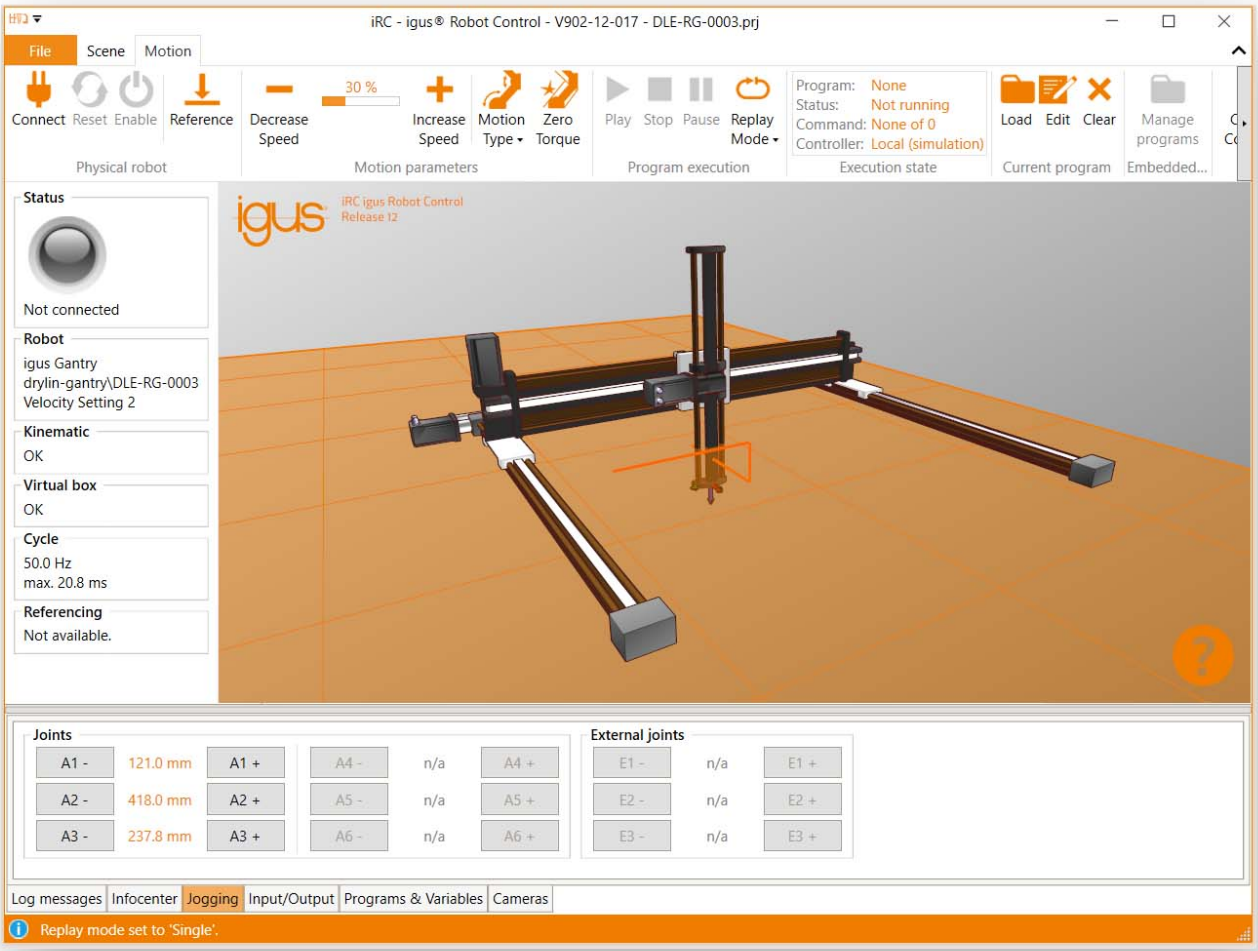
igus®



iRC - igus® Robot Control - V902-12-017 - DLE-RG-0003.prj

Program: None
Status: Not running
Command: None of 0
Controller: Local (simulation)

Status
Not connected

Robot
igus Gantry
drylin-gantry\DLE-RG-0003
Velocity Setting 2

Kinematic
OK

Virtual box
OK

Cycle
50.0 Hz
max. 20.8 ms

Referencing
Not available.

Joints

A1 -	121.0 mm	A1 +	A4 -	n/a	A4 +
A2 -	418.0 mm	A2 +	A5 -	n/a	A5 +
A3 -	237.8 mm	A3 +	A6 -	n/a	A6 +

External joints

E1 -	n/a	E1 +
E2 -	n/a	E2 +
E3 -	n/a	E3 +

Replay mode set to 'Single'.

LOW COST AUTOMATION by igus®

igus® Robot Control User Guide

Version 1.2 - preliminary EN version
December 2, 2021

Contents

1 Introduction

1.1 Contact

igus® GmbH
Spicher Str. 1a
D-51147 Köln

Tel.: +49(0)2203 / 96498-255
E-Mail: ww-robot-control@igus.net

Internet: www.igus.de

1.2 Intended Use

The intended use of the product is defined by the uses within the defined limits from the technical data. The permissible electrical parameters and the defined permissible ambient conditions must be observed in particular. These are specified in more detail later in this manual. The intended use for this product can be found in the following section 3.

1.3 Target Group and Qualification

The product and this documentation are intended for technically trained professionals such as development engineers, maintenance personnel, system integrators or application engineers. Installation, commissioning and operation may only be carried out by qualified personnel. These are persons who meet all the following requirements.

- have appropriate training and experience in the use of motors and their control systems
- know and understand the contents of this technical manual
- know the regulations in force

1.4 Symbols Used

All notes in this document follow a consistent form and are structured according to the following classes.

The WARNING notice alerts the reader to possible dangerous situations.
Disregarding a warning can **possibly** result in moderate injury to the user.

- Within a warning, this describes ways to avoid hazards.

This note indicates possible incorrect operation of the product.
Failure to comply with this notice may **possibly** esult in damage to this product or other products.

1.5 Product Safety

The following EU directives were observed:

- RoHS-Directive (2011/65/EU, 2015/863/EU)
- EMV-Directive (2014/30/EU)

1.6 Regulations

In addition to this technical manual, operation, commissioning is subject to the applicable local regulations, such as:

- Accident prevention regulations
- Local regulations for occupational safety

2 System Overview

The robot controller described here is part of a robot system that consists of four basic components:

1. **Robot**: the mechanical robot arm, gantry, delta robot, scara robot, etc ...
2. **Robot control**: modular robot controller embedded computer, stepper motor driver modules for controlling the robot axes and IO modules.
3. **Robot control software**: Control software for moving the robot and executing robot programs.
4. **Programming environment**: Graphical software for setting up robot programs

The robot controller is used to control the movement of the robot's motors. The robot controllers are available with 48V motor voltage or 24V motor voltage. The stepper motor driver modules are for bipolar stepper motors of different sizes and configurable, or preconfigured. For precise positioning and control, quadrature encoder signals are normally read in, which feed back the actual motor or axis position to the stepper motor driver modules. In addition, different referencing methods and movements are supported to transmit the "actual position" of the robot to the software after a cold start of the robot system. Different inverse kinematics are supported in the software, so that e.g. controlled linear movements of a robot arm are as easy to program as those of a gantry robot.

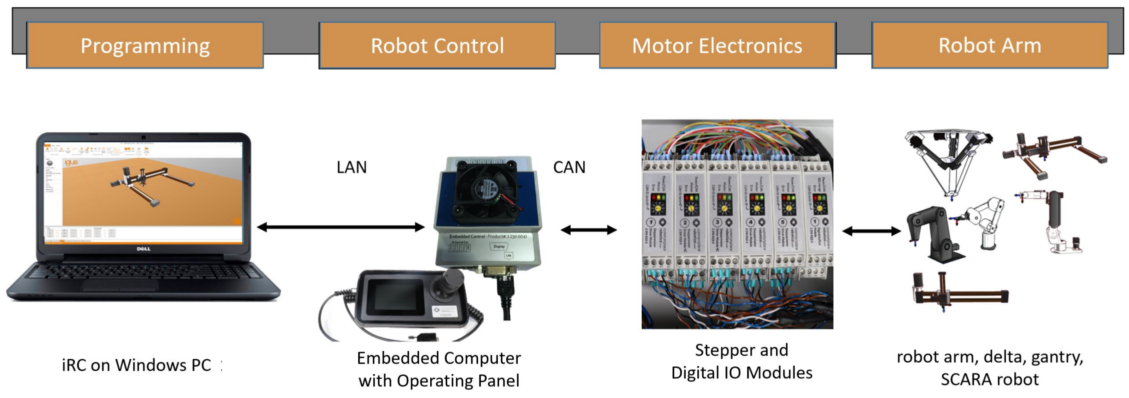

Programming
Robot Control
Motor Electronics
Robot Arm
LAN
CAN
iRC on Windows PC
Embedded Computer with Operating Panel
Stepper and Digital IO Modules
robot arm, delta, gantry, SCARA robot

Figure 1: **Robot system** - Robot, robot controller with integrated computer and optional handheld pendant. Separate Windows PC connected to the robot controller via LAN with iRC control software and programming environment. The Windows PC is required for programming the robot. Robot programs can also be executed without a Windows PC, since the robot controller uses the integrated computer as the control computer.

2.1 Supported Kinematics

The control hardware and software support a variety of kinematics that allow controlled movement of the robot. The following inverse kinematics are preconfigured.

- 3 axis delta robot
- 2 axis gantry robot (X and Y axis)
- 2 axis gantry robot (Y and Z axis)
- 3 axis robot arm (axis 1, 2, 3)
- 3 axis robot arm (axis 2, 3, 4)
- 4 axis robot arm (axis 1, 2, 3, 4)
- 4 axis robot arm (axis 2, 3, 4, 5)
- 5 axis robot arm (axis 1, 2, 3, 4, 5)

A complete list of supported robot models can be found in the "Open Project" section of iRC. In addition, up to three kinematically independent additional axes can be configured in iRC.

2.1.1 Reference Switches

Especially in order to be able to run linear movements and robot programs, it is important that the software knows in which position the robot is. During operation, the position is constantly monitored by quadrature encoders. After a cold start, however, it is necessary for each robot axis to move to a reference sensor whose position is known to the control hardware. This process is called referencing and is described in more detail in the section 9.3. A reference sensor of a linear axis is normally placed at the axis minimum. By definition, at the position of the reference sensor is the zero position. By an "Offset" (see list below) this position can be adjusted in firmware. For rotary axes, the 0° position must be set precisely (see "Offset" in the list below), especially in the case of robot arms, where inaccuracies in the 0° positions of several axes can add up. Robot arms that are delivered together with the controller have already been calibrated.

Different types of referencing can be set in the firmware parameters of the stepper motor modules. If a controller has been delivered for a specific robot model (see quality data sheet), these parameters are already set correctly. The firmware parameters can be downloaded, edited and written as described in section 11.4. Here are the most important parameters for setting the referencing behavior:

- Triggering behavior (rising/falling edge of the signal). **EndSwitchRising="True"** triggers on rising edge of the reference signal.
- Behavior after triggering. **StopOnEndSwitch="True"** stops the motor of the axis as soon as the reference signal has been detected. **StopOnEndSwitch="False"** allows a second, slower and more precise approach of the sensor.
- The encoder index signal can be used as a reference signal. **UseIndexAsRef="True"** ignores the limit switch of the axis and uses the index signal of the encoder. However, this is only useful for robots with output encoders, because for axes with gears the encoder index signal would trigger several times between axis minimum and axis maximum. Nor-

mally this value is set to "False" if a reference sensor is available.

- Reference run of a linear axis (to the axis minimum) **RefStraight="True"**. The motor rotation direction is defined as a sign together with the homing speed (see below).
- Sinusoidal homing of a rotation axis with a trigger pin. **RefSinus="True"** produces a sinusoidal search movement. The number of incrementally increasing homing oscillations is determined per **SinusRefMaxCycles="6"**.
- Homing of a rotary axis with half disks **RefHalf="True"**. Referencing in the direction specified by the half disk. Here, a half disk is mounted on the output of the axis so that the axis reference sensor is triggered from 0° to 179° and is not triggered from 0° to -179°. Here the transition point (i.e. the end of the half disk) is searched. The direction of rotation is defined as the sign of the referencing speed. Sensible values are between 1 and 100 (or -1 and -100).
- Referencing speed (fast, coarse) **RefSpeed="-20"** (unitless). The sign indicates the direction of motor rotation.
- Referencing speed (precise) **RefSpeedSlow="2"**. The sign indicates the direction of motor rotation. The slower, the more accurate. Sensible values are between 1 and 10 (or -1 and -10).
- If **RefFromBothSides="True"**, then the reference switch is approached from both sides and the center point is calculated.
- A **offset="-544"** (no unit) indicates the deviation of the reference position from the desired (actual) zero position of the axis. - In this way, the reference sensor does not have to be installed exactly on the zero position of the axis.

Generally, the reference sensors for linear axes are mounted near the axis minimum. If the stepper motor modules of the robot controller have been delivered preconfigured, the axis moves to the axis minimum during the referencing process. The maximum and minimum triggering distance, e.g. of a Hall reference encoder, is normally described in the data sheet of the reference encoder and depends on the material. When mounting the reference encoder, make sure that collisions with the reference encoder are avoided.

2.2 Specifications

Modular Control	Feature
Power supply	24V/5A and either 24V/10A or 48V/10A
. Type	DIN rail modules with 5-pole bus connector
Communication with control computer	CAN
Stepper motor module	For operation of a bipolar stepper motor
	quadrature encoder RS422
	12-24V reference switch input
Digital IO module	7 digital inputs (optocoupler)

Modular Control	Feature
	7 digital outputs, solid state relay, max 500 mA

Table 2: Modular Robot Control - Components

The features of the embedded control can be seen in the table below.

Embedded control	Features
Type	Single board computer with daughter module
Power supply	24V/1A
Operating system	Linux based
Software	TinyCtrl robot control software
Interfaces	CAN Bus (connection to the modules)
	Ethernet (connection to Windows PC)
	RS232 display connection
	Digital inputs for enabling switch
	DC/DC converter 5V/4A

Table 3: Integrated Computer - Specifications

This table lists the mechanical dimensions of the individual components in the control cabinet, as well as the control cabinet itself.

Objekt	L x B x H (mm)
control cabinet	600 x 300 x 155
control modules	100 x 22 x 92
embedded control	90 x 72 x 75
power supply 24V/10A	120 x 87 x 125
power supply 24V/5A	120 x 40 x 125
power supply 48V/10A	120 x 87 x 140
operation tablet	205 x 110 x 110

Table 4: Modular robot control - dimensions

3 Safety Instructions

Operate the robot carefully!
When operating a robot arm or commissioning a robot cell, always pay attention to the personal safety of the users and other persons! In particular, there must be no persons or obstacles in the working area of the robot.

- In the basic version, the robot control package does not contain any safety-related functions. Depending on the application, these may have to be added. See also "CE marking" below.
- CE marking: The robot arm and robot controller are part of a system that must be risk assessed in its entirety and comply with the applicable safety regulations to ensure personal safety. Depending on the outcome of the assessment, other safety components may need to be integrated. These are usually safety relays and door switches. The engineer commissioning the system is responsible.
- Depending on the equipment, the robot controller contains either a 24V/5A and a 24V/10A power supply or a 24V/5A and a 48V/10A power supply, which themselves require mains voltage (120 V / 240 V) depending on the configuration. Please check the label on the power supply. Only qualified personnel may connect the power supply to the mains and put it into operation.
- Work on the robot electronics should only be performed by qualified personnel. Check the guidelines for electrostatic discharge (ESD).
- Always disconnect the robot controller from the mains (120 V / 240 V) when working in the control cabinet or on electronics connected to the robot controller.
- DO NOT hot plug! This can cause permanent damage to the motor modules. Do not install or remove any modules or connectors (e.g. manual control unit, emergency stop switch, DIO modules or external relays, motor connections...) while powered up.
- The robot arm must be set up on a robust surface and screwed down or otherwise secured.
- Use and store the system only in a dry and clean environment.
- Use the system only at room temperature (15° to 32°C).
- The ventilation of the system must be able to work unhindered to ensure sufficient air flow for cooling the stepper motor modules. There must be at least 10 cm of space next to the fan of the robot controller. The fan must ideally point upwards or to the side (reduced efficiency). The fan must not point downwards.
- Save important data before installing the iRC - igus Robot Control.

4 Requirements

4.1 Environmental Conditions

Ambient conditions	Wert
Protection class	IP20
Ambient temperature (operating)	+10...+32°C
Ambient temperature (storage)	-10...+85°C
Humidity (non-condensing)	0...90%
Installation height above sea level (without power restriction)	1500m

Table 5: Ambient conditions

4.2 Software Requirements

To use iRC a computer with the following features is required:

- PC (min. Intel i5 CPU) with Windows 10 (64 Bit)
- .NET Framework 4.7.2 or newer
- at least 500MB free disk space
- graphics card (integrated or dedicated)
  - OpenGL 3.0 or newer
  - manufacturer driver (the standard Microsoft driver is not supported)
- a free USB port, if a robot is used via USB or USB-to-CAN adapter
- a free Ethernet port if a robot with integrated controller is used

5 Compatibility with previous Software Versions

From version 11 to version 12 of the iRCsoftware, adaptations were necessary that have an impact on the program execution.

Modified program execution!
Programs written with version 11 of the software can result in changed movements under version 12. A warning is issued when a corresponding program is loaded.

5.1 Changed Orientation of the TCP

The orientation of the Z-axis of the TCP for robot arms has been rotated by 180°. This adjustment was made to enable camera-based recording of workpieces.
It is possible to restore the behavior of version 11 via a parameter. This parameter is defined in the robot file, The tag ”TCPXDirToFront” in the line ”Kinematic”.

Standard in Version 12 is:
<Kinematic Type="CPRMover5" TCPXDirToFront="true" ... />

Standard in Version 11 was:
<Kinematic Type="CPRMover5" TCPXDirToFront="false" ... />

A change must be made both on the Integrated Control and on the Windows PC.

5.2 Smoothing and Acceleration Behavior

In version 12, the path is overground by a new algorithm and adapted to the maximum acceleration values. For this purpose, new acceleration values have been added to the robot file. Overgrinding is possible up to a look-ahead value defined in the robot file. Only sequences of motion commands of one type (joint or linear) are overground. Logic commands like DOut or the change from Joint to Linear interrupts the overgrinding.

5.3 Parameter Sets 24V and 48V

The projects can be loaded for motor voltages of 24V or 48V, a corresponding switch is in the project open menu. For the two variants different velocity and acceleration values are stored in the robot files.

6 Quick Start Guide

6.1 Set up and Connect

Before Starting the Work
To avoid injuries as well as damage to the components, observe the following instructions:

- Follow the safety instructions in section 3.
- Disconnect the robot or the controller from the mains. Never work on live parts. Work on control cabinets may only be carried out by qualified electricians.
- No hot plugging! Before plugging in or disconnecting modules/-plugs/electrical connections, disconnect the controller/robot from the mains.
- Ensure a safe stand of the robot and the controller.
- Observe the requirements for the environment 4.1.
- During the movements of the robot, no persons may be in the working area of the robot.

To set up and commission the robot, proceed in the following order:

1. Make sure that the controller is disconnected from the power supply: Unplug the power cord.
2. Mount the robot on a suitable base. Make sure that the cables are not under tension and that the sheet metal reference plate of axis 1 is not bent.
3. Feed the robot cables through the large circular hole in the control cabinet and plug them into the stepper motor modules. Each motor with its associated reference sensor is connected to its stepper motor module via 4 connectors (see Fig. 15). All connectors are labeled and coded to support this process. The following cables each belong to a motor the affiliation between motor and cable set is also numbered on the connectors:
   - motor cable (with the designation motor)
   - encoder cable (2 connectors labeled ENC-1 and ENC-2)
   - reference sensor (labeled End-Stop)
4. Connect the shield and ground wires of the encoder and motor cables, if present.
   - motor cable ground: green-yellow
   - encoder shield (ground): black
5. All motor and encoder cable ground wires are plugged into the ground block next to the controller.

6. Secure the robot cables against voltage, e.g. with a cable tie to one of the holes in the control cabinet. If present, plug in the display cable and secure it via the screw connection.

7. The robot is ready to be switched on after completing these steps.

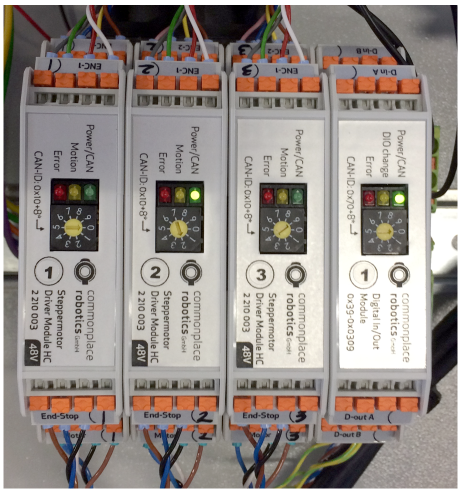


Figure 2: Three motor modules with 4 connectors per module, each leading to a motor and the corresponding reference sensor. Also in the picture: a digital I/O module (module on the right), which is not wired.

6.2 Switch On

Danger of Electric Shock! Before commissioning the components, make sure that all cables and control cabinet components are properly connected. Check all cables for tight fit and make sure that there are no loose cables in the control cabinet.

1. Connect the robot to your power supply using the enclosed power cable.

2. Switch on the robot using the On/Off switch on the control cabinet.

3. The green light emitting diodes (LEDs) on the modules will now light up, as will most of the red LEDs and possibly some of the yellow LEDs.

4. After the control computer has finished booting, the green LEDs will start flashing after about 60 s after the control system is switched on. This indicates communication with the modules. Now the robot controller is in operation. If available, you can now move the robot via the optional handheld operating device (see section 15).

6.3 Connecting and moving the robot

6.3.1 Preparation with the integrated computer

The following steps establish the LAN connection between the robot controller and the Windows PC.

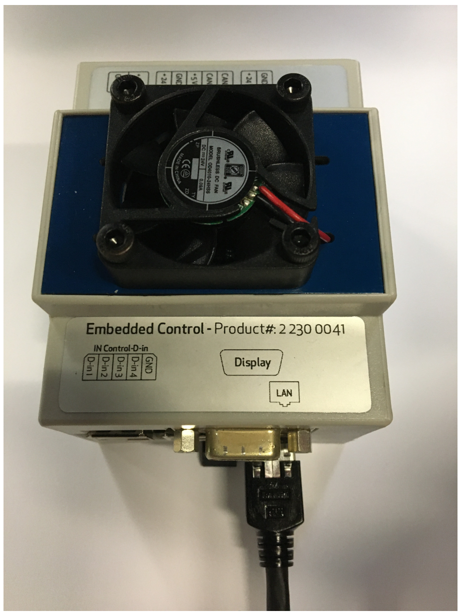


Figure 3: Integrated control computer with LAN cable.

