igus® logo

iRC - igus® Robot Control - V902-13-022 - Rebel.prj

Datei   Szene   Bewegung   Hilfe

Verbinden   Zurücksetzen   Aktivieren   Referenzieren   Langsamer bewegen   15 %   Schneller bewegen   Bewegungsmodus   Ausführen   Anhalten   Unterbrechen   Abspielmodus   Program Ausführu Anweisur Verbindu

Physischer Roboter   Bewegungsparameter   Programmausführung   Aus

iRC igus Robot Control Release 13

Zustand

Nicht verbunden

Roboter
igus Rebel
igus-Rebel\Rebel
Geschw.-Einstellung 0

Kinematik
OK

Virtuelle Box
OK

Zyklus
100,0 Hz
max. 11,0 ms

Referenzierung
Nicht verfügbar.

Achsen

A1 -	0,0°	A1 +	A4 -	0,0°	A4 +
A2 -	1,0°	A2 +	A5 -	-35,0°	A5 +
A3 -	115,0°	A3 +	A6 -	0,0°	A6 +

Externe Achsen

E1 -	n. v.	E1 +
E2 -	n. v.	E2 +
E3 -	n. v.	E3 +

Logeinträge   Informationen   Bewegen   Eingänge/Ausgänge   Programme und Variablen   Kameras   Statistik

Abspielmodus wurde auf 'Einmal' gesetzt.

LOW COST AUTOMATION by igus®

igus® ReBeL®
User Guide
Software and Electronics

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.eu
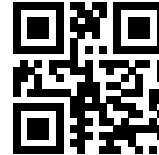
1.2 Intended Use

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
- plant designers
- assemblers/service personnel
- application engineers

Installation, commissioning, as well as operation is only allowed by qualified personnel. These are persons who meet all the following requirements.

- have appropriate training and experience in handling motors and their control.
- know and understand the contents of this technical manual.
- know the applicable regulations

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.

> ℹ️ **This note indicates possible incorrect operation of the product.**
> Failure to comply with this notice may **possibly** result 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

2. **Robot control**: modular robot controller consisting of embedded computer, 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.

2.1 Specifications

Modular Control	Feature
Power supply	24V
Type	DIN rail modules with 5-pole bus connector
Communication with PC/PLC	CRI or Modbus via Ethernet
Internal communication	CAN
Joint modules	Closed Loop motor modules
	absolute encoder
	integrated brake
Digital In/Out Base:	each 7 digital in/outputs
	solid state relay, 24V level, max 500 mA
Digital In/Out Arm:	each 2 digital in/outputs
	solid state relay, 24V level, max 500 mA

Table 2: Modular Robot Control - Components

The features of the embedded control can be seen in the table 3.

Embedded control	Features
Type	Single board computer with daughter module
Operating system	Linux based
Software	TinyCtrl robot control software
Interfaces	CAN Bus (connection to the modules)
	Ethernet (connection to Windows PC)

Embedded control	Features
	USB
	RS232 display connection
	Digital inputs for enabling switch
	DC/DC converter 5V/4A

Table 3: Integrated Computer - Specifications

2.2   Mechanical Dimensions

For mechanical dimensions, please refer to the data sheet of the ReBeL.

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 one or more 24V or 48V power supplies up to 10A, 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 4: 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 or 11 (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 Ethernet port

5   Quick Start Guide

5.1   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.

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

5.2   Switch On

⚠️ **Danger of Electric Shock!** Before commissioning the components, make sure that all cables and components are properly connected.

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.

5.3   status light

The on/off switch of the robot contains a multi-colored LED that gives an indication of the status. When starting up, it is initially yellow until the integrated control software has connected and the motors have been enabled by the user.

