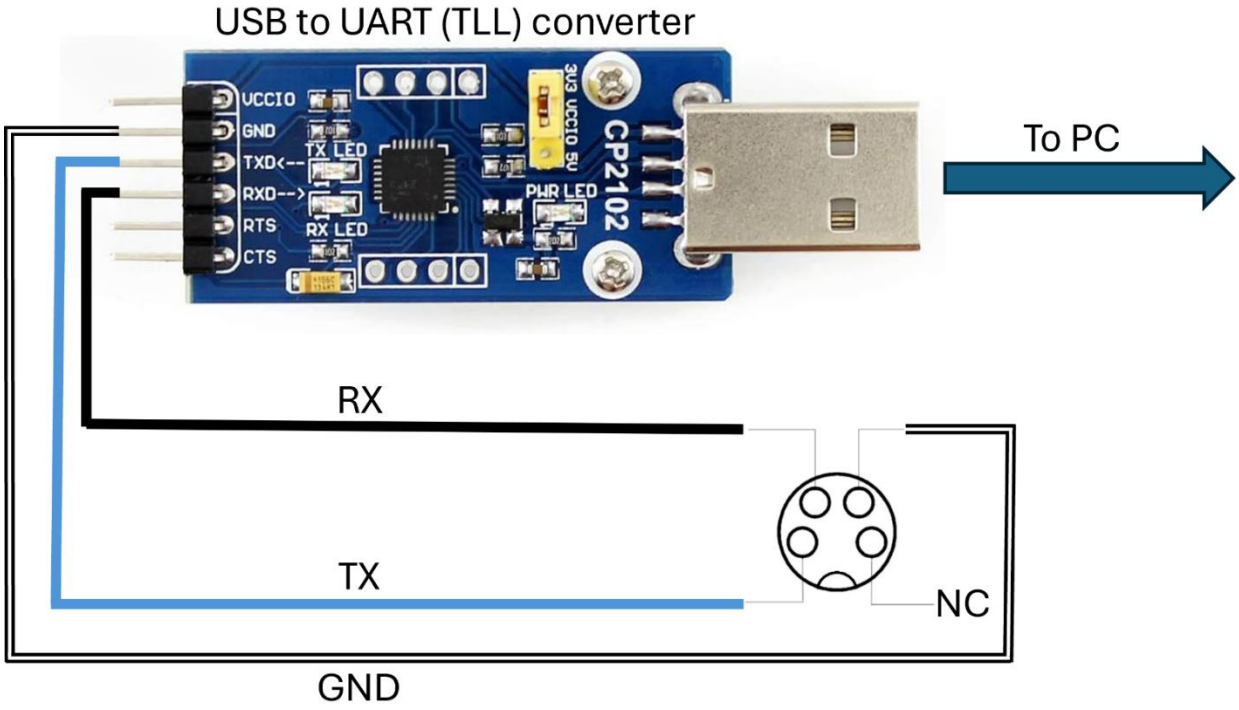
1 – 5V,

2 - GND,

3 – TX,

4 – RX





WARNING

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

7.4 ARDUINO or other Serial connection

For Serial (TTL) 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.

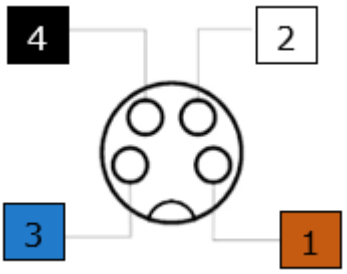
M8 connector pins:

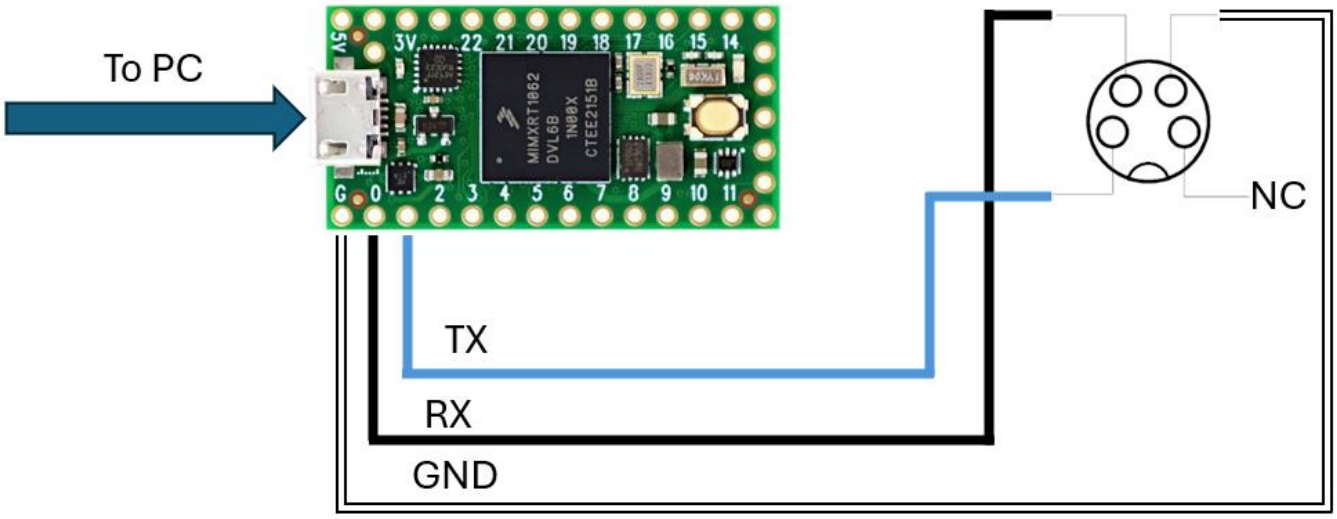
1 – 5V,

2 - GND,

3 – TX,

4 – RX





WARNING

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

8 Manufacturer information

For further questions, contact Kawasaki Robotics support.

Contact:

Kawasaki Robotics GmbH

tech-support@kawasakirobot.de

+49 (0) 2131 – 3426 – 1310

Kawasaki Robot
Communication protocol Manual

2024-09: 1st Edition

Publication: KAWASAKI Robotics GmbH

Copyright © 2024 by KAWASAKI Robotics GmbH.
All rights reserved.