1. Connect your PC to the robot controller using an Ethernet cable. Use the Ethernet port located right next to the USB socket on the integrated computer (see Fig. 3) of the robot controller.
2. Set the IP address of the PC to: static and 192.168.3.1 with a subnet mask of 255.255.255.0. Instructions for changing the IP address of your computer can be found on the Internet under the keyword "Change IP address Windows 10"..

6.3.2 Moving the Robot for the First Time

1. Install the iRC software on your PC.
2. Start the iRC software. At startup, you can select the project that applies to your robot. Please refer to the product number or product name, the project names are based on these (see Fig. 4).
3. You can now activate the robot by pressing the following buttons in the given order (see also Fig. 5):
   3.1 "Connect"
   3.2 "Reset"
   3.3 "Enable"
4. Now the status LED light on the left in iRC should be green and the status should show "No Error".
5. You can now move the robot's joints using the buttons on the "Jogging" tab (see Fig. 5).

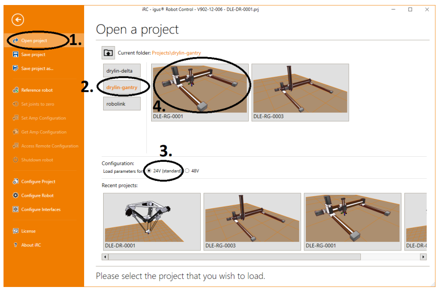


Figure 4: Choosing a project

Cartesian movements and programs
For the use of Cartesian commands and the playing of robot programs a referencing of the robot is mandatory.

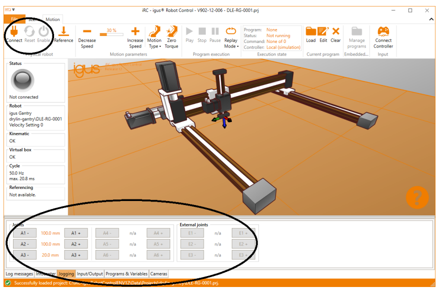


Figure 5: Jog Commands

7 CE-Conformity

7.1 CE marking

The modular robot controller is CE marked according to the RoHS, Low Voltage and EMC directives. The complete machine, i.e. the complete robot cell, is not covered by this CE marking. The complete machine is documented by the integrator as a complete system, assessed for risks and provided with a CE marking according to the Machinery Directive.

Creation of CE marking necessary! For the operation of the complete machine a CE marking according to the machine guideline must take place! This is carried out by the integrator or commissioning engineer!

7.2 Integration of safety components to establish CE conformity

The simplest way to establish the personnel safety required for CE conformity is to integrate safety components with the appropriate PL or SIL rating:

- Safety relay or safety control
- emergency off and door switch
- enclosure

The modular robot controller itself does not provide any safety functions. It does not carry a PL or SIL rating.

Two power supplies are provided in the controller, one for the logic (24V/5A), the other for the motor currents (24V/10A or 48V/10A). The first power supply should work continuously, the second one should be deactivated in case of emergency stop.
For this purpose, a relay or contactor can be used before or after the power supply:

- On the DC side, if 24V motor voltage is used. For 48V motor voltage, there are only a few safety relays that directly interrupt the DC currents.
- On the AC side, i.e. before the power supply. Here you have to test how long the overrun times are.

Do not interrupt the logic voltage! In case of emergency stop, the interruption of the motor voltage leads to a stop of the motors. The logic power supply for the Integrated Control must not be interrupted. Thus, the program can be continued or restarted after an EmergencyStop is released again without startup or referencing. For further details, refer to the circuit diagram of the control.

8 Installation

For the detailed pin assignment, please refer to the circuit diagram of the robot controller. The pin assignments in this chapter are intended exclusively for better illustration. Always consult the circuit diagram for the connection assignment.

8.1 Overview

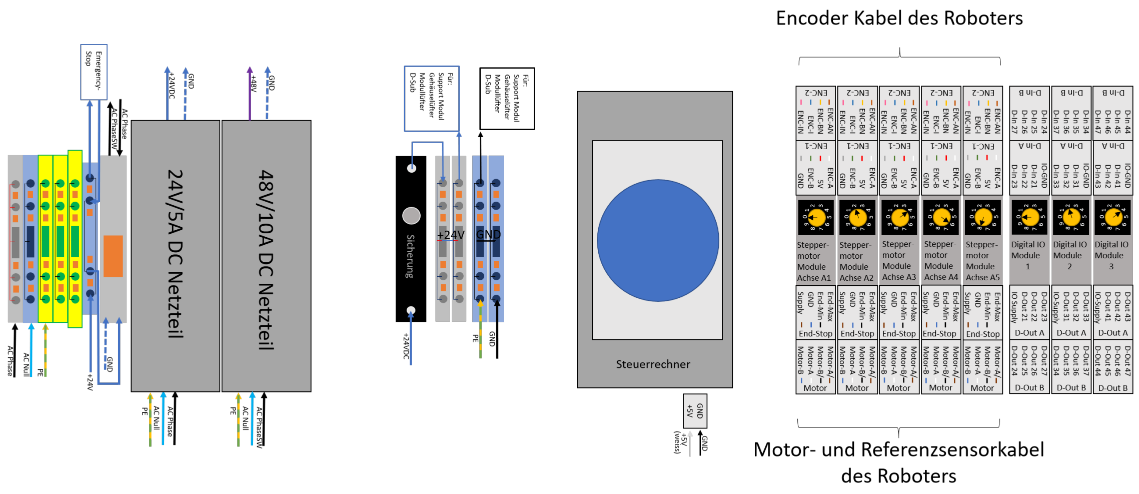

Encoder Kabel des Roboters
Emergency-Stop
24V/5A DC Netzteil
48V/10A DC Netzteil
Sicherung
+24V
GND
Steuerrechner
Stepper-motor Module Achse A1
Stepper-motor Module Achse A2
Stepper-motor Module Achse A3
Stepper-motor Module Achse A4
Stepper-motor Module Achse A5
Digital IO Module 1
Digital IO Module 2
Digital IO Module 3
Motor- und Referenzsensorkabel des Roboters

Figure 6: **Robotersteuerung** Shown here with 5 stepper modules and 3 DIO modules. In addition optional 24V/5A and 48V/10A power supplies for logic and motor voltage are shown. Detailed pin assignment in the separate circuit diagram.

8.2 Motor Module

Each stepper motor module drives a bipolar stepper motor with motor encoders. The encoder signals are evaluated by an RS422 module. The signals for each axis run over three cables: motor cable, encoder cable and reference switch cable. The motor cable is connected to a connector labeled "Motor", the encoder cable is connected to two connectors labeled "ENC 1" and "ENC-2", and the reference switch cable is connected to the connector labeled "End-Stop".

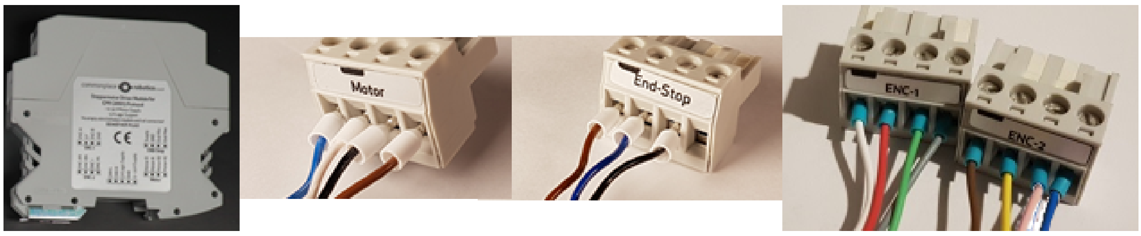


Figure 7: Motor module with connectors

Motor connector ("Motor"): Connects a bipolar stepper motor:

- Pin 1 (left): B (blue)
- Pin 2: A (white)
- Pin 3: B/ (black)
- Pin 4: A/ (brown)

Encoder connector 1 (ENC-1): Connects a quadrature encoder to an RS422 device.

- Pin 1 (left): A (white)
- Pin 2: 5 V DC (red)
- Pin 3: B (green)
- Pin 4: 0 V (grey)

Encoder connector 2 (ENC-2): Connects a quadrature encoder to an RS422 device.

- Pin 1 (left): A-N (brown)
- Pin 2: B-N (yellow)
- Pin 3: index (rose)
- Pin 4: index-N (blue)

cAll eight wires (encoder connectors 1 and 2) must be connected to read out the encoder.
End-Stop connector: Connected to a limit stop or reference switch.

- Pin 1 (left): 24 V (brown)
- Pin 2: Ground (GND) (blue)
- Pin 3: Signal (black)
- Pin 4: not connected

8.3 Digital I/O Module

The DIO module provides input and output channels, e.g. for controlling a gripper valve. The outputs can switch up to 500 mA. The inputs use optocouplers and are compatible with input voltages between 12 and 24 V.
A circuit switched by the output relays must not contain larger capacitors. If the current exceeds 500 mA, the solid state relays may be damaged.
For safety reasons, the inputs and outputs of the DIO module are galvanically isolated. This means that the circuits of the inputs and outputs are not connected to the internal circuits of the controller. It is therefore necessary to connect a supply voltage for the outputs and a ground line for the inputs. For this purpose, the 24 V available in the robot controller can be used, but also an external independent voltage source. In the software, the inputs and outputs of the first DIO module are numbered 21-27, of the second DIO module (if supplied) 31-37 and of the third 41-47.

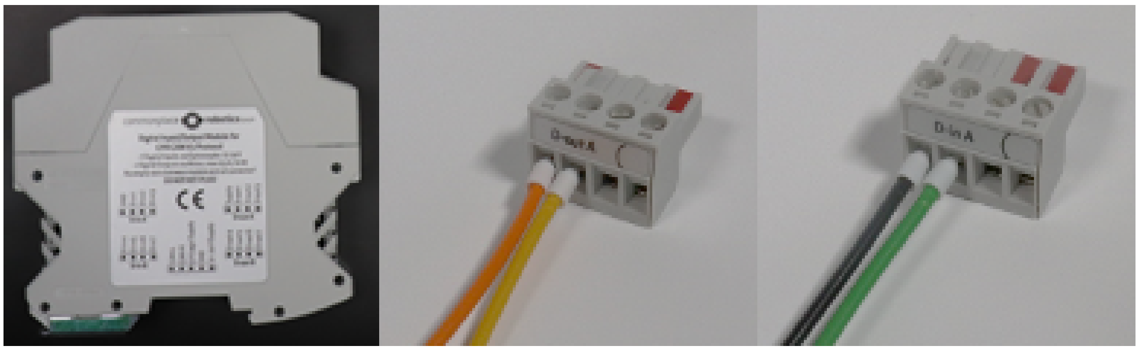


Figure 8: DIO module with connectors

Digital-Out connector A (D-out A): The output relays connect the power supply pin to the corresponding output pins.

- Pin 1 (left): Supply for all seven channels
- Pin 2: D-Out channel 1 (in software: Dout21)
- Pin 3: D-Out channel 2 (in software: Dout22)
- Pin 4: D-Out channel 3 (in software: Dout23)

Digital Out Connector B (D-out B): The D-out B pins are (from left to right) Digital Out channels 4-7 (image not shown).
Digital-In connector A (D-in A): Pin 1 in D-In A is the ground pin for all input pins.

- Pin 1 (left): Signal ground for all 7 channels
- Pin 2: D-In channel 1 (in software Din21)
- Pin 3: D-In channel 2 (in software Din22)
- Pin 4: D-In channel 3 (in software Din23)

Digital-In connector B (D-In B): The D-In B pins are (from left to right) the digital-in channels 4-7 (picture not shown).

8.3.1 DIO module structure

The easiest way to connect a programmable logic controller (PLC) is via digital inputs and outputs. Each robot controller is supplied with a DIO module. This provides 7 inputs and 7 outputs. If additional inputs and outputs are required, up to two additional DIO modules can be integrated. In total, up to 3 DIO modules can be controlled.
The outputs are controlled by solid state relays and can switch up to 500 mA. This value must not be exceeded during the switching process (e.g. by charging currents of capacitors) to prevent damage to the relays.

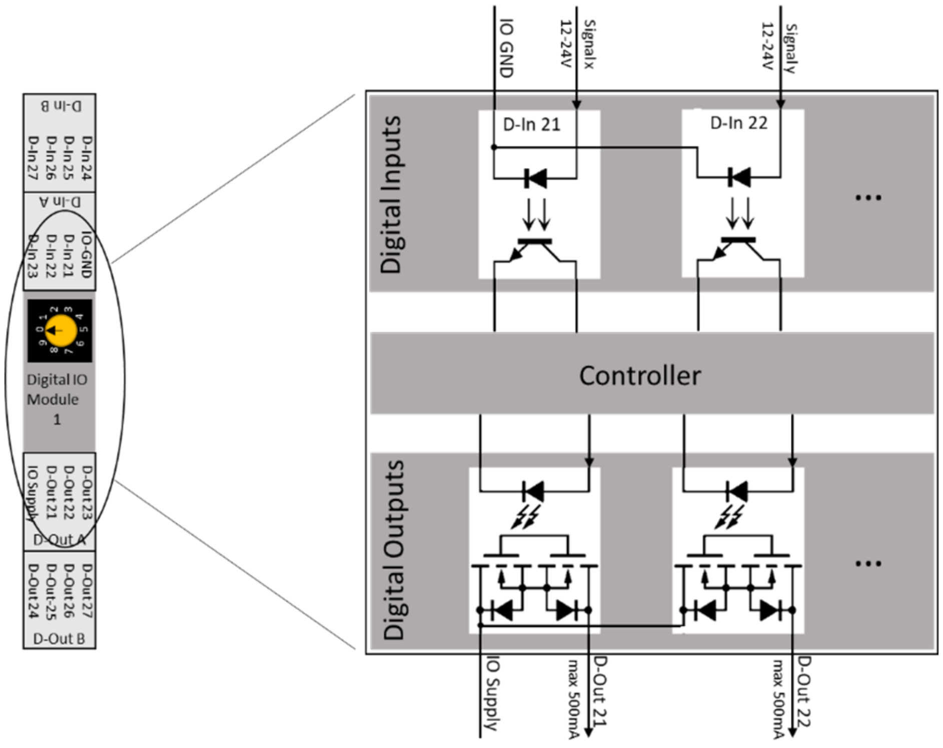


Figure 9: Internal structure of a DIO module

The inputs and outputs are galvanically isolated from the robot controller. A power supply (labeled "IO Supply" in the picture above) must be connected. The integrated 24 V supply can also be used.

8.3.2 Connecting sensors

- Pin 1 (GND) of the D-in 1 connector must be connected to the negative terminal (-) of the sensor power supply.
- The sensor signal (positive) must be connected to an input pin: D-in 1 connector pins 2-4 or D-in 2 connector pins 1-4.
- The positive side of the sensor (+) must be connected to the power supply of the sensor.
- The status of the inputs can be monitored in the "Input/Output" tab at the bottom of iRC.
- Inputs can be queried and reacted to in a robot program, e.g. in an if-then-else command....

8.3.3 Connect motors

- Pin 1 (supply voltage) of the D-out 1 connector must be connected to a power supply (e.g. 24V).

- The actuator (relay etc.) is then supplied with power via a free pin of the D-out connectors (D-out 1 pin 2-4 and D-out 2 pin 1-4).
- You can set the outputs manually in the "Input / Output" tab at the bottom of iRC.
- In a robot program you can set the state of the outputs with the digital-out command in iRC.

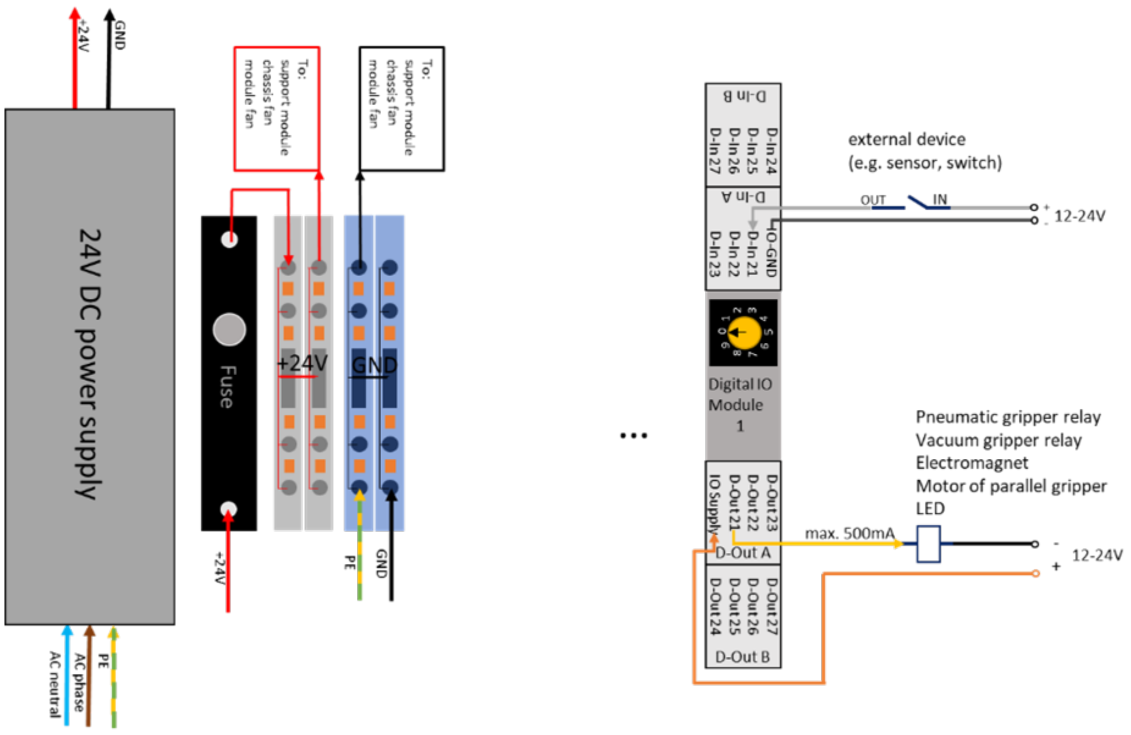


Figure 10: Example: Connection of a sensor and an motor

8.4 Embedded Control

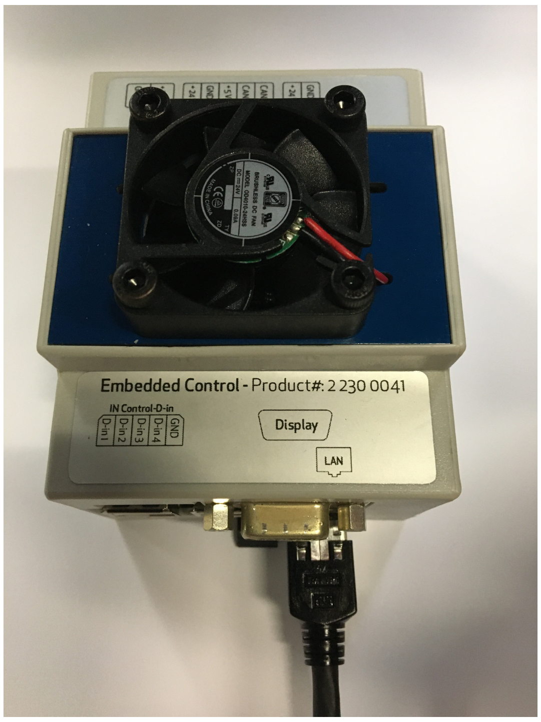


Figure 11: Embedded control

An integrated computer is combined with the modular robot controller so that an external computer is only required for programming, but not for daily operation. The Windows iRCsoftware basically communicates via LAN with the control software on the integrated computer (Linux). The control software communicates via CAN with the modules of the robot controller.

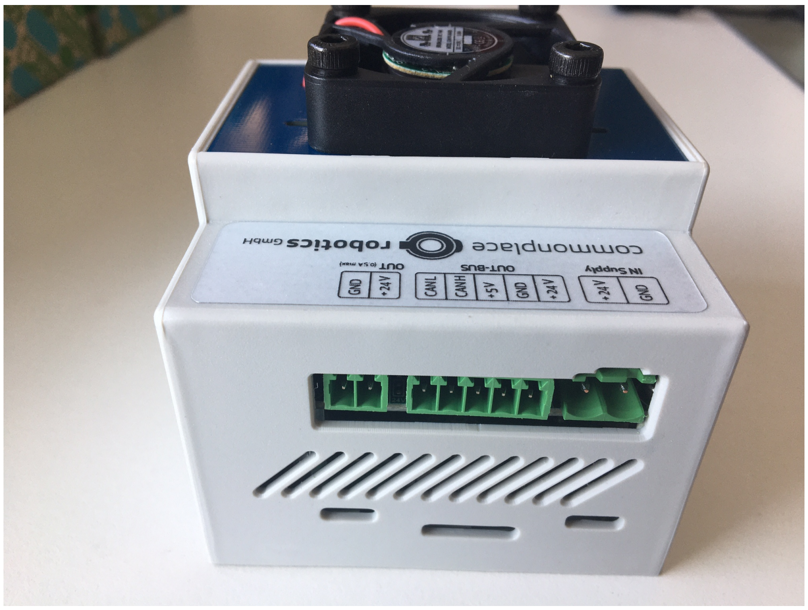


Figure 12: Bus and power supply connector. The "+24V Out" connector remains unused.

8.4.1 Power Supply

At the connector "IN Supply" of the control computer (rightmost in figure 12) is connected (from left to right):

- Pin1: +24V
- Pin2: GND

8.4.2 Bus connection

The CAN connection is established via the middle 5-pin connector "OUT-BUS" (from left to right):

- Pin1: CAN-L
- Pin2: CAN-H
- Pin3: +5V logic supply for the modules
- Pin4: not connected
- Pin5: not connected

The bus connection is connected to the DIN rail modules (digital IO and motor modules) of the controller. A support module is not compatible with this type of connection, as it would also feed +5V to the same bus.

8.4.3 LAN Connection to the Windows PC

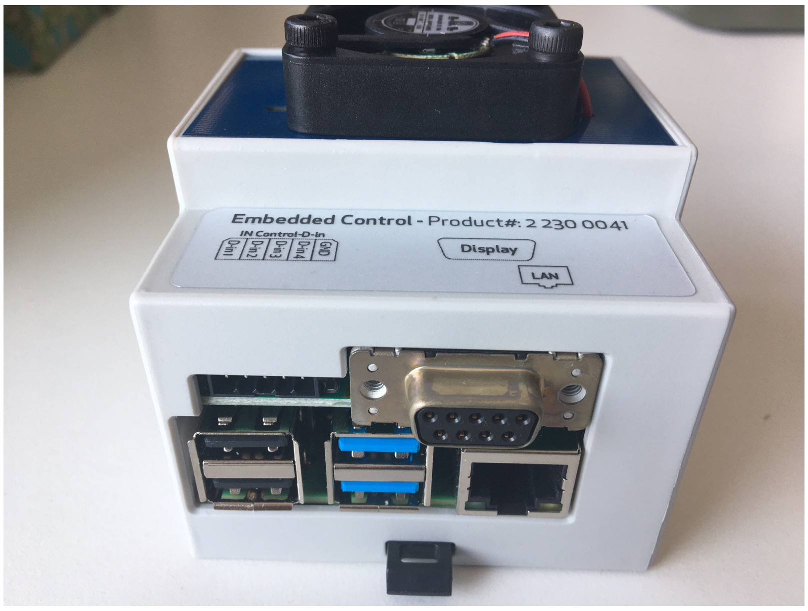


Figure 13: LAN and display port. The USB ports remain unused.