color	meaning
green	Motors enabled
yellow	Motors disabled (no error) OR robot control software is not yet running / has stopped OR no communication to the switch LED, e.g. due to an error in the configuration of the input/output modules. Wait approx. one minute after starting. Try to activate the motors via iRC. If this is not possible, reset the configuration of the robot via a software update.
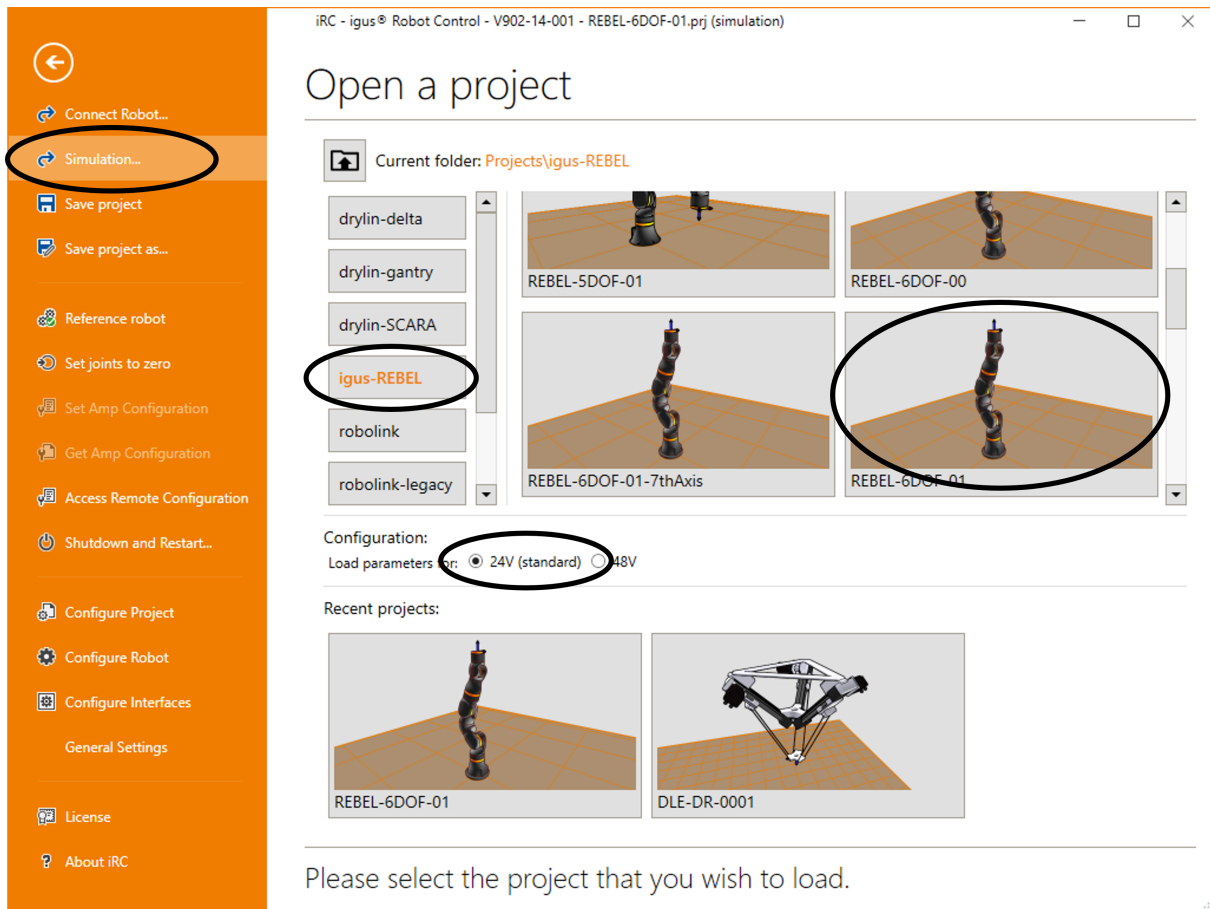
Figure 1: Choosing a project

| red | Error. Try to reset the error by activating the motors. iRC provides further information. |

5.4   Connecting and moving the robot

5.4.1   Preparation with the integrated computer

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

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 of the robot.

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"..