Ethernet connection between control computer and Windows PC can be established via LAN cable. The IP of the control computer is 192.168.3.11, subnet is 255.255.255.0.

8.4.4 Connection to the Manual Control Unit

The manual control unit (display) is connected to the 9-pin D-Sub connector. Attention! The connection is proprietary. A null modem cable or similar cannot be connected here. The pin assignment is as follows (the pins are numbered on the connector):

- Pin 1: GND
- Pin 2: CAN-H
- Pin 3: RS232 RX
- Pin 4: GND
- Pin 5: +24V out
- Pin 6: CAN-L
- Pin 7: RS232 TX
- Pin 8: not connected
- Pin 9: GND

8.5 Handheld

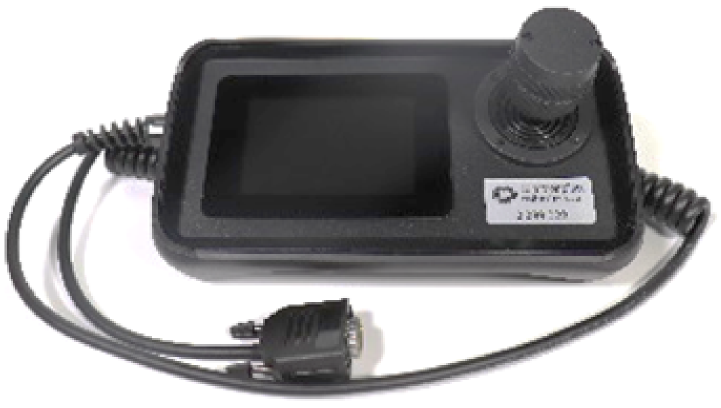


Figure 14: Handheld control unit (touch display and 3-axis joystick)

The handheld control unit is used to control the robot via the integrated computer. The connector of the D-Sub 9 plug must be connected to the D-Sub 9 socket of the control cabinet. The pin assignment of the handheld terminal is as follows:

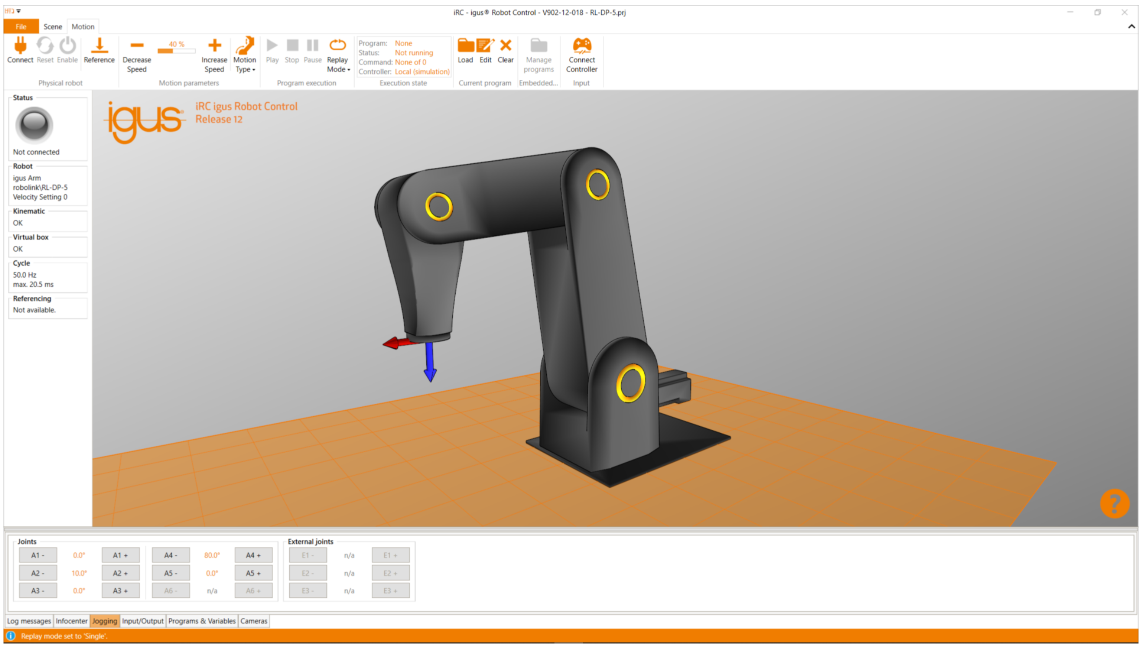
- Pin 1: not connected
- Pin 2: not connected
- Pin 3: RS232 TX
- Pin 4: not connected
- Pin 5: +24V input
- Pin 6: not connected
- Pin 7: RS232 RX
- Pin 8: not connected
- Pin 9: GND

9 Moving the Robot with iRC

iRC - igus Robot Control is a control and programming environment for robots. The 3D user interface helps to get the robot up and running quickly. Due to the modular structure, different kinematics and motor drivers can be controlled.
This operating manual is supplemented by the respective robot-specific operating manual.

9.1 The Graphical User Interface

This section explains the iRC software. All steps can be simulated even without a robot connected. In section 9.2 the real robot is then connected and moved.
The programming environment iRC enables the control and programming of the robot. You can work both online and offline, i.e. with the robot or in simulation (with the robot switched off or not connected).


iRC - igus® Robot Control - V902-12-018 - RL-DP-5.prj

Program: None
Status: Not running
Command: None of 0
Controller: Local (simulation)

Status: Not connected
Robot: igus Arm robolink\RL-DP-5 Velocity Setting 0
Kinematic: OK
Virtual box: OK
Cycle: 50.0 Hz max. 20.5 ms
Referencing: Not available.

Joints: A1 0.0°, A2 10.0°, A3 0.0°, A4 80.0°, A5 0.0°, A6 n/a
External joints: E1 n/a, E2 n/a, E3 n/a

Replay mode set to 'Single'.

Figure 15: User interface of the iRC - igus Robot Control

In the upper left corner, the three tabs "File", "Scene" and "Motion" provide access to the main functions. In the left corner, information about the current state of the physical robot is displayed. Additional functions like loading another project ("Open Project") or "Robot Referencing" can be found in the "File" tab (see Fig. 15).
There are five tabs at the bottom of the window:

- "Log messages": Messages from the program about status or errors.
- "Infocenter": displays the axis values, Cartesian position and other information.
- "Jogging": keys to move the robot.
- "Input/Output": Display and set the DIO interfaces of the robot controller.

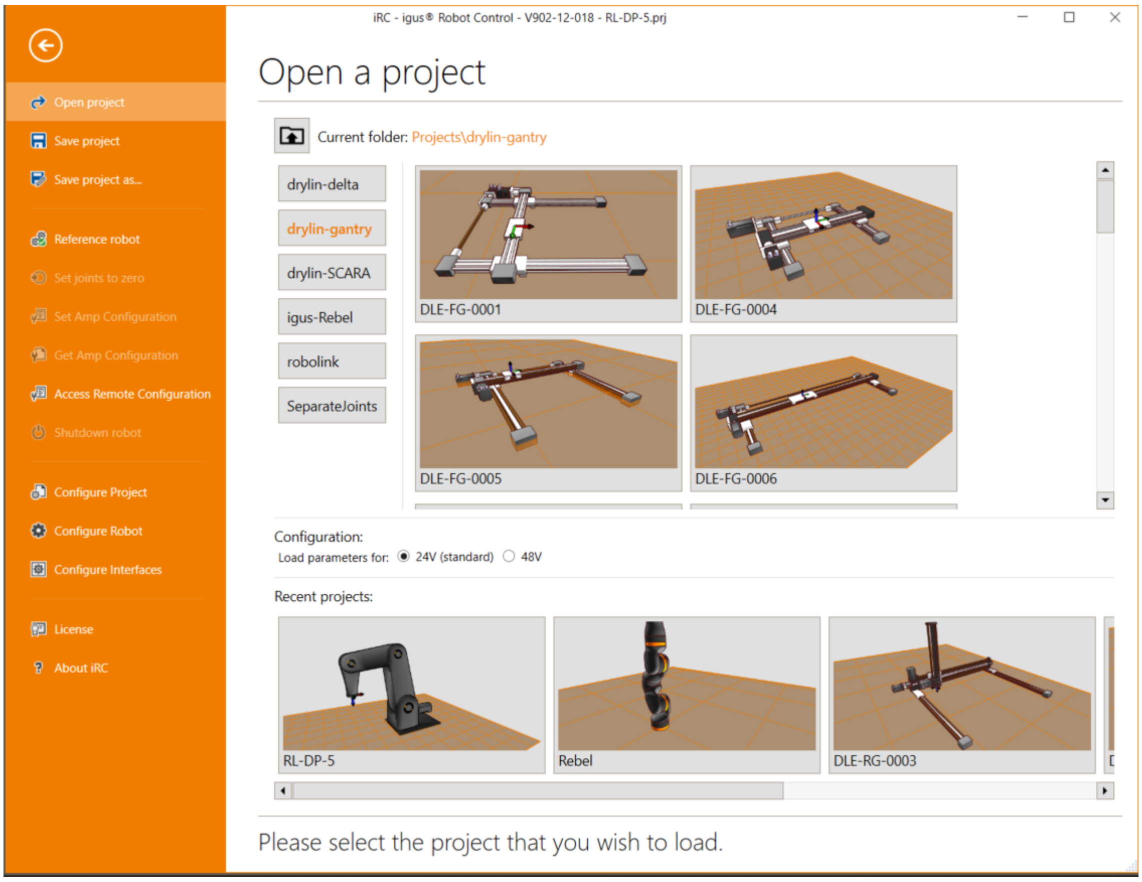
- "Programs & Variables": displays the current status of program variables.

The "Help" icon in the lower right corner contains links to the wiki pages ("Online Documentation", "Software Updates", "Examples", "Troubleshooting") and a link to "Contact Support". The log files of iRC and the integrated controller can also be accessed here.

9.1.1 Selecting the type of robot

iRC - igus Robot Control provides project-related settings for different robot types, such as gantries, robot arms or delta robots. Figure **??** shows the project open area, which can be used to load the corresponding project.

1. Click on the "File" tab in the upper left corner and select "Open Project".
2. Select the robot group, e.g. "robolink".
3. Make sure that the parameter set matching the operating voltage of the robot is selected. This will affect the speed and acceleration.
4. Now click on the named image that matches your robot, e.g. RL-DP-5.



Open a project

Current folder: Projects\drylin-gantry

Configuration:
Load parameters for: [✓ 24V (standard)] 48V

Recent projects:

Please select the project that you wish to load.

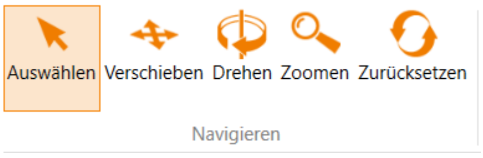
Figure 16: Selection of the robot type via the menu item "File" → "Open Project"

9.1.2 Navigation and movement of the robot in the 3D view

A 3-button mouse is recommended for navigating in the iRC - igus Robot Control 3D environment:

- Left button:
  - Selecting icons and functions in the menu.
  - Moving a robot axis: Place the cursor over a joint (it will be highlighted), then click and move the cursor up and down while holding down the left mouse button.

    middle button/mouse wheel:Navigate the scene to rotate the robot: Move the cursor while holding down the middle mouse button. Mouse wheel rotation: Zoom in/out to the current cursor position.
  - Right button: move the image section.



The function of the left mouse button can be changed in the "Scene" tab under "Navigate". The movement options "Select", "Move", "Rotate" and "Zoom" are available. "Reset" returns you to the home screen.

9.2 Connecting the robot

9.2.1 Hardware connection

The real robot can be controlled in the same way as the simulated one, only the hardware must first be connected and activated by clicking the "Connect", "Reset" and "Activate" icons in the "Physical Robot" button group in the "Motion" tab (see Fig. 17). After that, the robot must be referenced.

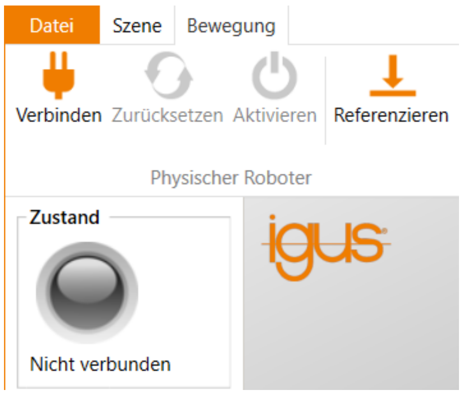


Figure 17: Buttons for connecting to the hardware, resetting errors and activating the motors, referencing and "Status" display.

1. "Connect": Establish the connection to the hardware.
   - This initializes the connection (mostly via Ethernet, in some cases via USB CAN adapter).
   - The "Status" indicator on the left side changes from gray to red.
   - Several error messages are displayed below the "Status" indicator.
2. "Reset": Resets the errors.
   - This key is used to reset the error memories of the electronic modules of the robot controller.
   - The axis positions are transferred from the real robot to the simulation environment. The 3D visualization of the robot should now correspond to the current position of the real robot.

     This must be checked with every error reset! If the values do not match, referencing must be performed, which is described in section 9.3.

   - The "Status" display remains red. The error messages are cleared, only "Motors not enabled" remains. If other error messages are displayed, try again and follow the instructions in the robot documentation.

3. "Activate": Activation of the motors.
   - The "Status" indicator is now green.

9.2.2 Move the robot

It is now possible to move the robot using the jog keys, a mouse in the user interface or a gamepad, see sections 9.1.2 and 9.4.

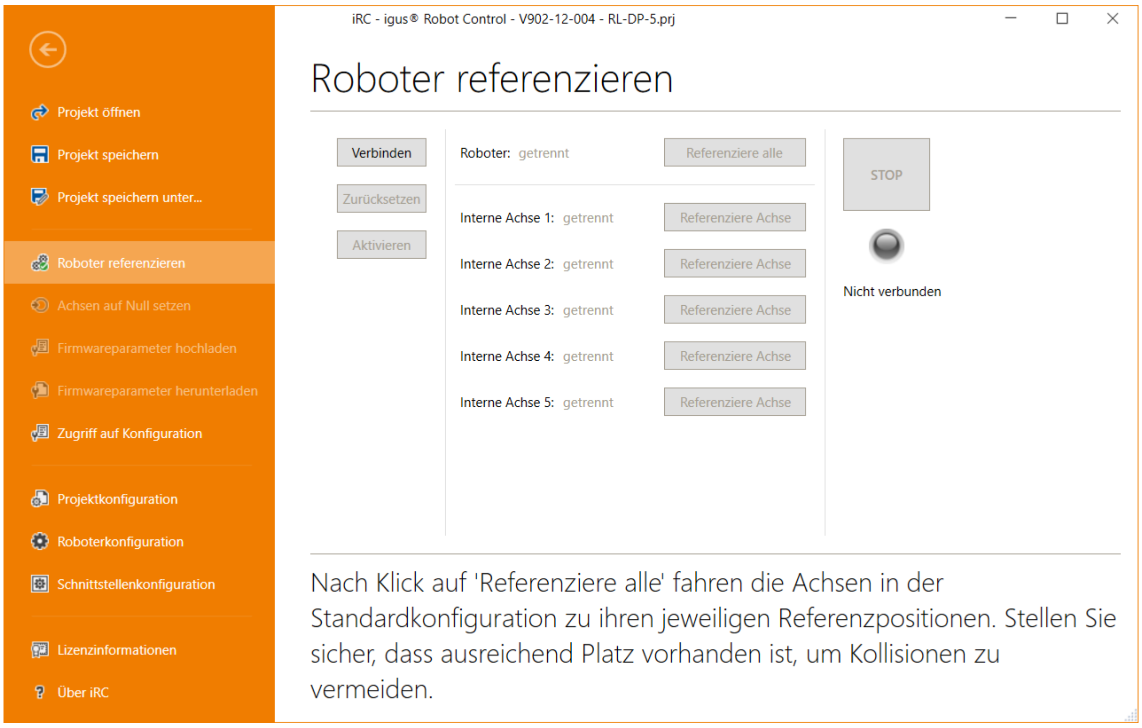
9.3 Referencing the robot

- After startup, the robot must be referenced. Before referencing, the robot's axes can only perform motions in joint mode. This is to avoid collisions during unreferenced robot operation.
- Cartesian movements or the start of a program are only possible after referencing.
- The status is displayed on the left side of the iRC - igus Robot Control.

The motor modules store the position in an EEPROM. However, due to gravity or other forces, the axes may move when the motor power is switched off. In this case, the motor modules no longer report the correct position to the software. In order to synchronize the position between software, stepper motor module and robot axis, referencing must be performed.

9.3.1 Step by step guide of referencing

1. Start the robot controller and the iRC - igus Robot Control.
2. Press the buttons "Connect", "Reset" and "Activate" (Fig. 17).
3. Click the "Reference" button (Fig. 17) in the button group "Physical Robot" in the "Motion" tab (or "Robot Reference" in the "File" tab) to open the referencing window.
4. Click on the "Reference Axis" buttons to start referencing the axes, see Fig. 18. Multiple joints can perform referencing in parallel.
5. You can also click on "Reference all", then the axes will start referencing in an order defined in the project file.
6. Once all movements have been executed and the robot is at rest again, click on "Reset" and "Activate". Now the robot is fully functional.



Roboter referenzieren

Roboter: getrennt

Interne Achse 1: getrennt

Interne Achse 2: getrennt

Interne Achse 3: getrennt

Interne Achse 4: getrennt

Interne Achse 5: getrennt

Nicht verbunden

Nach Klick auf 'Referenziere alle' fahren die Achsen in der Standardkonfiguration zu ihren jeweiligen Referenzpositionen. Stellen Sie sicher, dass ausreichend Platz vorhanden ist, um Kollisionen zu vermeiden.

Figure 18: Referencing of the axes.

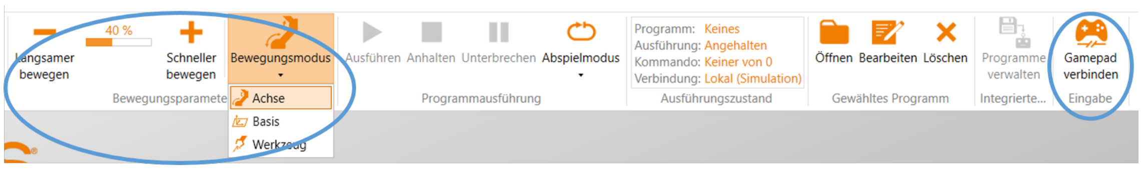
9.4 Moving the robot

The robot can be moved manually (or "jogged") when no program is running. The following options are available for this purpose:

- Software keys
- dragging joints in 3D area
- gamepad

The most important settings can be found in the "Movement" tab (see Fig. 19):

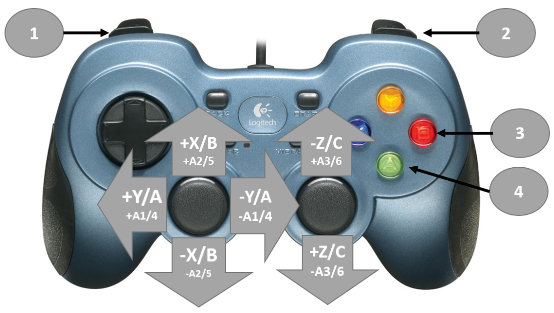
- "Input": Connecting a gamepad
- "Motion parameters": Selection of movement modes and speed



Programm: Keines
Ausführung: Angehalten
Kommando: Keiner von 0
Verbindung: Lokal (Simulation)

Figure 19: Control panels for moving the robot (highlighted in blue).

A gamepad is possibly the most intuitive way to move the robot. The fig. 20 shows the key assignment. Pressing "Connect Gamepad" will connect iRC to a gamepad. If the connection is successful, an OK sign will be displayed under the icon. The device must be of type "Joystick" or "Gamepad". For more information about the connection, see the "LogMessages" tab (at the bottom of the window area).



1. Change movement type
2. Change key assignment:

   Switch between X, Y, Z and A, B, C in Cartesian motion mode or A1, 2, 3 and A4, 5, 6 in Axis motion mode.
3. Add Joint program instruction
4. Add Linear program instruction

Figure 20: Key assignment of the gamepad

Software buttons allow the selection of the motion mode. Three modes are available, each of which allows the movement speed to be changed between 0 and 100% (Fig. 19):

- "Axis": Clicking on A1 to A6 moves the corresponding robot axis (if present). E1 - E3 moves the external joints. This could be a linear or a rotational axis (21).
- "Base" (Cartesian mode) moves the robot in straight lines along the X, Y, and Z axes of the base coordinate system. {Tool": (Cartesian mode) moves the robot in X, Y and Z of the current tool coordinate system.