5.4.2   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. 1).
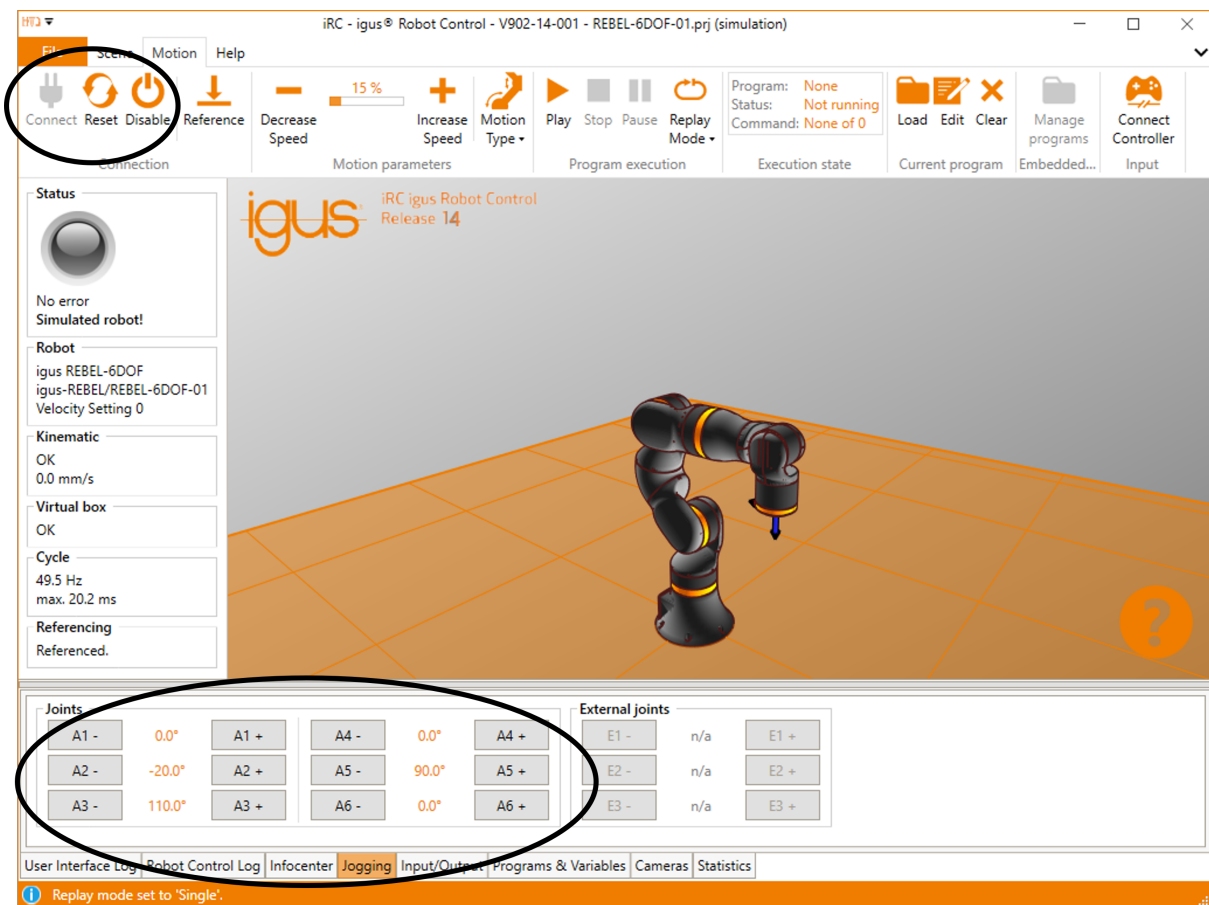
Figure 2: Jog Commands

3. You can now activate the robot by pressing the following buttons in the given order (see also Fig. 2):

    3.1 "Connect"

    3.2 "Reset"

    3.3 "Enable"

> Wait a few seconds after activation before moving the axes or starting a program. The axes release their brakes and align themselves before they can be moved. You may hear clicking noise during this process.

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. 2).

6   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.

6.1   The Graphical User Interface

This section explains the iRC software. All steps can be simulated even without a robot connected. In section 6.2 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).
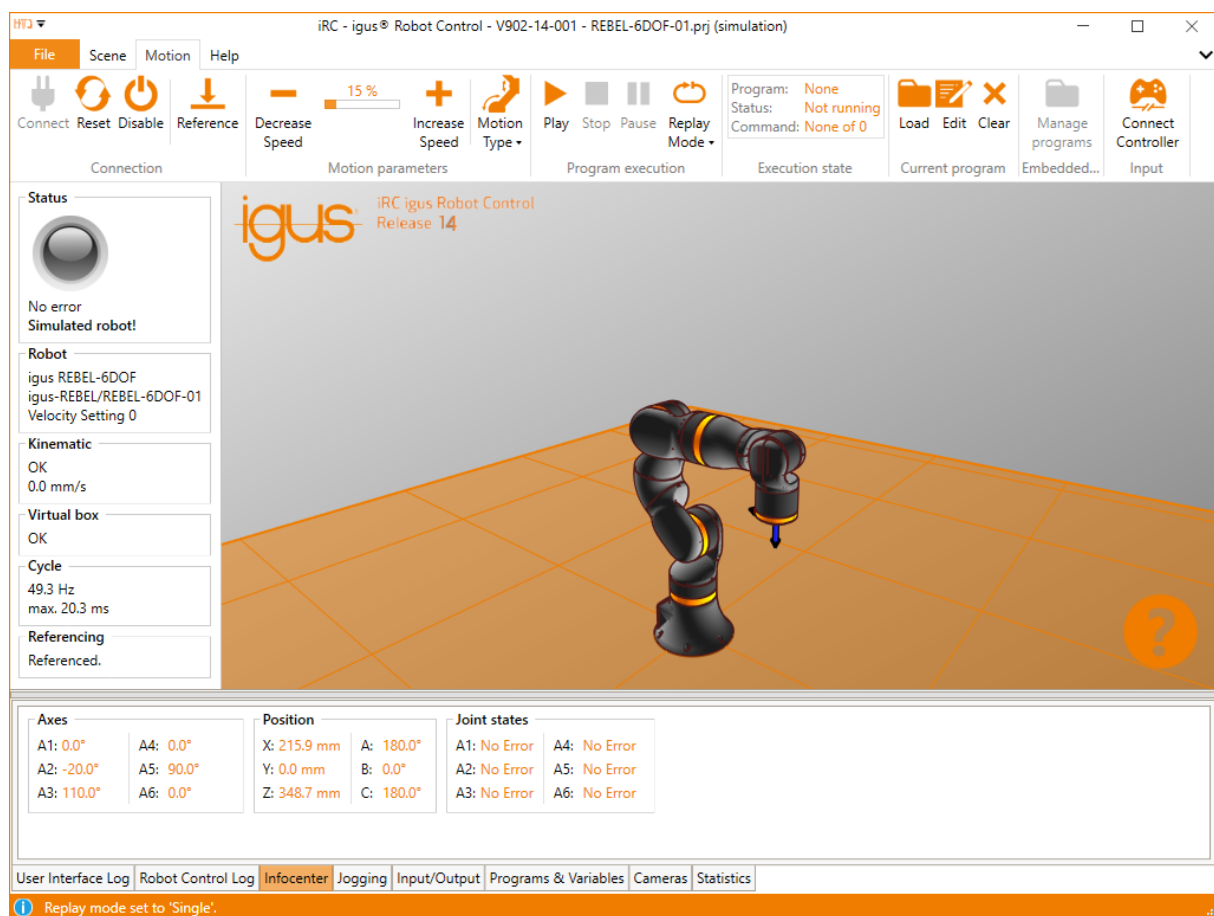
Figure 3: User interface of the iRC - igus Robot Control

In the upper left corner, the three tabs "File", "Scene", "Motion" and "Help" provide access to the main functions. At the left side, 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. 3).

You will find the following tabs at the bottom of the window:

- "User Interface Log": Messages about events and errors of the graphic user interface,
- "Robot Control Log": Messages about events and errors of the robot control.
- "Infocenter": Displays the axis values, Cartesian position and other information.
- "Jogging": Buttons to move the robot.

- "Input/Output": Display and set the DIO interfaces of the robot controller.
- "Programs & Variables": displays the current values of the program variables.
- "Cameras": Images of the connected cameras and detected object positions.
- "Statistics": Statistics about the system and the running robot program.

> 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.