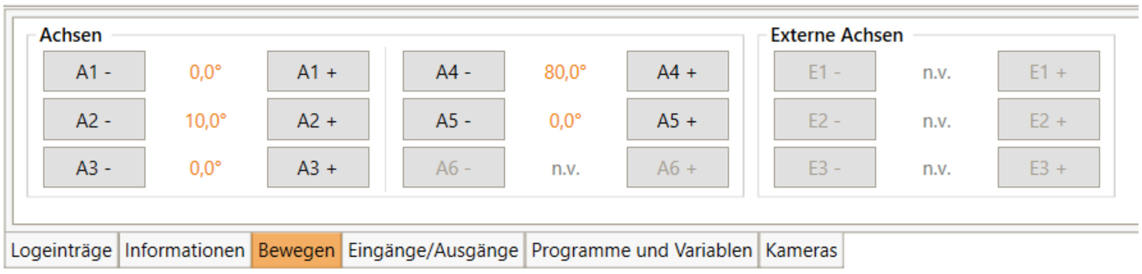
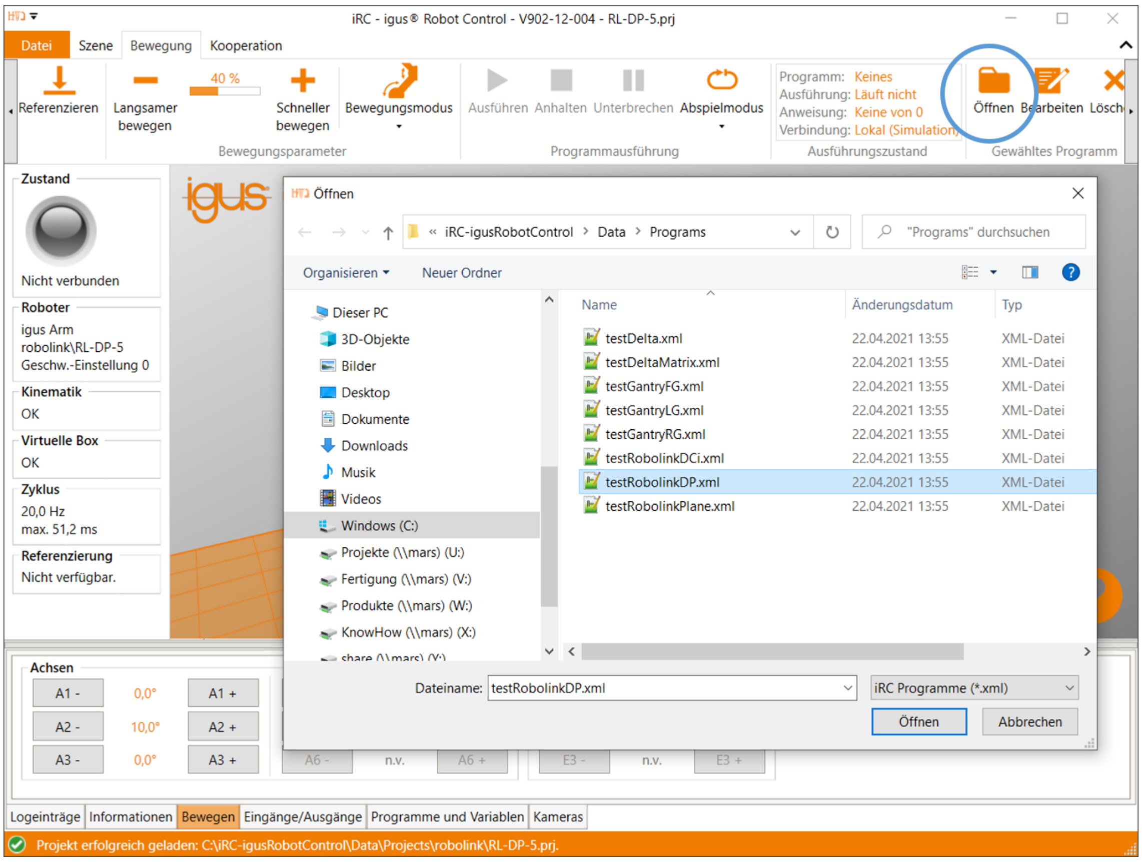


Figure 21: The software buttons for "axis" movements. In both Cartesian modes, the buttons change to X, Y, Z, A, B and C.

9.5 Starting robot programs

A robot program can be loaded and started as follows:

1. To load the program, click on the "Open" folder icon in the "Selected program" group of the "Movement" tab and select a program, e.g. "testRobolink.xml".



Programm: Keines
Ausführung: Läuft nicht
Anweisung: Keine von 0
Verbindung: Lokal (Simulation)

igus Arm
robolink\RL-DP-5
Geschw.-Einstellung 0

Kinematik: OK
Virtuelle Box: OK
Zyklus: 20,0 Hz, max. 51,2 ms
Referenzierung: Nicht verfügbar.

« iRC-igusRobotControl › Data › Programs

Name	Änderungsdatum	Typ
testDelta.xml	22.04.2021 13:55	XML-Datei
testDeltaMatrix.xml	22.04.2021 13:55	XML-Datei
testGantryFG.xml	22.04.2021 13:55	XML-Datei
testGantryLG.xml	22.04.2021 13:55	XML-Datei
testGantryRG.xml	22.04.2021 13:55	XML-Datei
testRobolinkDCi.xml	22.04.2021 13:55	XML-Datei
testRobolinkDP.xml	22.04.2021 13:55	XML-Datei
testRobolinkPlane.xml	22.04.2021 13:55	XML-Datei

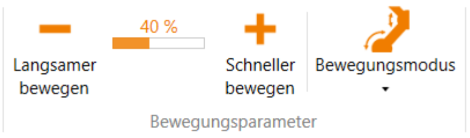
Dateiname: testRobolinkDP.xml
iRC Programme (*.xml)

A1 -	0,0°	A1 +			
A2 -	10,0°	A2 +			
A3 -	0,0°	A3 +	A6 - n.v. A6 +		E3 - n.v. E3 +

Projekt erfolgreich geladen: C:\iRC-igusRobotControl\Data\Projects\robolink\RL-DP-5.prj.

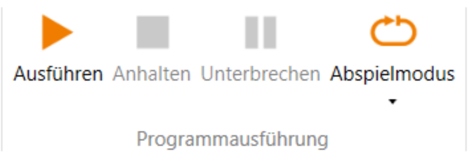
Figure 22: Loading a program (highlighted in blue).

2. Set the basic speed:



- Before starting an untested program, set the speed to e.g. 20%.
- Be especially attentive during the first complete program run and have the emergency stop key ready.

3. Start the program:



the Execute icon in the Program Execution button group of the Motion tab.

4. Pause or interrupt the program:
   - After pressing the "Pause" icon, the robot can continue with the program by clicking the "Run" icon again.

- After pressing the "Stop" icon, the program will start with the first command when the "Run" icon is clicked again.
- The "Play" mode can be set to three different values:
  - Once (the program stops after a single cycle).
  - Repeat (the program stops only by "Stop" or "Interrupt").
  - Single step (this is useful for debugging a program).

9.6 Digital inputs and outputs

The simplest way to control external devices, communicate with programmable logic controllers or enable human operation via buttons and lights is via digital inputs and outputs. Each robot controller is supplied with a DIO module. This provides 7 inputs and 7 outputs. If additional inputs and outputs are required, up to two additional DIO modules can be integrated. In total, up to 3 DIO modules can be controlled.
The status of the inputs and outputs can be monitored under "Inputs/Outputs". Both inputs and outputs can be manually activated or deactivated:

- Outputs can be set manually when no program is running.
- Inputs can only be set in simulation when no robot is connected. So you can test the reaction of programs to different inputs even without the corresponding hardware.

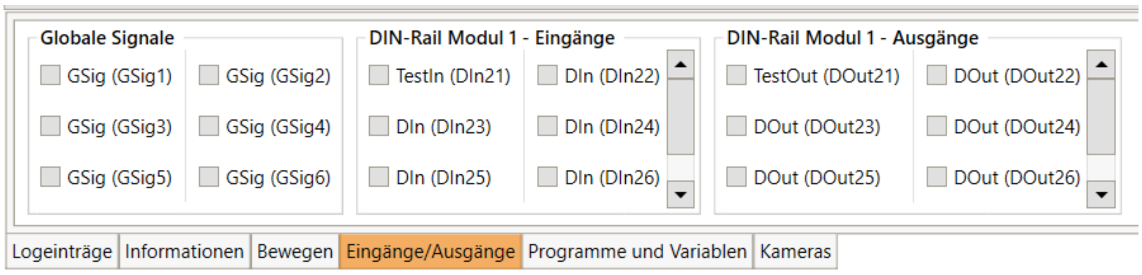


Figure 23: Input/output area of the iRC - igus Robot Control.

The configuration of the inputs and outputs is described in section **??**.

9.7 Software interfaces

The robot controller provides various interfaces:

- PLC interface for controlling the robot via a PLC, especially for starting and stopping programs.
- plug-in interface for integrating cameras, for example. The plug-in then transmits the target positions to the robot controller.
- Container Runtime Interface (CRI) to enable further interaction. This interface can be used, for example, to generate workpiece-specific programs from a database.
- ROS interface to operate the robot via the Robot Operating System (`www.ros.org`).

See section 11.3 for the configuration of these interfaces.

9.8 Update the software

Updates of the iRCsoftware can be found at the following address: `https://wiki.cpr-robots.com/index.php/IgusRobotControl-EN`.

Create a backup because files can be overwritten during the update!
Rename your old iRC-Folder (z.B.) before starting the installation. This way you can go back to the old version.

The following may need to be copied from the previous installation:

- The created robot programs
- Changes in the project or in the robot configurations.

10 Programming a Robot with iRC

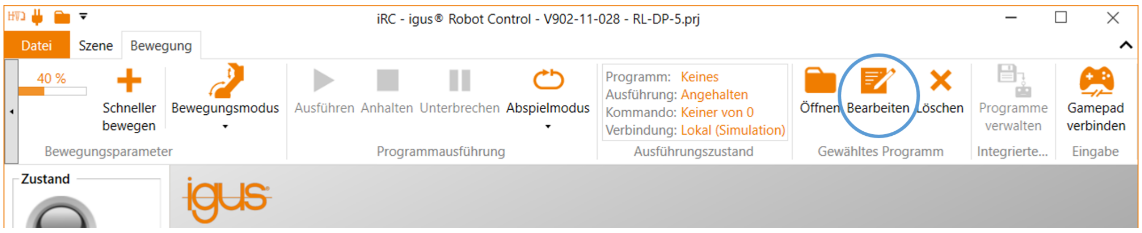
The iRC - igus Robot Control enables the creation of robot programs. The method of programming is called "teach-in programming", which works as follows:

1. Move the robot manually to the position you want to record.
2. Record the position and define how this position should be reached (linear/joint movement).
3. Repeat these steps and add digital output commands or program flow commands in between if needed.

The integrated editor is provided for creating and editing these programs.

10.1 Program Editor

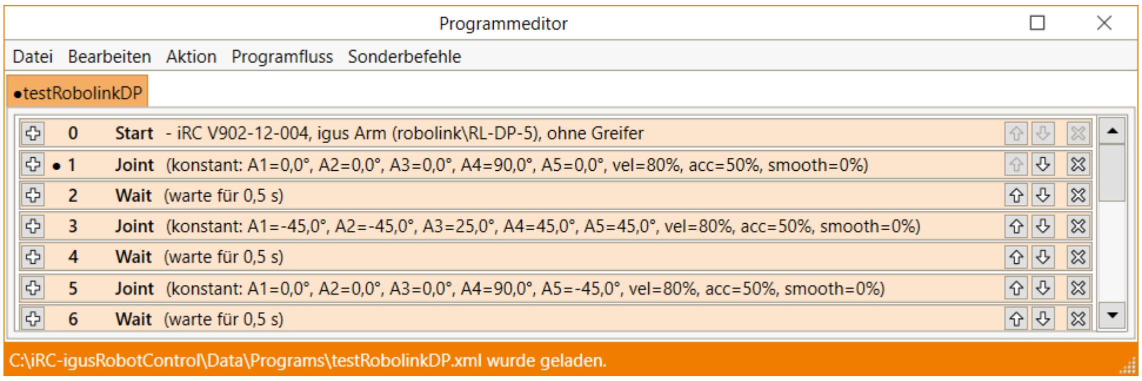
Each command of the robot program consists of one line, e.g. "Joint" or "Wait". Only commands that control the program flow are divided into several lines, e.g. "Loop" and "EndLoop". The program editor is opened with the "Edit" button in the "Motion" tab ofiRC.



Programm: Keines
Ausführung: Angehalten
Kommando: Keiner von 0
Verbindung: Lokal (Simulation)

Figure 24: The program editor is opened by clicking "Edit".

The following window opens, here with a short program:



```
0    Start  - iRC V902-12-004, igus Arm (robolink\RL-DP-5), ohne Greifer
• 1  Joint  (konstant: A1=0,0°, A2=0,0°, A3=0,0°, A4=90,0°, A5=0,0°, vel=80%, acc=50%, smooth=0%)
2    Wait   (warte für 0,5 s)
3    Joint  (konstant: A1=-45,0°, A2=-45,0°, A3=25,0°, A4=45,0°, A5=45,0°, vel=80%, acc=50%, smooth=0%)
4    Wait   (warte für 0,5 s)
5    Joint  (konstant: A1=0,0°, A2=0,0°, A3=0,0°, A4=90,0°, A5=-45,0°, vel=80%, acc=50%, smooth=0%)
6    Wait   (warte für 0,5 s)
```

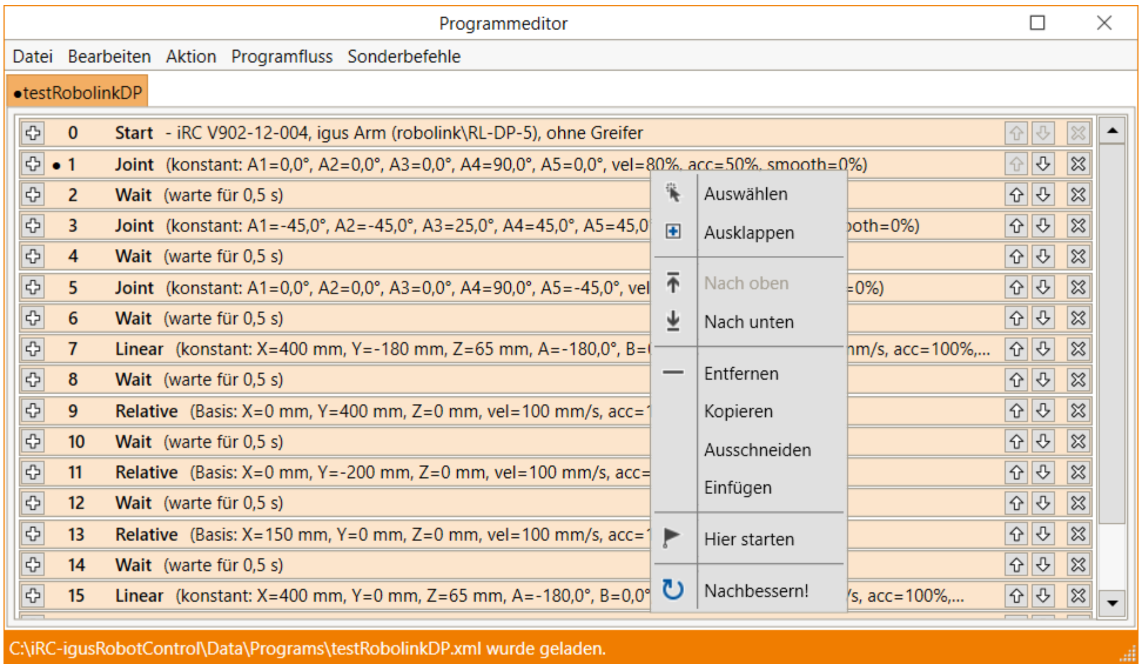
C:\iRC-igusRobotControl\Data\Programs\testRobolinkDP.xml wurde geladen.

Figure 25: Program editor with a shord programm

The following sections show how to create a program with the program editor.

10.1.1 Changing the Command Sequence

To move a command, use the arrows on the right side of the command line. Alternatively, you can click "Down" or "Up" from the command context menu (see Fig. 26).



```
0   Start  - iRC V902-12-004, igus Arm (robolink\RL-DP-5), ohne Greifer
• 1 Joint  (konstant: A1=0,0°, A2=0,0°, A3=0,0°, A4=90,0°, A5=0,0°, vel=80%, acc=50%, smooth=0%)
2   Wait  (warte für 0,5 s)
3   Joint  (konstant: A1=-45,0°, A2=-45,0°, A3=25,0°, A4=45,0°, A5=45,0 ...oth=0%)
4   Wait  (warte für 0,5 s)
5   Joint  (konstant: A1=0,0°, A2=0,0°, A3=0,0°, A4=90,0°, A5=-45,0°, vel ...=0%)
6   Wait  (warte für 0,5 s)
7   Linear  (konstant: X=400 mm, Y=-180 mm, Z=65 mm, A=-180,0°, B= ...nm/s, acc=100%,...
8   Wait  (warte für 0,5 s)
9   Relative  (Basis: X=0 mm, Y=400 mm, Z=0 mm, vel=100 mm/s, acc=
10  Wait  (warte für 0,5 s)
11  Relative  (Basis: X=0 mm, Y=-200 mm, Z=0 mm, vel=100 mm/s, acc=
12  Wait  (warte für 0,5 s)
13  Relative  (Basis: X=150 mm, Y=0 mm, Z=0 mm, vel=100 mm/s, acc=
14  Wait  (warte für 0,5 s)
15  Linear  (konstant: X=400 mm, Y=0 mm, Z=65 mm, A=-180,0°, B=0,0° .../s, acc=100%,...
C:\iRC-igusRobotControl\Data\Programs\testRobolinkDP.xml wurde geladen.
```

Figure 26: The context menu of a command

The program editor prevents invalid commands that would break the structure of the program. If moving a command up or down is not possible, the corresponding buttons and menu items are grayed out.

10.1.2 Touching Up Positions

Certain commands require position values as parameters. It is often desirable to use the current position of the robot. Entering it by hand can take some time and is prone to errors. For such cases you can use the command "Touch Up":

- Select the command and click "Touch Up!" from the "Edit" menu.
- Select the command and then press Ctrl+T.
- Open the context menu by right-clicking on the command and click on "Touch Up!" (see Fig. 26).

The program editor then replaces the position values in the command with the current position of the robot.

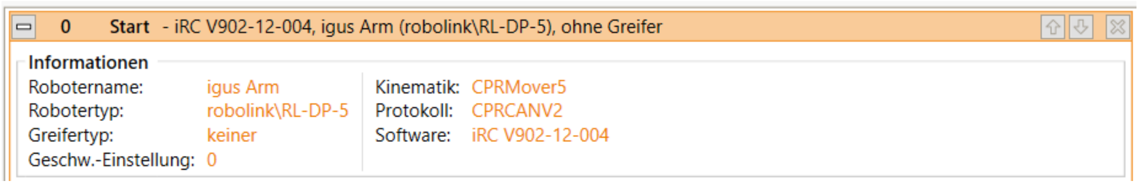
10.1.3 Set Start Command

It is possible to execute programs command by command or to select a specific command as the starting point of a program for test purposes. The command that will be executed first the next time the program is started, or - if the program is currently running - the currently executed command is marked by a dot in the program editor. The subprogram containing this command is also marked by a dot in front of the program name.
To select a specific command as the starting point for execution, click on "Start Here" in the context menu (see Fig. 26).

10.2 Comments and Information within the Program

10.2.1 Information about the Program

The program editor inserts the pseudo command Start at the beginning of each program. It does not represent a real command, but displays information about the current hardware, software and kinematics. It is not possible to move or remove it.
When loading a program, this information is compared to avoid executing an incompatible program.



0 **Start** - iRC V902-12-004, igus Arm (robolink\RL-DP-5), ohne Greifer

Informationen

Robotername:	igus Arm	Kinematik:	CPRMover5
Robotertyp:	robolink\RL-DP-5	Protokoll:	CPRCANV2
Greifertyp:	keiner	Software:	iRC V902-12-004
Geschw.-Einstellung:	0		

Figure 27: The Start line contains information about the current hardware.

10.2.2 Descriptions

Each command of a program contains a description. It should be used to describe to other users what the command is for.

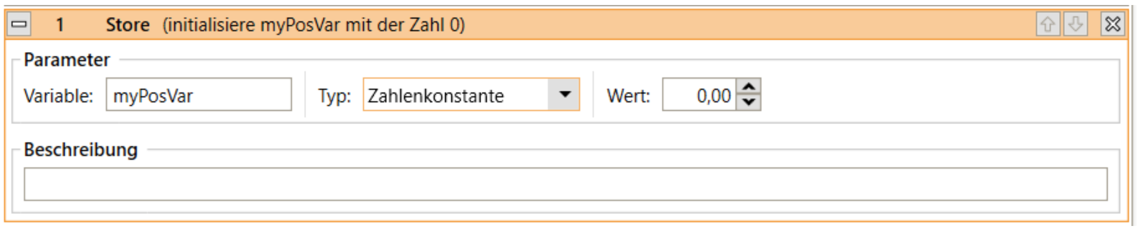
10.2.3 Comments

The Comment command can be used to insert plain descriptions into programs. It has no effect to the robot during execution.
It can be found in the program editor in the menu item "Special" → "Comment".

10.3 Variables and Variable Access

Programs for iRC and TinyCtrl support two types of variables:

- Number variables: These can be used to store integer or floating point numbers.
- position variables: These can be used to store cartesian and joint positions. Whether such a variable is interpreted as cartesian or joint depends on the context.


1 Store (initialisiere myPosVar mit der Zahl 0)

Parameter

Variable: myPosVar Typ: Zahlenkonstante Wert: 0,00

Beschreibung

Figure 28: Definition of a number variable.

The cartesian components x, y, z are in mm, the euler angles A, B, C are in degrees. The joint values are measured in mm or degrees depending on the type of axis.

10.3.1 User-defined Variables

It is possible to define variables with the Store command, which is accessible in the program editor through the menu items under "Special" → "Variable definition".
Three types of store operations can be selected:

- "Current position":
  A position variable is initialized with the cartesian and axis position of the robot when the command is executed.
- "NumberConstant":
  A number variable is initialized with the constant specified in "value" (see 28).
- "PositionConstant":
  A position variable is initialized with the constants specified in "Cartesian Position", "Joint Position" and "External Joints" (see fig. 29). Depending on the kinematic model of the current robot, certain axes may not be available.

The name of the variable can be set under "Variable". If a variable with the same name is already defined, its value and type will be overwritten. All variables are global, i.e. they are also accessible from subroutines.

10.3.2 System Variables

The following predefined variables are available without having to define them:

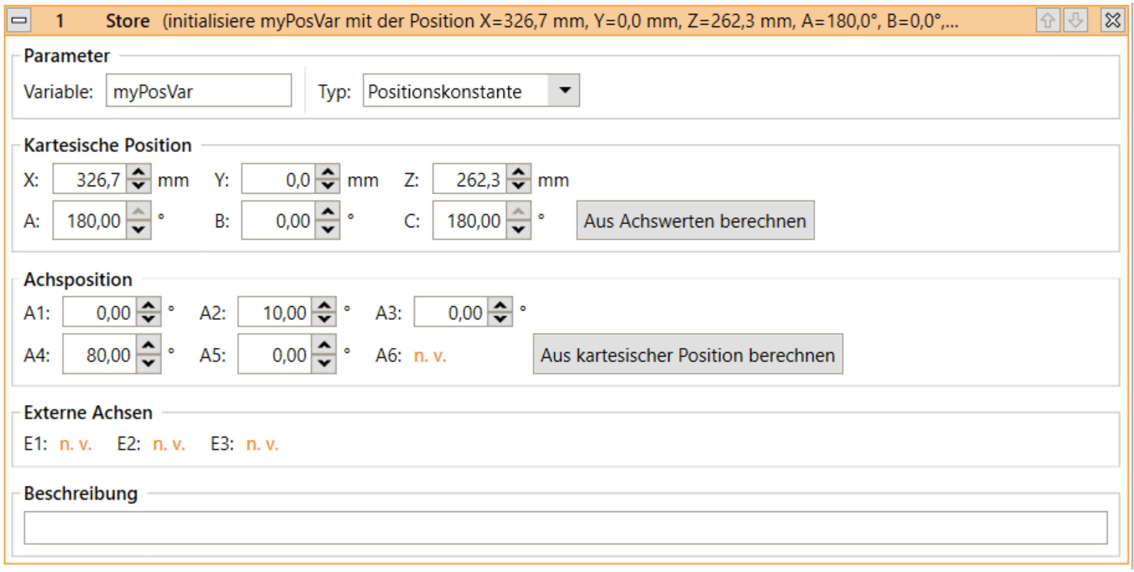
- #position: The current position of the robot.
- #programrunning: 1 if the robot program is running, otherwise 0.
- #logicprogramrunning: 1 if the logic program is running, 0 otherwise.

Note that the system controls the values of predefined variables - they cannot be changed by the program. The names of predefined variables always start with "#"

10.3.3 Accessing Elements

Position variables contain the following elements:

- Position: x, y, z
- Orientation: a, b, c



1 Store (initialisiere myPosVar mit der Position X=326,7 mm, Y=0,0 mm, Z=262,3 mm, A=180,0°, B=0,0°,...

Parameter

Variable: myPosVar Typ: Positionskonstante

Kartesische Position

X: 326,7 mm Y: 0,0 mm Z: 262,3 mm

A: 180,00 ° B: 0,00 ° C: 180,00 °

Aus Achswerten berechnen

Achsposition

A1: 0,00 ° A2: 10,00 ° A3: 0,00 °

A4: 80,00 ° A5: 0,00 ° A6: n. v.

Aus kartesischer Position berechnen

Externe Achsen

E1: n. v. E2: n. v. E3: n. v.

Beschreibung

Figure 29: Definition of a position variable.

- Joint positions: a1, a2, a3, a4, a5, a6, e1, e2, e3

The elements are accessed by appending them with a dot, e.g. "myvariable.x" or "myvariable.a3".

10.3.4 Calculating with Variables

Calculations with variables can be performed using the Math command, which is accessible in the program editor via the menu item "Special" → "Variable operation".
"First operand" defines the first operand of the operation to be executed. It is also used to store the result.
"Second operand" defines the second operand of the operation. It can contain numeric constants, names of number variables or components of position variables.
The following operations are supported and can be selected under "Operation":

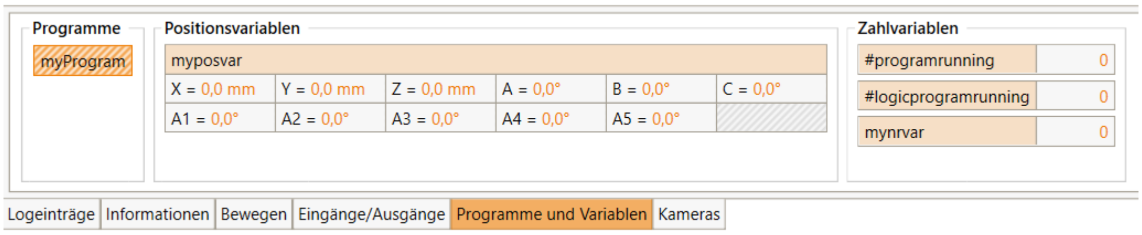
- Assignment: The first operand is set to the value of the second operand.
- addition: The first operand increased by the value of the second operand.
- subtraction: The first operand is decreased by the value of the second operand.
- Multiplication: The first operand multiplied by the value of the second operand.
- Division: The first operand divided by the value of the second operand.
- Modulus: The remainder of the division of the first operand by the second operand is stored in the first operand.

The following combinations of operands and operators are allowed (number here also means position components):

	Assignment	Plus	Minus	Mult.	Division	Modulus
Both are number	x	x	x	x	x	x
Both are position	x	x	x			
Op 1 is position, Op 2 is number				x	x	x
Op 1 is Number, Op 2 is position						

10.3.5 Observing Variables

You can observe the current values of all defined variables in iRC in the "Programs and Variables" tab in the status area.



Programme

myProgram

Positionsvariablen

myposvar					
X = 0,0 mm	Y = 0,0 mm	Z = 0,0 mm	A = 0,0°	B = 0,0°	C = 0,0°
A1 = 0,0°	A2 = 0,0°	A3 = 0,0°	A4 = 0,0°	A5 = 0,0°	

Zahlvariablen

#programrunning	0
#logicprogramrunning	0
mynrvar	0

Figure 30: The values of the variables are displayed in the info area.

10.4 Program Flow

10.4.1 Conditions

Conditions can be used in if-then-else statements, loops, and as abort conditions in motion statements. Conditions can be combinations of digital inputs, global signals, Boolean operations, and comparisons. Examples:

DIn23	True if digital input 23 is set.
DIn23 AND !DIn27	True if digital input 23 is set and 27 is not set.
modelclass = 31	True if the variable "modelclass" is 31
mempos.x > 350.0	True if the x component of the position variable "mempos" is greater than 350.

The syntax of conditional statements is defined by the following EBNF definition:

Expression	: = ["!"] <Boolean> <BooleanOperator> <Boolean> ...
Boolean	: = <BooleanConstant> | <Expression> | "(" <Expression> ")" | CompExpression | "(" <CompExpression> ")" | <DigitalInputs> | "(" <DigitalInputs> ")"

BooleanOperator	: = "And" | "Or"
BooleanConstant	: = "True" | "False"
Digital Inputs	: = <ChannelType> <ChannelId>
ChannelType	: = "Din" | "GSig"
ChannelId	: = Integer value
CompExpression	: = <CompValue> <CompOperator> <CompValue>
CompValue	: = <Variable> | <Number>
Variable	: = <Numbervariable> | < PositionComponent>
Numbervariable	: = Name of a number variable
Positions component	: = <Position variable> "." <Component>
PositionVariable	: = Name of a position variable
Component	: = "x" | "y" | "z" | "A" | "B" | "C" | "A1" | "A2" | "A3" | "A4" | "A5" | "A6" | "E1" | "E2" | "E3"
Number	: = Integer or floating point number
CompOperator	: = "=" | ">" | "<" | ">=" | "<="

10.4.2 Stop

The command "Stop" stopps the program execution.
It is available through the menu item "Flow" → "Stop".

10.4.3 Pause

The Pause command pauses the execution of the program. The execution can be resumed later by the user.

10.4.4 Wait

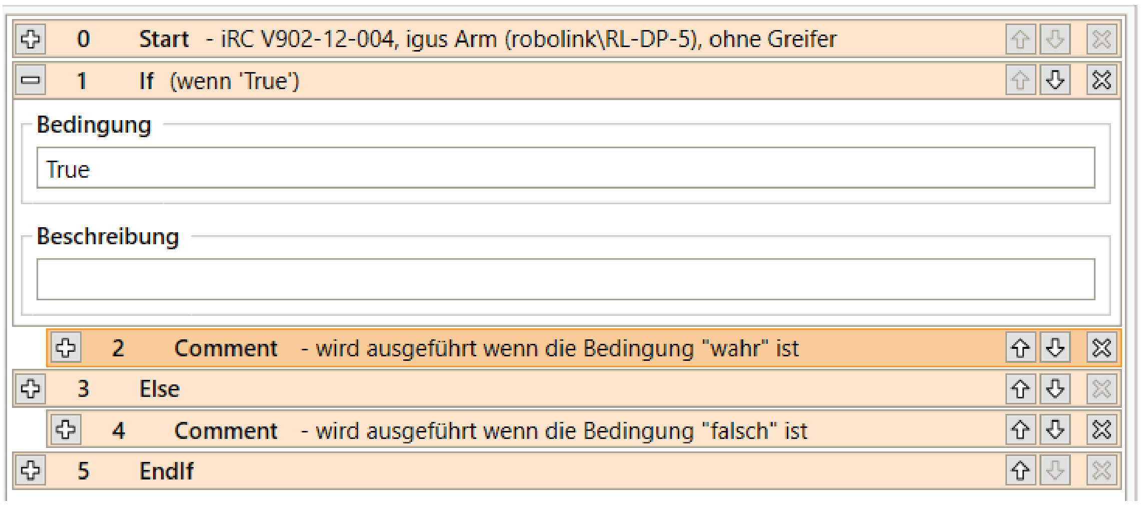
The Wait command instructs the robot to wait until a specified amount of time has passed or a condition is met. It is accessible via the menu items under "Program Flow" → "Wait" in the program editor of the iRC.
The different modes can be selected under "Type":

- "Timeout": The time specified in "Timeout" will be waited.
- "Condition": Waits until the condition specified in "Expression" evaluates to "true".

10.4.5 If-then-else

The If command branches the execution of the program depending on the value of a conditional expression. It is accessible through the "Flow" → "If...then...else" menu item in the iRC program editor.

The specified condition must conform to the syntax described in section **??**. The statements between "If" and "Else" will be executed if the condition evaluates to true. Otherwise the statements between "Else" and "EndIf" will be executed.



0 Start - iRC V902-12-004, igus Arm (robolink\RL-DP-5), ohne Greifer
1 If (wenn 'True')
Bedingung: True
Beschreibung:
2 Comment - wird ausgeführt wenn die Bedingung "wahr" ist
3 Else
4 Comment - wird ausgeführt wenn die Bedingung "falsch" ist
5 EndIf

Figure 31: The If statement branches the program flow.

10.4.6 Loops

The Loop command allows the definition of execution loops. Under "Type" you can choose between the following loop types:

- "condition": The loop is repeated until the specified condition evaluates to "true". It must conform to the syntax described in section **??**.
- "counter": The loop will repeat the number of times specified in "repeats".

The loop command is accessible through the menu items "Flow" → "Loop".

10.4.7 Matrix Loops

The Matrix command is designed to execute loops that allow the robot to perform grid movements, for example for palletizing tasks. It can be called from the menu item "Program Flow" → "Loop" → "Raster".
The following figure shows a motion pattern that can be executed by using the Matrix command.

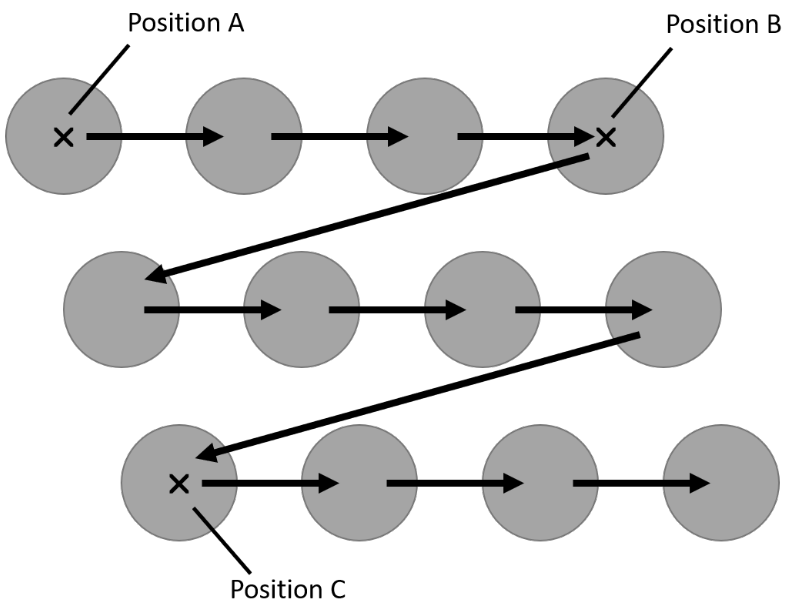
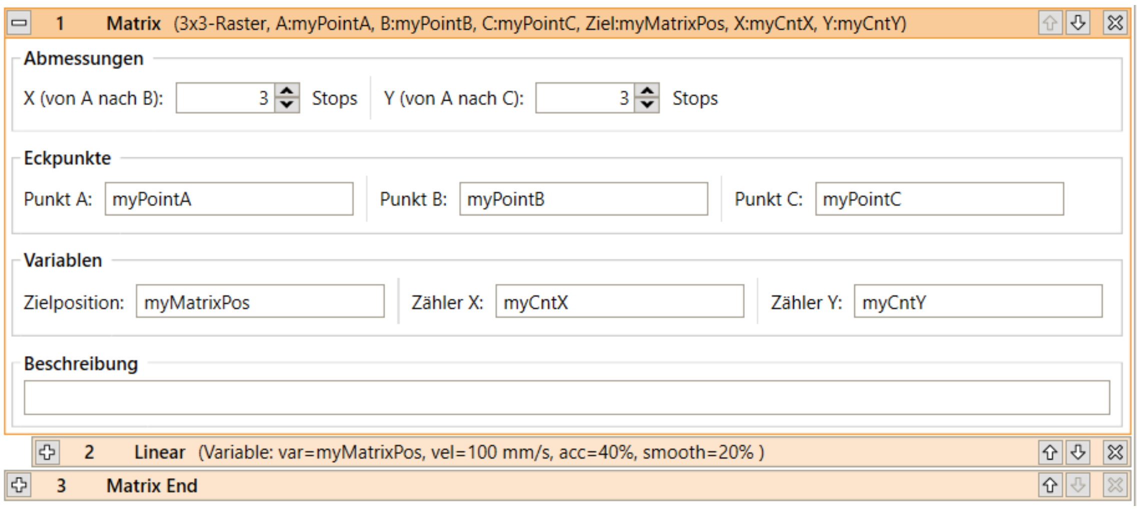


Figure 32: The matrix movement is from point A to B, then offset in direction C

.

The position variables specified as "Point A", "Point B" and "Point C" define the corners of the area covered by the matrix loop (see figure above). The number of steps to be taken is indicated by "counter X" (from A to B) and "counter Y" (from A to C). For example, in the figure above, X=4 and Y=3.


1 **Matrix** (3x3-Raster, A:myPointA, B:myPointB, C:myPointC, Ziel:myMatrixPos, X:myCntX, Y:myCntY)

Abmessungen
X (von A nach B): 3 Stops Y (von A nach C): 3 Stops

Eckpunkte
Punkt A: myPointA Punkt B: myPointB Punkt C: myPointC

Variablen
Zielposition: myMatrixPos Zähler X: myCntX Zähler Y: myCntY

Beschreibung

2 **Linear** (Variable: var=myMatrixPos, vel=100 mm/s, acc=40%, smooth=20%)

3 **Matrix End**

Figure 33: Definition of a matrix motion.

The block between "Matrix" and "Matrix End" is executed for each step. The position variable "TargetPosition" contains the position of the current target point for the respective step. Row and column of the current step are stored in the number variables given to "Counter X" and "Counter Y" respectively

10.4.8 Subprograms

Subprograms can be executed using the Sub command.
The path to the subroutine file is specified under "Filename". It is relative to the subfolder "Programs" of the iRC folder "Data". The command can be invoked from the menu item "Programflow" → "Subprogram".

10.5 Motion

10.5.1 Abort Conditions

Each motion command can be provided with a abort condition. It is a conditional expression that follows the syntax described in section **??**. During the execution of the motion command, the instruction is continuously evaluated, and the moment it evaluates to "true", the robot stops moving. It can be specified respectively under "abort condition" for each motion command.

10.5.2 Acceleration and Smoothing

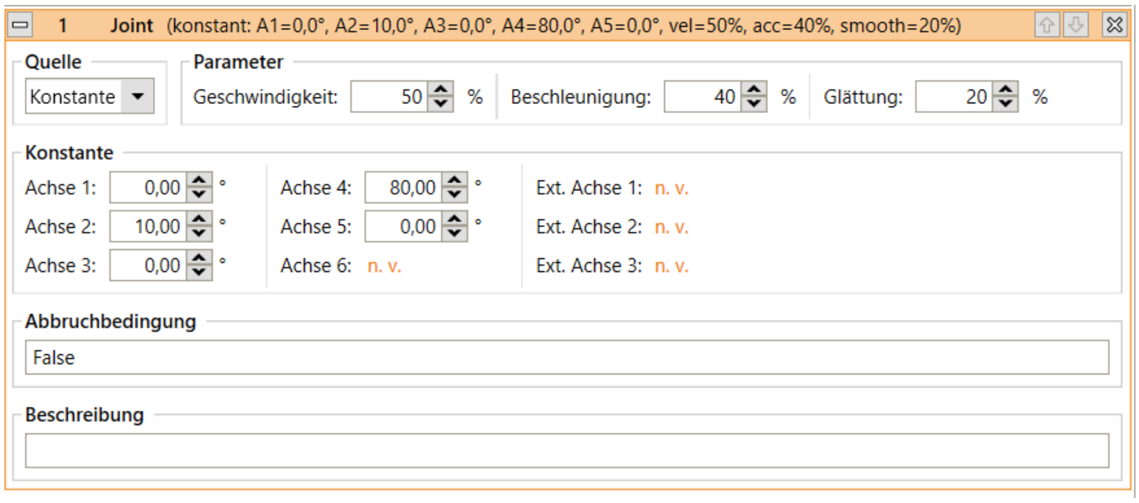
Axis accelerations can be defined in motion commands to prevent abrupt movements of the robot. This can be done for a single motion command, but also for a combination of up to a maximum of 10 commands. It is also possible to smooth the corners within the path defined by blocks of consecutive motion commands. This allows the robot to move through the corners without coming to a stop. When corner smoothing is set to 0%, the robot stops at each corner for a short moment. This results in a less fluid motion.
If motion commands are separated by other commands, the motion block is split and smoothing is interrupted.
Parameters:

- "Acceleration": Percentage of the maximum possible acceleration.
- "Smoothing": Amount of smoothing in percent.

10.5.3 Joint Motion

The Joint command moves the robot to an (absolute) target position specified in axis coordinates. The resulting movement of the TCP is usually a curve and not a straight line. The target position can be specified in the following way (select the appropriate "source"):

- "Constant": The target position is a constant value for each axis.


1 Joint (konstant: A1=0,0°, A2=10,0°, A3=0,0°, A4=80,0°, A5=0,0°, vel=50%, acc=40%, smooth=20%)

Quelle: Konstante

Parameter: Geschwindigkeit: 50 % Beschleunigung: 40 % Glättung: 20 %

Konstante

Achse 1: 0,00 ° Achse 4: 80,00 ° Ext. Achse 1: n. v.

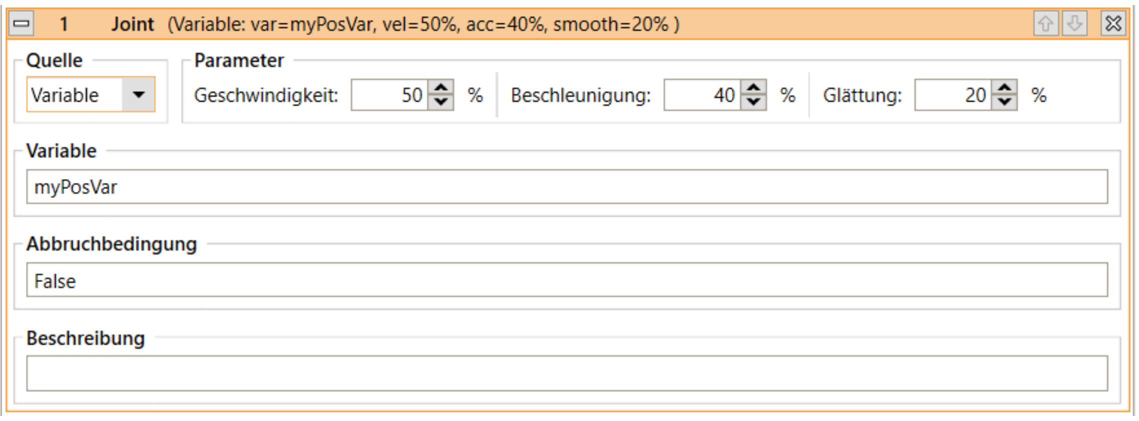
Achse 2: 10,00 ° Achse 5: 0,00 ° Ext. Achse 2: n. v.

Achse 3: 0,00 ° Achse 6: n. v. Ext. Achse 3: n. v.

Abbruchbedingung: False

Beschreibung

- "Variable": The target position is taken from the position variable specified in "Variable".


1 Joint (Variable: var=myPosVar, vel=50%, acc=40%, smooth=20%)

Quelle: Variable

Parameter: Geschwindigkeit: 50 % Beschleunigung: 40 % Glättung: 20 %

Variable: myPosVar

Abbruchbedingung: False

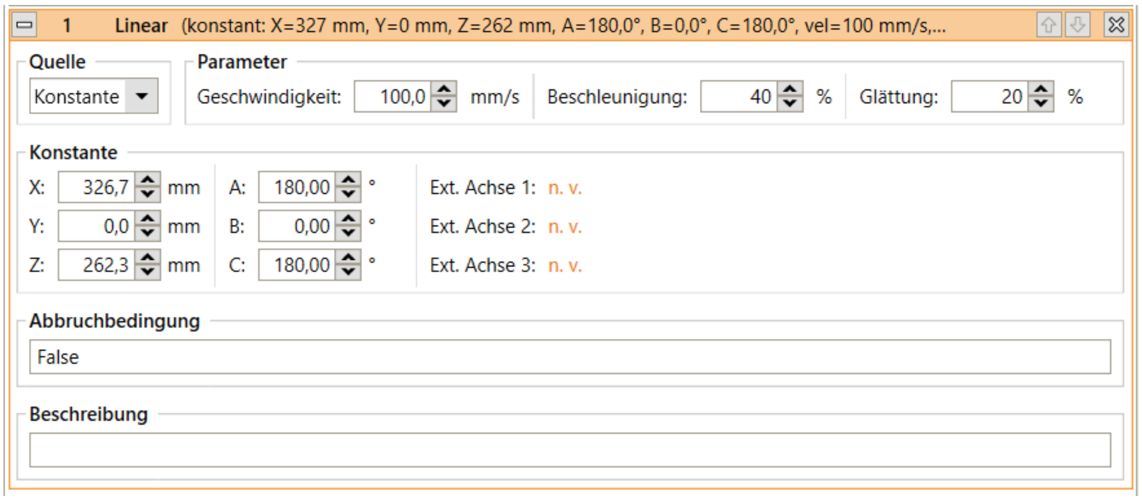
Beschreibung

The movement speed is indicated by "speed". It is measured in percent of the maximum allowed motion speed for the respective robot axes.
The joint command can be called in the program editor under the menu items "Action" → "AxisMotion" and "Action" → "VariableMotion" → "AxisMotion".

10.5.4 Linear Motion

The Linear command moves the robot to an (absolute) target position specified in cartesian coordinates. The resulting movement of the TCP follows a straight line. The target position can be specified as follows (select the corresponding "source"):

- "Constant": The target position is a constant given by Cartesian coordinates x, y, z and Euler angles A, B, C as well as the positions of the external axes if supported by the current robot kinematics.


1 Linear (konstant: X=327 mm, Y=0 mm, Z=262 mm, A=180,0°, B=0,0°, C=180,0°, vel=100 mm/s,...

Quelle: Konstante

Parameter: Geschwindigkeit: 100,0 mm/s Beschleunigung: 40 % Glättung: 20 %

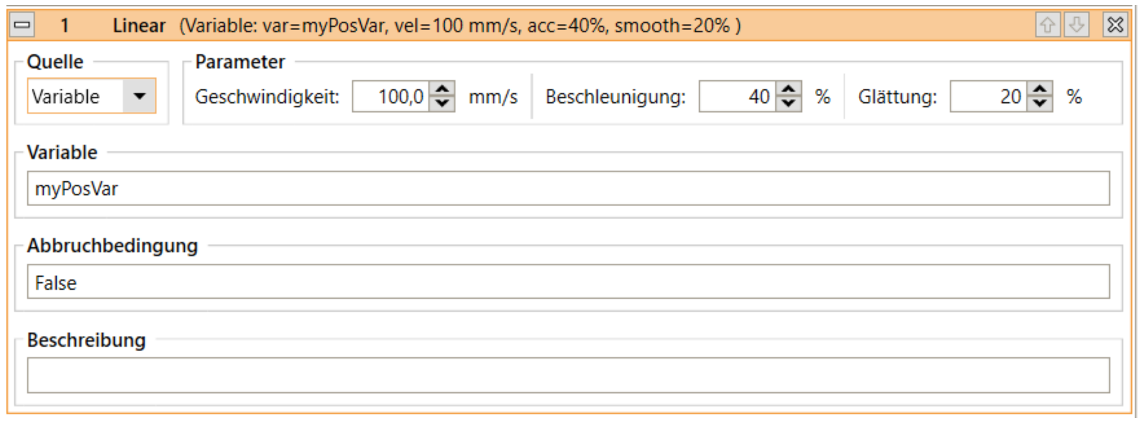
Konstante

X: 326,7 mm A: 180,00 ° Ext. Achse 1: n. v.
Y: 0,0 mm B: 0,00 ° Ext. Achse 2: n. v.
Z: 262,3 mm C: 180,00 ° Ext. Achse 3: n. v.

Abbruchbedingung: False

Beschreibung

- "Variable": The target position is taken from the position variable specified in "Variable".


1 Linear (Variable: var=myPosVar, vel=100 mm/s, acc=40%, smooth=20%)

Quelle: Variable

Parameter: Geschwindigkeit: 100,0 mm/s Beschleunigung: 40 % Glättung: 20 %

Variable: myPosVar

Abbruchbedingung: False

Beschreibung

The movement speed is specified by "Speed" in mm/s. If it exceeds the maximum allowed movement speed of the robot, it will cause a kinematic error during execution. The Linear command can be called in the program editor under "Action" → "Linear Motion" and "Action" → "Variable Motion" → "Linear Motion".

10.5.5 Relative Motion

The Relative command allows to move the robot relative to its current position. It can be called from the menu items under "Action" → "Relative Motion".
Under "Type" the following modes of relative movement can be selected:

- "Joint": The relative offset is specified in axis coordinates. The motion speed is specified by "Speed" in percent of the maximum allowed motion speed for the respective robot axes.
- "Linear - Base": A linear motion is performed with an offset specified in Cartesian coordinates. The coordinate system used for the offset is the robot coordinate system. The

velocity is specified by "Velocity". It is measured in mm/s, if it exceeds the maximum allowed motion speed of the robot, it will cause a kinematic error during execution.

- "Linear - Tool": A linear movement is performed with an offset specified in Cartesian coordinates. The coordinate system used for the offset is tool coordinates. The speed of movement is specified by "Speed". It is measured in mm/s. If it exceeds the maximum permissible movement speed of the robot, this will result in a kinematic error during execution.

10.6 Gripper and digital In-/Outputs

10.6.1 Digital Inputs

Digital inputs and global signals are available as reserved words in conditional statements (see section **??**) For global signals these are GSig1, GSig2, etc. For digital inputs Din1, Din2, etc.. Depending on the condition, they are evaluated as "true" or "false".

10.6.2 Digital Outputs

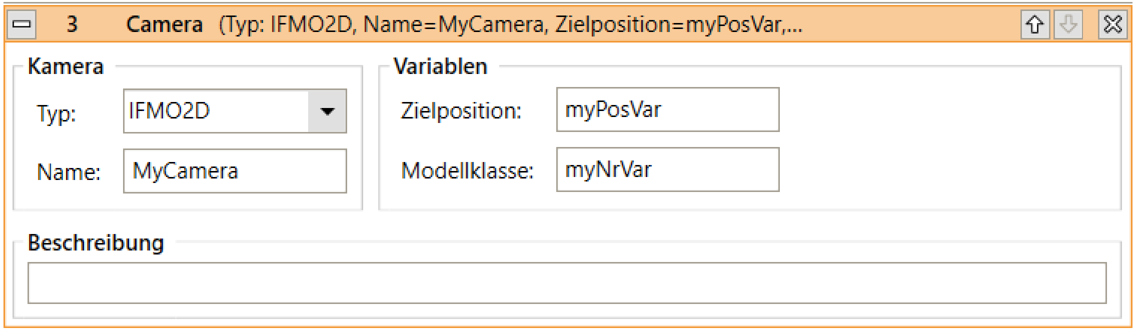
The Digital Output command is used to set digital outputs and global signals.
Under "Channel Type" it is specified whether a digital output or a global signal is to be set. Under "Channel ID" the channel of the digital output or the global signal is specified, under "State" the desired state after execution of the command is specified. The command is accessible in the program editor under "Action" → "Digital output".

10.6.3 Opening/Closing the Gripper

The Gripper command allows controlling the robot's gripper. It is accessible in the program editor through the menu item "Action" → "Gripper".
Under "Opening" you can set the desired opening, measured in percent. A value of 0% stands for a completely closed gripper, 100% for a completely opened gripper. For grippers that can only be either fully opened or fully closed, the threshold between these states is 50%.

10.7 Camera

The camera instruction allows object information to be retrieved from an object detection camera. The information includes grab position and orientation, as well as object type and detection state.
To use a camera, it must be defined and calibrated in the configuration area (see section **??**). The program instruction can be added via the menu item "Special Commands" → "Camera".



3 Camera (Typ: IFMO2D, Name=MyCamera, Zielposition=myPosVar,...

Kamera

Typ:	IFMO2D
Name:	MyCamera

Variablen

Zielposition:	myPosVar
Modellklasse:	myNrVar

Beschreibung

Figure 34: Camera command in the program editor.

Under Type the type of camera must be selected, under Name the name defined in the configuration must be entered. The output variables for target position and model class must have been declared beforehand by store command. The target position contains the position and orientation of the object in the coordinate system of the robot, while the model class contains an identification number for the detected object type. If no object was detected the value of the model class is "-1".

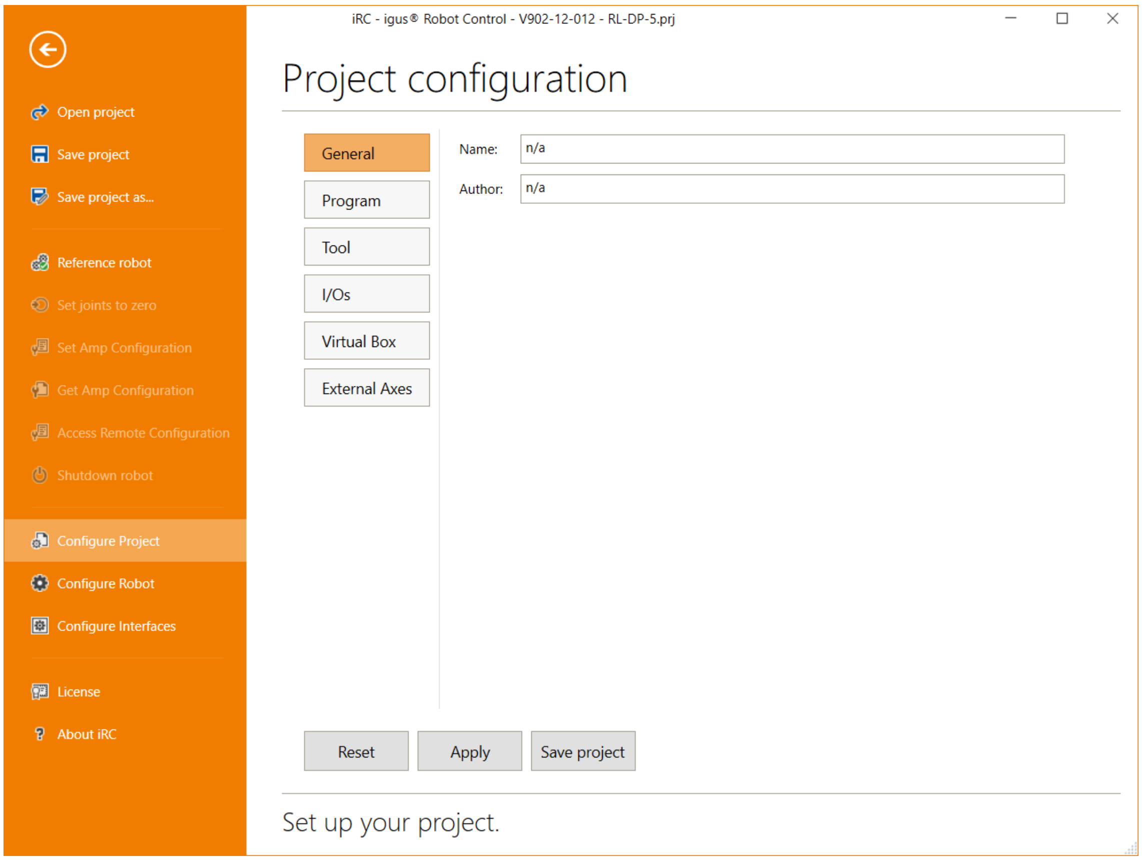
The camera command does not wait if no object has been detected. Use an If statement or a condition loop to check whether the camera has actually detected an object!

The orientation of the object can cause a slow linear movement of the robot!
If the rotation of the tool axis takes longer than the movement of the tool to the object position, then the movement is slowed down accordingly. This happens even if no tool axis is installed. If the object orientation is not relevant or no tool axis is installed, this can be avoided as follows:

1. Determine the orientation of the robot before moving to the object position, for example by defining a new position variable there and initializing it with the current position. If no tool axis is installed you can use the constant orientation values from the information area of iRC.
2. Create three assignment statements (Math statement) and overwrite the A, B and C components of the target position with the determined values.

11 Configuration

The behavior of the robot can be changed via the configuration. The most important parameters can be found in the configuration area of iRC - igus Robot Control, which can be opened via "File" (see Fig. 35). Settings concerning the project can be found under "Project configuration", cross-project settings under "Robot configuration". The interfaces can also be configured on a project-by-project basis via "interface configuration".



Name: n/a
Author: n/a

Figure 35: The project configuration area.

More specific settings can be made via the project, robot, and tool configuration files. The settings of the axis modules can be accessed and changed via "Get/Set Amp Configuration" (see section 11.4).

Change the configuration files only if you know what you are doing! Test the robot carefully, because it could move unexpectedly fast or collide! Changes of the firmware parameters can lead to overheating of the motors or the electronics!

If you use a robot with integrated controller (TinyCtrl), the changes must also be made there. When making changes via the configuration area, connect the robot first. Clicking on "Apply" or "Save Project" will automatically synchronize the changes with the robot controller. Changes to the configuration files must be transferred manually, use the "Access Configuration" area.

Some changes to the integrated robot controller are only applied after a restart. Wait at least 20 seconds after the transfer and restart the robot.

11.1 Project Configuration

11.1.1 Program

Here you can set the robot and logic program, the movement speed (as a percentage of the maximum speed), the replay mode and the reaction to program errors.

11.1.2 Tool

The mounted tool can be defined here. New tools can be defined as configuration file in the directory "Data/Tools". Changing the tool requires reloading the project or restarting the integrated control.

11.1.3 In-/Outputs

The digital I/O modules and the global signals can be configured here.
Basic inputs/outputs are currently only relevant for Mover series robots. IO modules are entered via the section "DIN-Rail Inputs/Outputs", enter the number of modules used for this.
You can enter a descriptive name for each input and output. This is only relevant for the display, but not as name in program conditions. The reset state indicates which state an output changes to on reset, the error state is set if an error occurs.

If no output or input of an I/O module is assigned a name, it is not displayed in the status area of iRC.

The global signals can be used as binary flags or virtual inputs/outputs by the robot program.

11.1.4 Virtual Box

The virtual box defines an area that the tool center of the robot must not leave. If the limits are exceeded, the movement stops.

11.1.5 External Axes

iRC supports up to 3 additional axes, which can be added and configured via the configuration area (File → Project configuration → External Axes). If a robot with integrated controller is to be configured, iRC must be connected to it.
By changing the number at "Number of external axes" external axes can be added or removed. A parameter set with the following entries appears for each axis:

Type	Type of axis. Is only needed if axis data is to be imported from a configuration file, parameters such as the orientation of the axis are relevant or the axis is to be displayed in the 3D area
Kinematics	If the robot is mounted on the linear axis enter "Dependent"
CAN-ID	CAN-ID of the axis module
Encoder steps per unit	Ratio factor from mm or degrees to encoder step (calculation see below)
Position min/max	Motion limits
Speed max	Maximum speed
Acceleration	Acceleration
Acceleration increase	Acceleration increase

If "Type" is set but there is no configuration file with the same name the axis will not work. Normally this field should be empty.

After changing the external axes, the project must be saved and reloaded. A robot with integrated control must be restarted after at least 20s.

The calculation formula 1 shows the calculation of the encoder steps per unit from the number of encoder steps, the gear ratio and the travelled distance of a full output revolution in any unit. Example 2 shows the calculation of a rotation axis, as it is installed for example as axis 4 of a RL-DP-5. Example 3 shows the calculation of a linear axis without gear.

$$\frac{4 \cdot encodersteps \cdot gearratio}{distance} = encoderstepsperunit \tag{1}$$

$$\frac{4 \cdot 500 \cdot 38}{360°} = 211,111 \tag{2}$$

$$\frac{4 \cdot 500 \cdot 1}{70mm} = 28,57 \tag{3}$$

11.2 Robot Configuration

11.2.1 General

Here a digital output for the axis brake can be defined. The output is switched to high when the motors are released so that it can control a relay that releases the electromagnetic motor brakes. In case of an error the output is set to low.

11.2.2 Gantries - Changing the axis lengths

The axis lengths of gantry robots can be changed in the configuration area (file → Robot configuration → Gantry). Depending on the number and configuration of axes, different parameters are available for this purpose.

Axis length	Length of the axis in the 3D simulation
Motion max	Usable length of the axis

After changing the axis lengths, the robot configuration must be saved and reloaded. A robot with integrated control must be restarted after at least 20s.

11.3 Interfaces

For communication with and control by other software and devices iRC offers a PLC interface, the CRI Ethernet interface and an interface for connecting ifm O2D cameras.

11.3.1 PLC Interface

The PLC interface enables the execution of basic functions and the signaling of states by means of digital inputs and outputs. In addition to control by a PLC, this interface also enables operation by hardware pushbuttons.
The PLC interface requires unused digital inputs and outputs. It reacts to rising edges at inputs. The following input functions are supported:

Enable	Executes Reset and Enable
RequestReference	Starts the referencing of all axes
Play	Starts the execution of the loaded program, stops the execution of a running program or continues a paused program
Pause	Pauses a running program or continues a paused program
Add-Joint	Add-Axis statement to the current position in the program editor
Add-Linear	Add-Linear statement to the current position in the program editor

Add-Joint and Add-Linear can only be used if iRC is connected to the robot.

The following output functions are supported:

NoFault	Output is active when the robot is in fault-free state
RobotIsReferenced	Output is active when all axes are referenced
ProgramRunning	Output is active when a program is running

The configuration of the PLC interface is done via the configuration area in iRC (File → Configure Interfaces → PLC interface). To configure a robot with integrated control it must be connected. The field "Active" activates the interface, "Automatically connect" causes iRC to try to connect to the robot automatically. The number fields to the inputs and outputs correspond to the numbers of the digital inputs/outputs. To disable individual functions, select a number that is not present on any hardware module, e.g. "1".

11.3.2 CRI Interface

The CRI interface allows you to send complex instructions and retrieve information and settings via the Ethernet interface using TCP/IP. iRC uses this interface to connect to robots with

integrated control or other instances of iRC. By default, this interface is disabled in iRC.
Documentation of all supported instructions as well as sample code can be found at the following link:

`https://wiki.cpr-robots.com/index.php/CRI_Ethernet_Interface`

To enable the interface in iRC open the configuration area (File → Configure Interfaces → CRI interface). The "Start" button starts the CRI server, "Stop" stops it. The status field shows the bound IP address and port, and the number of connected clients. If a specific IP address is to be used, it can be entered in the field below.

11.3.3 Camera Interface for ifm O2D

The camera interface enables the use of an object detection camera. Currently only the ifm O2D camera is supported as well as cameras that can emulate the TCP/IP protocol (as described here: `https://wiki.cpr-robots.com/index.php/Remote_Variable_Access#Protocol`). Cameras that do not perform object detection are not supported.

For step-by-step instructions and tips on parameterization, please refer to our Wiki article:

`https://wiki.cpr-robots.com/index.php/2D_Camera_Integration`

To use a camera it must be configured in the configuration area (File → Configure Interfaces → Cameras). If the camera is to be configured on a robot with integrated controller, iRC must be connected to it.
Select the type of camera ("IFM O2D") and click "Add Camera" to add a camera. The "General" area contains the following parameters:

Enabled	Enables or disables the camera. In the deactivated state, the values from the simulation area are used (only iRC)
Image enabled	If the camera sends images they will be displayed in the status area of iRC. Only images in the format "Windows Bitmap" are supported.
Name	This name is used to identify the camera in the robot program
Description	Optional description
IP address	IP address of the camera
Port	Port number of the camera

The entries in the geometry area control the processing of the positions provided by the camera (pixel position of the camera image) to positions relative to the robot:

Scaling	Scales the pixel position
Origin	Position of the camera in the robot coordinate system
Look	Viewing direction of the camera. A downward pointing camera has Z=-1
Up	X Direction of the camera in the robot coordinate system
Z Distance	Distance of the objects from the camera

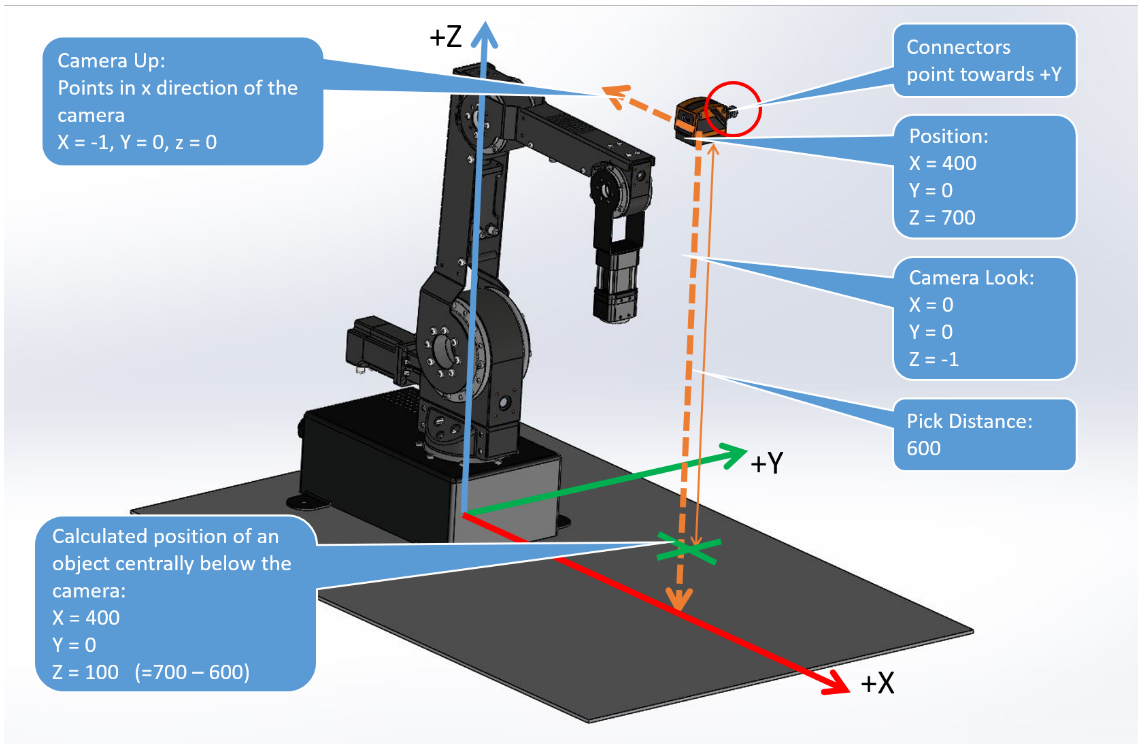
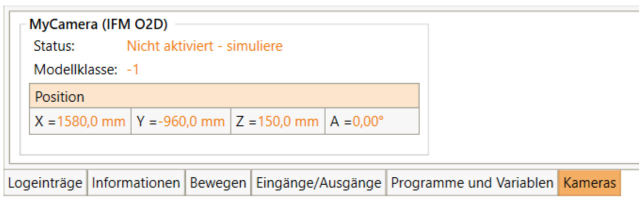


Figure 36: Measuring the camera position and orientation.

The simulation section enables the simulation of the camera. This function is not available in the integrated control.

SimX, SimY	Object position in pixels (0-640 and 0-480 respectively)
SimA	Orientation in degrees
Model class	Class of detected object, -1 corresponds to not detecting an object

After configuration, the detected and calculated values can be observed in the status area. If the camera image is activated and sent by the camera, it is also displayed here.


MyCamera (IFM O2D)

Status: Nicht aktiviert - simuliere

Modellklasse: -1

Position

X =1580,0 mm | Y =-960,0 mm | Z =150,0 mm | A =0,00°

11.3.4 Secure Shell Access

The control computer can be accessed e.g. for maintenance purposes via secure shell (ssh) (port 22), e.g. to change project files or robot files manually. Username is "robot", password is "robot".

Illustrated instructions are available here:

`https://wiki.cpr-robots.com/index.php/FTP_and_putty_Access`

The Windows software iRC and the Embedded control software TinyCtrl use configuration files. If changes are made manually (e.g. in a text editor) to the configuration files on one of these systems, they must also be made on the other side. Otherwise the controllers do not behave identically, collisions may occur!

Via the software Putty one can connect to the Linux board and work on a command line. This way you can for example view the live outputs of the robot controller.
For the work on the command line Linux knowledge is necessary!

1. Download and run Putty.exe.

   `https://putty.org`

2. Establish the Ethernet connection. Connect LAN cable to Windows PC and integrated computer. The Windows network adapter must have IP address 192.168.3.1, subnet 255.255.255.0. The integrated computer has IP 192.168.3.11 in subnet 255.255.255.0.
3. Start Putty and in the field "Host Name (or IP address)" enter the address of the integrated computer: 192.168.3.11. Set "Port" to 22 and "Connection Type" to SSH. Then click on "Open". A window will open. You may be asked if this computer can be trusted.
4. Login: robot
5. Password: robot
6. After login you are in the home directory of the "robot" user, which contains the TinyCtrl directory, whose content is similar to the iRC directory in Windows.
7. Robot files are located in ~/TinyCtrl/Data/Robots/<robot type>/<robot model>/<robot model>.xml.
8. The project file where the robot file is referenced are located in ~/TinyCtrl/Data/Projects/EmbeddedCtrl.prj
9. As editor nano, vim and vi are preinstalled.
10. After editing and saving the file, the controller must be restarted for changes to take effect.
11. To display the TinyCtrl log output live in the terminal (this is especially useful after changes in the files), TinyCtrl must first be terminated: killall TinyCtrl

12. Then navigate to the directory ~/TinyCtrl and start ./TinyCtrl
13. The process can be terminated by pressing Ctrl+C.
14. After restarting the controller TinyCtrl starts automatically.

11.3.5 SFTP Access

The "Secure Shell File Transfer Protocol" (SFTP) access works in principle like the SSH access in the section above, but allows the user to make changes to the project and robot files for maintenance purposes without having to use a terminal.

Illustrated instructions are available here:

`https://wiki.cpr-robots.com/index.php/FTP_and_putty_Access`

The Windows software iRC and the embedded control software TinyCtrl use configuration files. If changes are made manually (e.g. in a text editor) to the configuration files on one of these systems, they must also be made on the other side. Otherwise the controllers do not behave identically, collisions may occur!

SFTP is the SSH File Transfer Protocol. It can be used to transfer or adjust files from one computer to another. FileZilla is a free FTP program.

1. Download and install FileZilla Client:

   `https://filezilla-project.org`

2. Establish Ethernet connection. Connect LAN cable to Windows PC and integrated computer. The Windows network adapter must have IP address 192.168.3.1, subnet 255.255.255.0. The integrated computer has IP 192.168.3.11 in subnet 255.255.255.0.
3. Start FileZilla and specify the following:
   - Host: 192.168.3.11
   - Username: robot
   - Password: robot
   - Port: 22

   Then click Quickconnect.
4. The SFTP connection to the robot is now established:
   - in the left window the local directory structure of the Windows PC is shown.

- The right window of FileZilla shows the directory structure on the Linux computer.
- The embedded equivalent of the iRC control software is called TinyCtrl. It is located in the ~/TinyCtrl folder.
- The directory structure inside the TinyCtrl folder strongly resembles that of iRC.

Adjusting parameters in a robot or project file

1. To do this, navigate to the appropriate folder and copy the file to a local folder using "Drag and Drop".
2. Here you can edit the file with a standard text editor like Windows Notepad. (Notepad++ is a better text editor `https://notepad-plus-plus.org/`).
3. Once all changes are saved in the local file, it can be copied to the destination folder on the Linux PC by "drag and drop".
4. This will overwrite the file in the destination folder "Overwrite"
5. To let the configuration change take effect, the controller must be restarted.

11.3.6 Uninterruptible power supply (UPS)

Under certain conditions it may be desirable to connect a low voltage supply. The "apcupsd" daemon is integrated on the integrated control computer, which allows the controller to shut down safely in the event of a power failure and possibly a subsequent low battery level. A "APC Back-UPS Pro 1500" is supported. The UPS is connected to the control computer via USB cable. The robot controller is connected to the UPS via Schuko plug. The UPS is connected to the mains. No further configuration is necessary, because the "apcupsd" is preconfigured. If it is necessary to change the configuration, e.g. if another UPS compatible with apcupsd is connected, we refer to the apcupsd documentation:

`http://www.apcupsd.org`

11.4 Configuration of the motor controls

For the fine adjustment of the movement and the referencing, each axis module contains its own configuration set. This can be retrieved and modified via the buttons "File" → "Get Amp Configuration" or "Set Amp Configuration". After downloading, the configuration file is created in the installation directory of iRC under Data\Backup.
A detailed description of the parameters can be found at the following link:

`https://wiki.cpr-robots.com/index.php/Firmware_Parameter_Configuration`

Change the firmware parameters only if you know what you are doing. Test the robot at slow speed and watch the temperatures of the electronic modules and motors.

11.5 Advanced configuration

Things beyond that can be changed in the project and robot files.
The access to the configuration files of a robot with integrated controller is possible via "File" → "Access Remote Configuration".

Do not change the configuration files unless you know what you are doing. Test the robot at slow speed, as it may behave unexpectedly, move too fast or collide if the configuration is incorrect.

12 Modbus

The Modbus TCP protocol enables the control and retrieval of configuration and status information from TinyCtrl-based integrated robot controllers. This allows robots to be easily controlled by programmable logic controllers (PLCs) and integrated into processes with other devices.

12.1 Licensing

The use of the Modbus functions requires a license. The functions can be used freely for test purposes for 30 minutes, after which another test period can be started by restarting the controller. The installation of a license file is described in the iRC documentation. Further information is available at

```
https://shop.cpr-robots.com/?product=modbus-
tcp-ip-schnittstelle
```

12.2 Configuration of the Modbus Server

The Modbus server can be configured via the configuration area in iRC. To do this, connect iRC to the robot and open the Modbus configuration (file → Interface configuration → Modbus). The Modbus server becomes active when the "Active" box is checked and "Apply" or "Save Project" is clicked. If required, the port (default: 502) and the maximum number of connections can be changed.

12.3 TIA-Portal Library

For the implementation of the PLC side with a S7-1200 or S7-1500 CPU a library is available to the user. The library contains data types and communication blocks. The download of the library can be done via the following link.

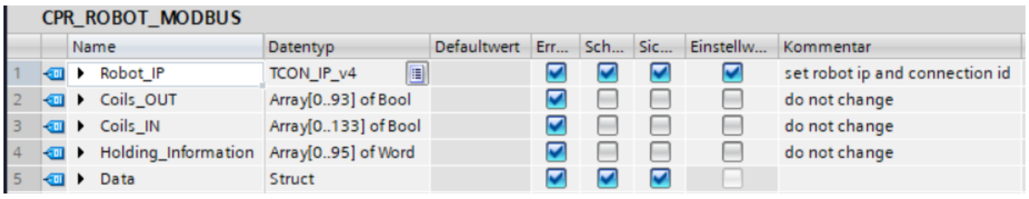
```
https://wiki.cpr-robots.com/index.php/Modbus
_Server
```

For including the library please contact the Siemens support.

```
https://support.industry.siemens.com/cs/ww/d
e/view/37364723
```

12.3.1 Creating the Robot Data Block

At first the insertion of a robot data block of type "CPR_ROBOT_MODBUS" is started. This data block contains all important information and help for the later communication with the robot. The following figure shows the robot data type. In the first place it contains a "TCON_IP_v4" object. This object can be used to set the IP address of the robot and the port to use. The connection ID can also be set.



CPR_ROBOT_MODBUS

	Name	Datentyp	Defaultwert	Err...	Sch...	Sic...	Einstellw...	Kommentar
1	Robot_IP	TCON_IP_v4		✓	✓	✓	✓	set robot ip and connection id
2	Coils_OUT	Array[0..93] of Bool		✓	☐	☐	☐	do not change
3	Coils_IN	Array[0..133] of Bool		✓	☐	☐	☐	do not change
4	Holding_Information	Array[0..95] of Word		✓	☐	☐	☐	do not change
5	Data	Struct		✓	✓	✓	☐	

If you have not changed the IP address of your robot, the corresponding addresses are already entered in the default values. In the structure Data all entries from the Modbus mapping are accessible.

Nomenclature
All data to send to the robot are marked with CMD or OUT. All data, from the robot to the controller, are marked with Info or IN.

12.3.2 Inserting the Robot Communication FB

The FB CPR_Robot is responsible for communication with the robot. This function block requires the following input signals.

Request_MB	Bool	Retrieve data from the robot. As long as this input is set, the FB maintains active communication with the robot.
Disconnect_MB	Bool	Disconnects the TCP/IP connection with the robot, can be used for resetting errors
Reset	Bool	Resets the robot
Enable	Bool	Enables the robot
Reference	Bool	References all robot joints
StartProgram	Bool	Starts the robot program
StopProgram	Bool	Stops the robot program
Robot_Data	CPR_ROBOT_MODBUS	In/Out for the robot data block

The Robot_Data input provides the function block with all the data it needs to communicate with the robot. By using several data blocks and CPR_Robot FB's it is possible to communicate with several robots at the same time. The following signals are available as outputs.

`Enabled`	Bool	Robot is enabled
`Referenced`	Bool	Robot is referenced
`ProgrammRunning`	Bool	The robot program is running

12.3.3 Data Access

To access the robot data, the data in the robot DB can be manipulated. These are then automatically transferred to the robot and processed.

12.4 Address Mapping

This section describes the address allocation to allow own implementations and extensions of the PLC function blocks.
Modbus defines four memory areas that can be read and written by different messages. In the address mapping of the igus robots, the areas are used as follows. In some cases information can be retrieved both bitwise and as register.

Address section	**Access**	**Usage**
Coils	1 Bit, read and write	Actions and changeable states
Discrete Inputs	1 Bit, read only	States, information
Holding Registers	16 Bit, read and write	Changeable values and states
Input Registers	16 Bit, read only	Not changeable values, information

Table 16: Use of the Modbus address sections

The following Modbus function codes can be used to read and write the memory areas.

Address section	**Read**	**Write**
Coils	1	5 (single), 15 (multiple)
Discrete Inputs	2	-
Holding Registers	3	6 (single), 16 (multiple), 22 (masked), 23 (read and write)
Input Registers	4	-

Table 17: Supported Modbus function codes (decimal)

Since Modbus defines only 1-bit and 16-bit access, complex data types and actions are defined here as follows:

Data type	**Description**
boolean	Writing bit "0" or "1" starts the corresponding action or sets the state.
enum	Enumeration. Meaning depends on the register, see section 12.4.5.
Info	Information, not writable

int32 / uint32	Two 16-bit registers, least significant register first.
rising edge _|¯	Action is executed when first a 0 and then a 1 is written. Attention: Some coils return the actual value when reading, not the last written one. In case of double assignment (e.g. enable/disable motors) the action is chosen depending on the actual state.
string	character string. Two 8-bit characters per register, least significant byte first. The string ends with a zero byte or when the maximum number of registers is reached.

Table 18: Definition of complex data types

The following tables describe the Modbus address assignment version 1 (see input register 3).

12.4.1 Coils and Discrete Inputs - 1 Bit, Read Only

Via coils and discrete inputs 1-bit accesses are possible. This is used here to query inputs and outputs as well as simple states and to trigger actions.

So that fewer Modbus messages have to be sent for reading, read access to 1-bit data is possible both as coils and as discrete inputs. It should be avoided to read values as discrete input, which were written as coil before. Therefore it is recommended to use only the coil access.

Addresses	**Type**	**Description**
10	Info	Has robot axes
11	Info	Has external axes
12	Info	Has gripper axes
13	Info	Has platform axes
14	Info	Has digital input/output modules
20	Info	Modules - No error
21	Info	Module error - Temperature
22	Info	Module error - Emergency stop / undervoltage
23	Info	Module error - Motor not activated
24	Info	Module error - communication
25	Info	Module error - contouring error
26	Info	Module error - encoder error
27	Info	Module error - overcurrent
28	Info	Module error - driver error
29	Info	Module error - Bus dead

30	Info	Module error - module dead
31-36	Info	Module error - reserved for future errors
37	Info	Kinematics - no error
38	Info	Kinematics - axis limit min
39	Info	Kinematics - axis limit max
40	Info	Kinematics - Central axis singularity
41	Info	Kinematics - Out of range
42	Info	Kinematics - wrist singularity
43	Info	Kinematics - Virtual box reached
44	Info	Kinematics - Movement not allowed
45-49	Info	Kinematics - reserved for future errors
50	↑	Is CAN bus connected? / Connect (1) / Disconnect (0) (Connect / Disconnect not possible with TinyCtrl)
51	↑	Shutdown control computer
52	↑	Robot reset
53	↑	Are the motors active? / Enable motors (1) / Disable motors (0)
54	Info	Normal operation (see operation mode, table **??**)
60	↑	Are all axes referenced? / reference
61-66	↑	Is robot axis referenced? / reference
67-69	↑	Is external axis referenced? / reference
70-72	↑	Is gripper axis referenced? / reference
73	↑	Set all axes to 0
100	↑	Start MoveTo - cartesian
101	↑	Start MoveTo - Cartesian relative base coordinates
102	↑	Start MoveTo - Cartesian relative tool coordinates
103	↑	Start MoveTo - joint movement
104	↑	Start MoveTo - joint movement relative
110	Info	Is Zero-Torque (manual guidance mode) available?
111	boolean	Is Zero-Torque enabled? / enable (1) / disable (0)
112	Info	Is the robot moving?
120	Info	Is a robot program loaded?
121	Info	Is a logic program loaded?
122	↑	Is the robot program running? / start / continue
123	↑	Is robot program paused? / pause
124	↑	Is the robot program stopped? / stop
130	↑	Select next directory entry

131	↑	Select previous directory entry
132	Info	Is the selected directory entry a program file
133	↑	Load selected directory entry as robot program / open directory
134	↑	Go to the base directory (.../Data/Programs)
135	↑	Unload robot program
136	↑	Unload logic program
200-299	boolean	Global signals
300-363	boolean	Digital outputs
364-427	Info	Digital inputs

Table 19: Assignment of coils and discrete inputs

12.4.2 Input Registers - 16 Bit, Read Only

The input registers provide read access to configuration, status and statistical information. To convert numerical values into the correct unit, multiply by the factor specified under Unit. The meaning of the state registers with data type enum is described in section 12.4.5.

Addresses	Type	Unit	Description
0	uint16		Software ID (902=iRC, 980=TinyCtrl)
1	uint16		Software major version (e.g. 12)
2	uint16		Software minor version (e.g. 6)
3	uint16		Modbus mapping version
4-5	uint32	minutes	Uptime complete
6-7	uint32	minutes	Uptime last
8-9	uint32	minutes	Uptime enabled
10-11	uint32	minutes	Uptime movement
12	uint16		Program starts
13	uint16	0.1ms	Cycle time target
14	uint16	0.1ms	Cycle time max (last 50 cycles)
15	uint16	0.01Hz	Cycle frequency (average)
16	uint16	0.01%	Workload
20	uint16		Number of robot axes
21	uint16		Number of external axes
22	uint16		Number of gripper axes
23	uint16		Number of platform axes
24	uint16		Number of input/output modules
25-30	bit field		Module error codes robot axes

31-33	Bit field		Module error codes external axes
34-36	bit field		module error codes gripper axes
37-40	bit field		module error codes platform axes
41-43	bit field		module error codes input/output modules
44-49	int16	0.1°C	Temperature electronics robot axes
50-52	int16	0.1°C	Temperature electronics external axes
53-55	int16	0.1°C	Temperature electronics gripper axes
56-59	int16	0.1°C	Temperature electronics platform axes
60-65	int16	0.1°C	Temperature motors robot axes
66-68	int16	0.1°C	Temperature motors external axes
69-71	int16	0.1°C	Temperature motors gripper axes
72-75	int16	0.1°C	Temperature motors platform axes
76-81	uint16	mA	Currents robot axes
82-84	uint16	mA	Currents external axes
85-87	uint16	mA	Currents gripper axes
88-91	uint16	mA	Currents platform axes
92	uint16	0.01V	Voltage
93	uint16	mA	Total Current
94	uint16	0.1%	Battery charge (not in TinyCtrl)
95	uint16	enum	Kinematics - error code
96	uint16	enum	Operating mode
130-135	int32	0.01mm	Current Cartesian position
136-141	int16	0.01°	Actual cartesian orientation
142-153	int32	0.01	Actual robot axis position
154-159	int32	0.01	Actual axis position ext. axes
160-165	int32	0.01	Actual axis position of gripper axes
166-173	int32	0.01	Actual axis position platform
262	uint16		Number of loaded robot programs
263	int16		Number of current program, 0 for main program
264	uint16		Number of instructions in current program
265	int16		Number of current instruction, -1 if program is not running
266	enum		Reason for last program stop or pause
331	uint16		Number of entries in current directory
333-364	string		Name of the selected directory entry
365-396	string		Name of the current directory

207-210	bit field		Digital inputs
400-431	string		Info/error message short (as on manual control unit)
440-455	int16		Number variables mb_num_r1 - mb_num_r16
456-711	int16	0.1	Position variables mb_pos_r1 - mb_pos_r16 (see sec. 12.4.4)

Table 20: Assignment of the Input Registers

12.4.3 Holding Registers - 16 Bit, Read and Write

Target positions and variables as well as the name of a program to be loaded can be written via the holding registers.

Addresses	**Type**	**Unit**	**Description**
130-135	int32	0.01mm	Target position cartesian
136-141	int16	0.01°	Target orientation cartesian
142-153	int32	0.01	Target position robot axes
154-159	int32	0.01	Target position external axes
174-177	int32	0.01mm	Target position platform
178-179	int32	0.01°	Target orientation platform
180	int16	0.1	Speed for MoveTo (percent or mm/s)
181-186	int32	0.1	Target velocity of ext. axes in velocity mode
181-186	uint16	0.01%	Velocity override
188	enum		Jog mode
260	enum		Robot program RunState
261	enum		Robot program Replay mode
267-298	string		Name of loaded robot program / load on write
299-330	string		Name of the loaded logic program / load on write
332	uint16		Number of the selected directory entry
200-206	bit field		Global signals
207-210	bit field		Digital outputs
440-455	int16		Number variables mn_num_w1 - mb_num_w16
456-711	int16	0.1	Position variables mb_pos_w1 - mb_pos_w16 (see sec. 12.4.4)

Table 21: Assignment of the Holding Registers

12.4.4 Number and Position Variables

For communication with robot and logic programs, predefined program variables can be used in addition to the global signals. For this purpose, 16 readable and 16 writable number and position variables are available in each case.

Name	Type	Access via Modbus
mb_num_r1 - mb_num_r16	Number variable	read only (input register)
mb_num_w1 - mb_num_w16	Number variable	read and write (holding register)
mb_pos_r1 - mb_pos_r16	Position variable	read only (input register)
mb_pos_w1 - mb_pos_w16	Position variable	read and write (holding register)

Table 22: Program variables for communication via Modbus

When using the PLC function blocks, only use the writable variables to send values from the PLC to the robot. Do not change these variables in the robot program, as they are regularly overwritten by the PLC.

Unlike normal program variables, the Modbus variables are always available. No program has to be started and the variables do not have to be declared with the store statement.

Each number variable is mapped in a register. Since only integers are supported here, the robot program must convert to the desired value range by multiplication or division if necessary (Math instruction).
Position variables consist of 16 registers each, whose values are specified in tenth precision:

- 9 registers for robot and external axes (A1-A6, E1-E3).
- 3 registers for Cartesian position (X, Y, Z)
- 3 registers for Cartesian orientation (A, B, C)
- 1 register for selection of conversion type

According to the translation type, the kinematics converts from axis angles to Cartesian coordinates or vice versa. This can be helpful if, for example, target positions are only available as coordinates, but the robot is to move per joint instruction.

Value	Meaning
0	No conversion, axis positions and Cartesian coordinates are taken over without any change (default)
1	Cartesian coordinates and orientation are calculated from the axis positions
2	The axis positions are calculated from the Cartesian coordinates and orientation

Table 23: Conversion type

Beachten Sie, dass möglicherweise nicht alle Positionen von der Kinematik erreicht werden können. Prüfen Sie die Werte daher auf Plausibilität.

12.4.5 Meaning of Enumeration Values

The following tables describe the meaning of the enumeration values (enums).

Value	Meaning
0	No error
13	Axis limit Min
14	Axis limit max
21	Central axis singularity
22	Out of range
23	Wrist singularity
30	Virtual box violated in X+
31	Virtual box violated in X-
32	Virtual box violated in Y+
33	Virtual box violated in Y-
34	Virtual box violated in Z+
35	Virtual box violated in Z-
50	NAN in calculation
90	Motion not allowed
65535 (0xFFFF)	Unknown error

Table 24: Kinematic error code

Value	Meaning
0	Standard - normal operation
1	Serious error, control must be restarted
2	CAN-Bridge (CRI, e.g. retrieve firmware parameters)

Table 25: Operation mode

Value	Meaning
0	Axes
1	Cartesian base coordinate system
2	Cartesian tool coordinates
3	Platform

0xFFFF	Invalid

Table 26: Jog Mode

Value	Meaning
0	Program is not running
1	Program is running
2	Program paused

Table 27: RunState

Value	Meaning
0	Run program once
1	Repeat program
2	Execute instructions step by step
3	Fast (not used)

Table 28: Replay Mode

Value	Meaning
0	User (Teach pendant, CRI, Modbus, etc.)
1	PLC
2	Program (stop/pause instruction)
3	Replay Step (step operation)
4	Shutdown (system shuts down)
100	Error
101	Path generator error 1
102	Path generator error 2
103	Error in state machine

Table 29: Reason for last stop/pause of program

13 CANopen

Using CANopen, it is possible to control motor controllers and digital input/output modules from various manufacturers with an embedded control based on TinyCtrl.

13.1 Controlling Motors

For the control of motors, the CANopen device profile CiA 402 is supported with the following operating modes:

- Interpolated Position Mode
- Cyclic Synchronous Position
- Profile Velocity (only external axes)
- Homing

The control system was designed and tested with the following motor control modules in particular:

- Nanotec C5-E-1-09 und C5-E-2-09
- Nanotec CL3-E

13.2 Controling In-/Output Modules

Digital input/output modules are supported via the CANopen device profile CiA 401. Currently, up to 3 modules with up to 10 digital inputs and outputs each can be used. Inputs and outputs of motor control modules according to CiA 402 are not supported as general inputs/outputs. The controller was designed and tested with the following input/output modules in particular:

- frenzel + berg hipecs CIO300

13.3 Licensing

The use of the CANopen functions requires a license. The functions can be used freely for test purposes for a period of 30 minutes, after which a further test period can be started by restarting the controller. The installation of a license file is described in the iRC documentation. Further information is available at

`https://shop.cpr-robots.com/?product=canopen-master`

13.4 Setup

In order to use CANopen, the control modules must be configured according to the connected hardware and the communication and parameters of the robot must be defined in the software. The following sections summarize what has to be done for this.

When making changes to the project and robot configuration file, it is important to make the changes on both the integrated controller and iRC, otherwise iRC will overwrite them and may not model motions correctly. When you connect, make sure that the configuration is synchronized, accept the configuration of the integrated controller, and save the project in iRC.

13.4.1 Configuration of Control Modules

Make sure that the control modules are set up according to their instructions. Pay particular attention to current parameters and current reduction, as well as speed and acceleration limits, in order to avoid damage to the motors. If motors are to be operated in closed-loop mode, set this up.
The controller uses a baud rate of 1 MBit. The CAN IDs of the motors can be freely selected.

13.4.2 Creating the Configuration Files

To set up a new robot, a corresponding set of configuration files must be created. In the simplest case you take over the following configuration files of an existing robot in the data directory of iRC, which uses the same kinematics. Category and type can be freely chosen.

- Project configuration: Data\Projects\<Kategorie>\<Typ>.prj
- Robot configuration: Data\Robots\<Kategorie>\<Typ>\<Typ>.xml
- CANopen parameters: Data\Robots\<Kategorie>\<Typ>\CANopenParameter.xml

The CANopen parameter file is only available in CANopen projects.
Change the files as follows:

1. Copy the above files according to your own category and type name, but with same directory structure.
2. Open the project file with a text editor and change the name and type in the Robot line. Type must be "<category>\<type>".
3. Open the robot configuration with a text editor and in the INFO line change the NAME field to the name or type of your robot.

To test if the configuration was created correctly try to load the project with iRC. If this fails, check the above steps.
If necessary, the 3D files (.obj) for the visualization can be exchanged.

13.4.3 CANopen Parameter File

The CANopen parameter file contains parameters that are sent to the control modules when they are connected. This can ease parameterization by simply entering parameters into this file instead of having to send and save them to the modules via configuration software. Via the parameter file only CANopen objects can be sent whose index and subindex are predefined by TinyCtrl (see following table). If objects are not predefined, TinyCtrl writes a corresponding error message into its log file.

Index	Subindex	Beschreibung	Hinweise
0x2052	0	Encoder Resolution	
0x3240	1,2,6	Digital Inputs Control	Reference- and end switches
0x320A	3,4	Motor Drive Sensor Display	
0x3212	1	Motor Drive Flags - Legacy Power Mode	Can avoid dropping joints on reset
0x3701	0	Limit Switch Error Option Code	
0x6065	0	Following Error Window	
0x6066	0	Following Error Timeout	
0x6081	0	Profile Velocity	
0x6083	0	Profile Acceleration	
0x6084	0	Profile Deceleration	
0x607B	1,2	Position Range Limit	Overflow for rotating joints
0x607C	0	Home Offset	
0x607D	1,2	Position Limit	Software limits, but may cause that axes can only be moved in a restricted way before referencing.
0x6098	0	Homing Method	
0x6099	1,2	Speed during Search for Switch	Referencing speed
0x60F2	0	Positioning Option Code	

Table 30: Allowed CANopen objects in the parameter file

13.4.4 Robot Configuration

The robot configuration file determines among other things the used hardware protocol and the kinematics. To use CANopen the BusConfiguration line must contain the following entries:

```
<BusConfiguration Protocol="CANopen" CycleTimeMS="20" GapMS="1" Type="PCANUSB">
```

Parameter	Beschreibung
Protocol	"CANopen" or "CANopen_IPO" for IPO mode, "CANopen_CSP" for CSP mode (see section **??**).
Type	Must be "PCANUSB".
CycleTimeMS	Cycle time with which new target positions are calculated and transmitted, and information of the control modules is queried in milliseconds. E.g. 20 or 50. The value must be greater than SyncGapMS + GapMS x number of axes.

GapMS	Time in milliseconds to wait between sending the positions of each axis, by default 0ms for CANopen.
SyncGapMS	Time in milliseconds to wait between sending the SYNC message and sending the target positions. This prevents collisions of incoming and outgoing messages. Default is 0ms.
PreciseTiming	More Precise Timing: If there are frequent jerks during movements these can possibly be fixed by setting this parameter to "True", as well as GapMS and SyncGapMS to 0. Caution: This will result in a higher computational load. Default: "False

Table 31: parameter in BusConfiguration tag of robot configuration file

The number of robot axes must be entered in the Joints line. The maximum is 6. Additional axes are defined in the project file (see section **??**). The following lines Joint0 - Joint5 define the translation of the axis positions in degrees or millimeters to the unit of the control modules (e.g. motor steps). Relevant for this are GearScale and GearZero, where GearScale is the factor of millimeters or degrees to the corresponding unit of the motor control. The position limits and gear backlash compensation specified here are not currently used for CANopen.

```
<Joints NrOfJoints="6" />
<Joint0 ID="1" Min="-170.0" Max="170.0" CurrentScale="3.0" CurrentZero="70.0"
    GearScale="1090" GearZero="0" GearPlay="0.5" GearPlayInc="0.5"/>
```

In the Kinematic tag, the type must correspond to the kinematics of the robot. The lz and lx parameters must be set to the lengths of the axis elements in millimeters. The required parameters depend on the kinematics used and should be taken as a template from the configuration of an existing robot.

```
<Kinematic Type="StandardSixAxisPlus" Wrist="Standard" lz0="69.0" lx1="64.0" lz1="83"
    lx2 ="305.0" lx3="0.0" lz3="0.0" lx4="222.0" lx6="23"/>
```

The ManufacturerJointConfig line allows a simple transformation of the axis positions. For example, the axis direction can be reversed by negating the AxDir parameters or an offset can be added by AxOffset. In SoftwareMinMax the axis limits are to be entered in degrees or millimeters.

```
<ManufacturerJointConfig zOffset="0.0" A1Dir="1.0" A1Offset="0.0"  A2Dir="1.0"
    A2Offset="0.0"  A3Dir="1.0" A3Offset="90.0"  A4Dir="1.0" A4Offset="0.0"  A5Dir="
    1.0" A5Offset="0.0"  A6Dir="1.0" A6Offset="0.0"  />
<SoftwareMinMax A1Min="-110.0" A1Max="110.0" A2Min="-50.0" A2Max="65.0" A3Min="-110.0
    " A3Max="60" A4Min="-170.0" A4Max="90.0" A5Min="-70.0" A5Max="70.0" A6Min="-120.0
    " A6Max="120.0"/>
```

13.4.5 Project File

When creating a new project it is important, as already mentioned in section 13.4.2, that the type is correctly specified. This must correspond to the directory structure where the robot configuration file is located.

```
<Robot Name="My Robot Name" Type="MyCategory\MyRobotType" Homepos="0.0 0.0 0.0 0.0
    0.0 0.0" OffsetX="0" OffsetY="0" OffsetZ="0" OffsetRX="0" OffsetRY="0" OffsetRZ="
    0" Parent="-1" Tool="" FlipMesh="true"/>
```

The external axes should ideally be created via the graphical configuration of iRC. Only if the velocity mode is to be used instead of the position mode (e.g. for conveyor belts) this must be entered manually as MotionMode Velocity.

```
<ExternalAxis Type="" Kinematic="Independent" Number="0" ID="6" MotionMode="Velocity"
     GearScale="10" Min="0" Max="0" VelMax="2000" Acc="3000" AccInc="9000"
    DirectionAngleToY="0" lz0="0" Dir="1" Offset="0"/>
```

The digital input/output modules should also be configured via iRC. By default the CAN IDs 112, 113 and 114 are set for modules 1-3.

```
<DIOModule ID="112"/>
```

14 Extensions

The control is modular and can be extended. For example, a maximum of 3 DIO modules are supported as core components of the control. To install additional modules, the control must first be disconnected from the power supply. Then the bus connector enclosed with the new module is plugged onto the top-hat rail and then inserted into the existing bus connectors. Now the new module can be plugged on and is electrically connected to the bus system.

14.1 Additional Motor Modules / External Axes

The CAN address of the module must be set so that the controller can address the new module. The switch position specifies the CAN address as follows:

- The module of the 1st axis has switch position 0: CAN address 0x10 (Dec: 16)
- The module of the 2nd axis has switch position 2: CAN address 0x20 (Dec: 32)
- The module of the 3rd axis has switch position 4: CAN address 0x30 (Dec: 48)
- The module of the 4th axis has switch position 6: CAN address 0x40 (Dec: 64)
- The module of the 5th axis has switch position 8: CAN address 0x50 (Dec: 80)
- The module of the 6th axis has switch position 9: CAN address 0x58 (Dec: 88)

The wiring to the motors of the additional axes is identical to the wiring of the remaining motor modules. See section 8.2. Now the Additional Axis must be included in iRC. This works as described in section 11.1.5.

14.2 Additional Digital I/O Modules

In order for the controller to address the new module, the CAN address of the module must be set. The switch position specifies the CAN address as follows:

- The 1st DIO module has switch position 0: CAN address 0x70 (Dec: 112)
- The 2nd DIO module has switch position 2: CAN address 0x80 (Dec: 128)
- The 3rd DIO module has switch position 4: CAN address 0x90 (Dec: 144)

The wiring of the modules with actuators and sensors is described in chapter 8.3. After starting the controller, the new DIO module must still be entered in the software. See section **??**

14.3 Brake relay for motor brakes

To prevent an axis from dropping, electromagnetic brakes can be used that open when a voltage is applied. Without voltage they are breaking by springs.
The robot controller iRC offers a function to control brakes. A digital output is set to 24V whenever the brake is to be released.

14.3.1 Electrical connections to the brake

Disconnect the robot and controller from power before wiring the brake. The controller sets an output of the DIO module of the controller. Since this cannot switch the currents needed to release the brake, a relay must be interposed, such as Wago 788 series relays (e.g. 788-312).

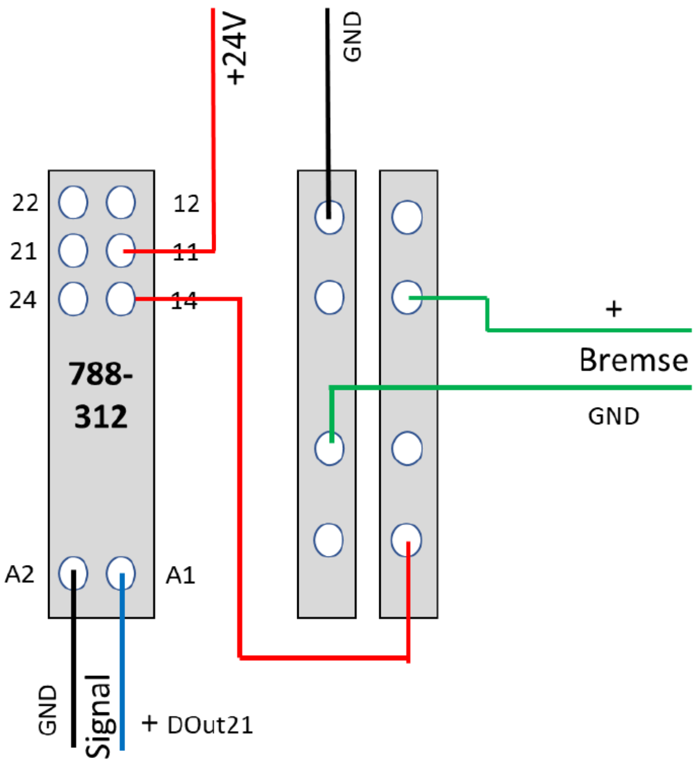


Figure 37: Brake relay and brake connection

The temperature development of the motors should be observed when the brake is released. If the motor brake has no integrated current reduction, it may be useful to reduce the braking current by means of an additional circuit.
Now the brake must still be activated in the software. See section 11.2.1.

15 Moving the Robot via Handheld Control Unit

To operate the robot without a Windows PC connected, the integrated computer is required. It runs the TinyCtrl software as the robot control software. The integrated computer allows the robot arm to be moved (via the handheld control unit) and robot programs to be played. To set up new programs, it is connected via Ethernet to a Windows computer running iRC.
This section shows how to operate the robot with integrated computer and handheld control unit.
After all electrical connections to the robot have been made, the robot has been powered on, and the emergency stop switch has been released, the faults must first be "reset" and then the robot must be "activated".
When moving the robot, always keep one hand on the emergency stop switch to prevent it from hitting an object unexpectedly, e.g. if it collides with the table.

15.1 Robot Reset and Enable

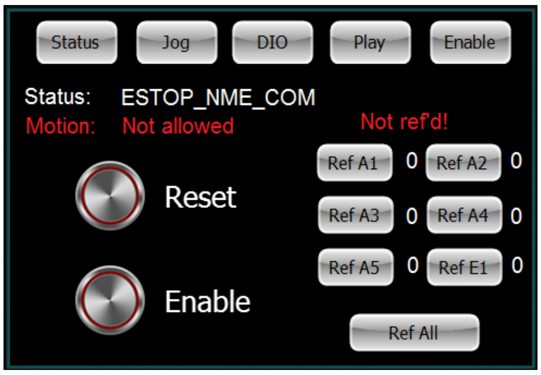

Status: ESTOP_NME_COM
Motion: Not allowed
Not ref'd!

Figure 38: Press the "Enable" button in the top menu (top right) on the touchscreen display to go to the Enable page.

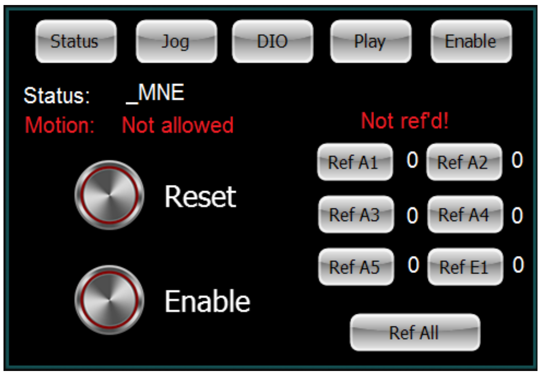

Status: _MNE
Motion: Not allowed
Not ref'd!

Figure 39: Now press "Reset": The status changes to "MNE" (Motor Not Enabled).

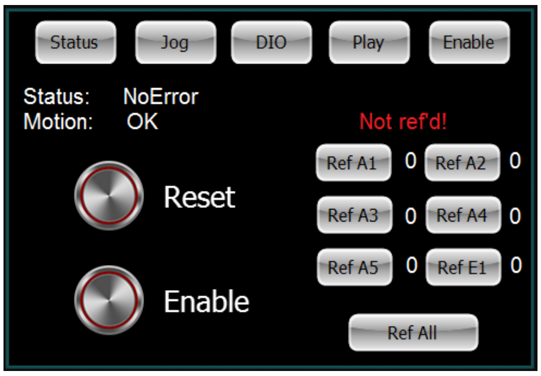


Figure 40: Press "Enable": The status changes to "NoError". "Not refd!" (in red) means "not referenced".

15.2 Moving the robot with the 3-axis joystick

Once the robot is activated, the axes of the robot can be moved.

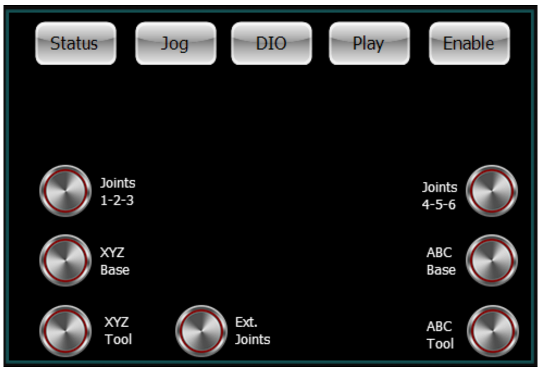


Figure 41: To do this, press the "Jog" key at the top of the display. 2. Press "Joints 1-2-3". Then move and rotate the joystick. You can now move axes 1, 2 and 3. 3. Press "Joints 4-5-6". Now move axes 4 and 5 by moving and turning the joystick.

15.3 Referencing

To enable automatic program execution and linear movements ("XYZ Tool" / "XYZ Base")the robot must be referenced. The type of reference movement depends on the robot or encoder type. In case of doubt, each axis of the robot should first be moved (Fig. 41) close to the reference sensor of the axis as described above.

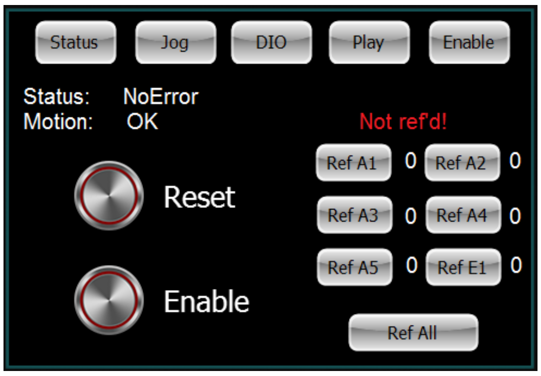


Figure 42: Navigate to the Enable menu via the button "Enable" in the upper right corner.

- Press the round "Reset" button followed by the round "Enable" button: the status changes to "NoError". "Not refd!" (in red) means "not referenced".
- The robot must be in the status "NoError".
- Now press the button "Ref All" to reference all axes. The robot will now perform search movements for each axis. A successfully referenced axis is indicated by the number "0" behind the button "Ref XXX" changing to "1!.
- After referencing, the axes are in error state. This is necessary because the actual position has changed during the referencing process.

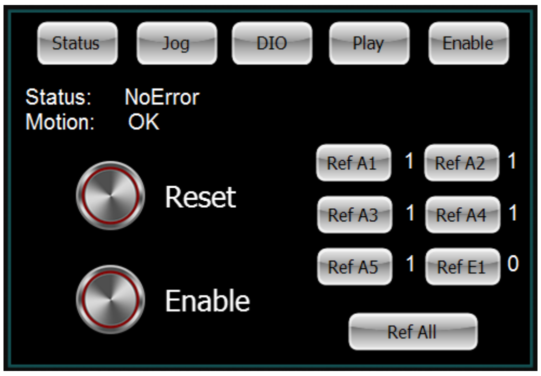


Figure 43: Now press the round button "Reset" again, followed by the round button "Enable". The robot is now referenced. Linear movements and programs can now be executed.

15.4 Starting and Stopping a Program

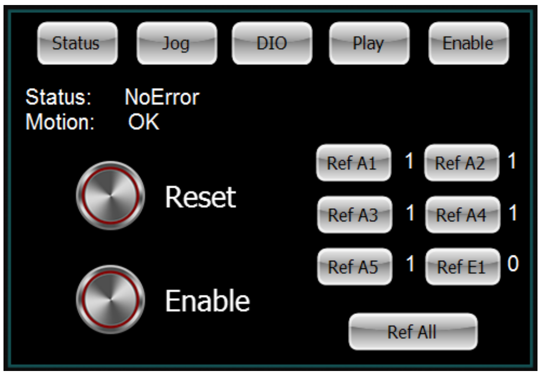


Figure 44: To do this, press the "Play" key at the top of the display and use the "prev" and "next" keys to select a program written for your robot.

- Load the program with the button "Load".
- Let the program play once with "Single Play".
- "Cont. Play" plays the loaded program continuously.
- "Stop" stops the movement.
- The slider "Override" can be moved to the right to increase the speed of the movement or to the left to decrease it.

15.5 Manual Setting of the Digital Outputs

The digital outputs can be set by pressing the switches shown. With the three buttons on the left side you can switch between several DIO modules.

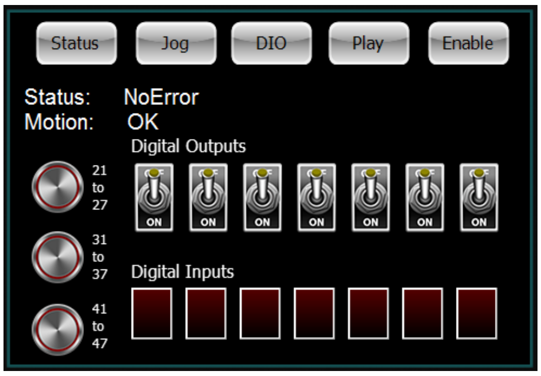
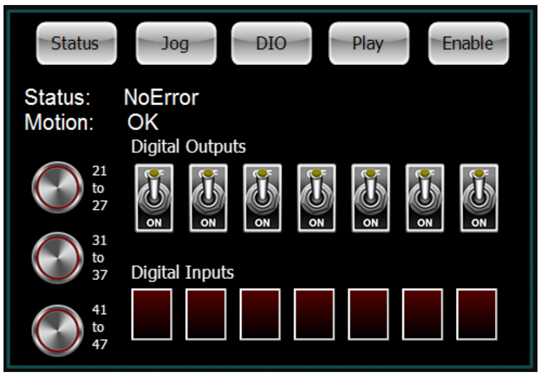


Figure 45: If, for example, a gripper is connected to the DIO module, it can be activated or deactivated by toggling the switches shown.

15.6 Reading the States of the Digital Inputs

The digital inputs are read out by the controller during operation and shown on the display. With the three buttons on the left side you can switch between several DIO modules.

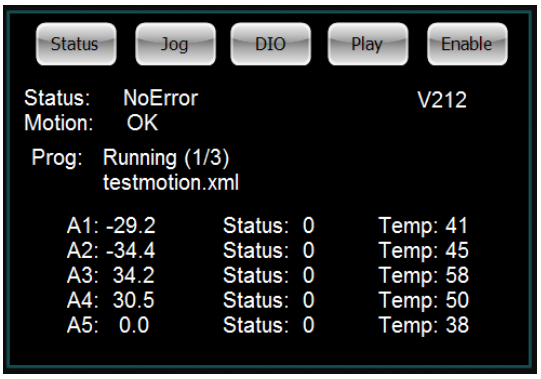


Status: NoError
Motion: OK

Figure 46: When a signal is present at the digital input, it is shown in the digital input field at the bottom of the display.

15.7 Displaying Status Information

Status information can be viewed by pressing the "Status" key in the upper left corner of the display.



```
Status:   NoError                         V212
Motion:   OK
 Prog:  Running (1/3)
        testmotion.xml

    A1: -29.2        Status:  0        Temp: 41
    A2: -34.4        Status:  0        Temp: 45
    A3:  34.2        Status:  0        Temp: 58
    A4:  30.5        Status:  0        Temp: 50
    A5:   0.0        Status:  0        Temp: 38
```

Figure 47: The name of a program is displayed, e.g. while a program is running. The axis positions of the axes 1-5 are displayed (A1-A5). The temperature of the stepper modules is also displayed. Due to the different loads of the individual axes, different holding currents are applied, which lead to changing temperatures of the stepper modules.

16 Maintenance

16.1 Cleaning

- After the control unit is disconnected from the mains, it can be wiped with a damp cloth.
- After the control unit is disconnected from the mains, fans or ventilation slots can be cleaned carefully with a damp cloth or light compressed air. In doing so, the rotors of the fans must be held firmly so that they do not receive bearing damage due to excessive speed (rpm).

17 Troubleshooting

17.1 Frequently Asked Questions

Antworten zu häufig gestellten Fragen finden Sie in unserem Wiki:

https://wiki.cpr-robots.com

17.2 Error Codes and Solutions

Our troubleshooting guide provides step-by-step assistance in identifying and solving problems:

https://wiki.cpr-robots.com/index.php/Troubleshooting

Error codes and problems with hardware and electronics are described in the following article:

https://wiki.cpr-robots.com/index.php/Robot_Hardware_Troubleshooting

Solutions to common software problems are described in the following article:

https://wiki.cpr-robots.com/index.php/CPRog_Software_Troubleshooting

17.3 Test Software Module Control

The Module Control software can be used to test axes individually and without the influence of the robot control. Among other things, it enables the axis to be moved and referenced, and parameters to be read out and changed (see section 11.4).
Module Control can be downloaded under the following link. The operation is also described there in more detail.

https://wiki.cpr-robots.com/index.php/Config_Software_ModuleCtrl

17.4 Support Contact

If you have any problems, we will be happy to help!

- igus Support landingpage:

  `https://www.igus.de/info/igus-low-cost-automation`

- E-Mail: ww-robot-control@igus.net

  In case of software problems, please send us the log files of the iRC - igus Robot Control and the integrated controller. To do this, simply click on the question mark at the bottom right of the 3D area to automatically attach all relevant files to an e-mail

- Telephone: +49(0)2203 / 96498-255