6.1.1   Simulating a robot

To simulate a robot, click on "File" → "Simulation..." at the top left. A list of robot categories and various robots in these categories will appear. Below the list you can select whether the 24V or 48V configuration of a robot should be simulated, this determines the speed. Click on a robot to simulate it.
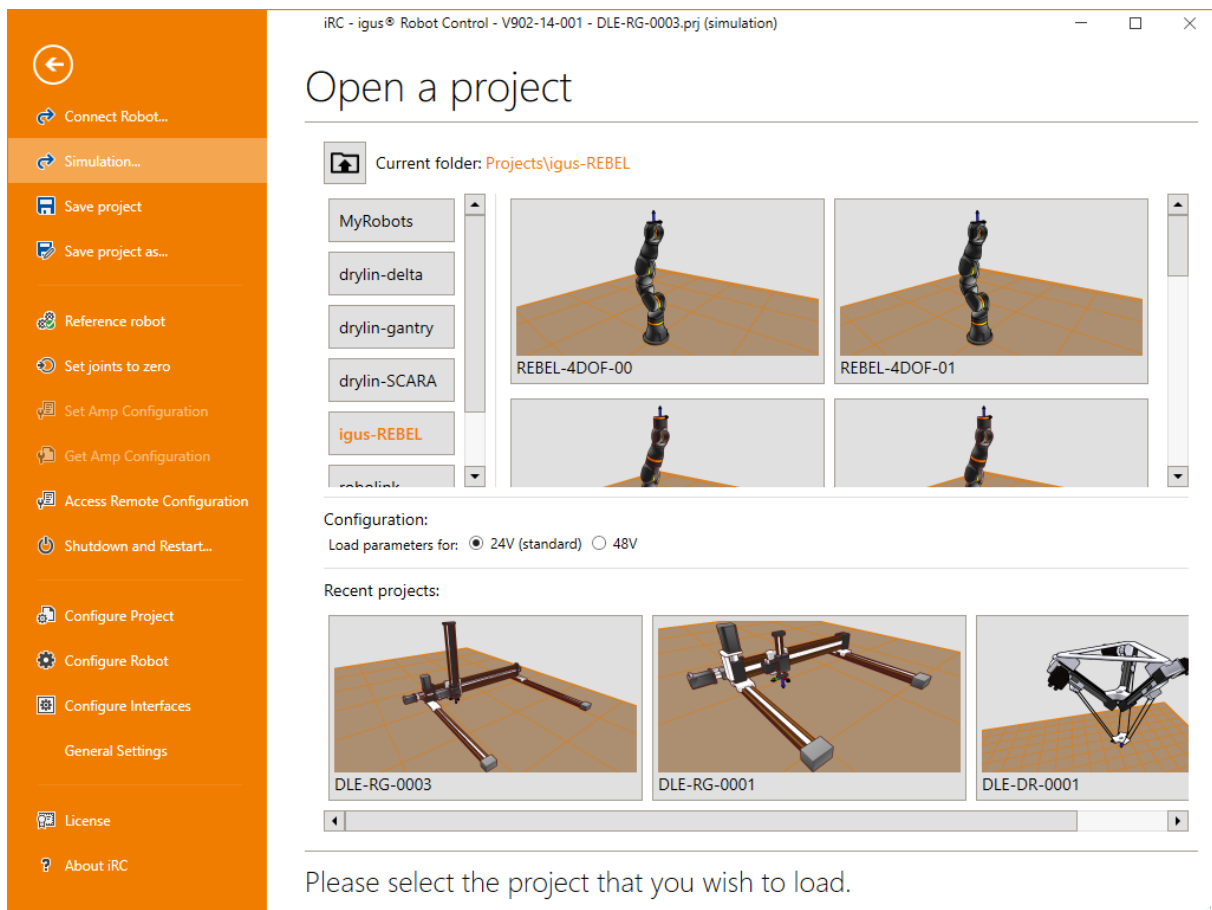
Figure 4: Selection of the robot to simulate via the menu item "File" → "Simulation..."

The entries in the list are called projects. By clicking on "Save project as..." you can save a copy of a robot with your own settings.

> The category "MyRobots" contains copies of the projects of your real robots to simulate them. The entries there are downloaded from the robot each time you connect

---

6.1.2   Connecting a real robot

To connect iRC to a real robot, click on "File" → "Connect robot...". There you will find the button "New connection" and a list of known robots.

Click on "Connect robot...". The button "Automatically connect" will then appear, which attempts to connect to a robot at a standard IP address. If the IP address of your robot has been changed, enter it on the right and click on "Connect". After a short moment, iRC displays the robot in the 3D view. The next time you connect, you can select it from the list of known robots without having to re-enter the address.
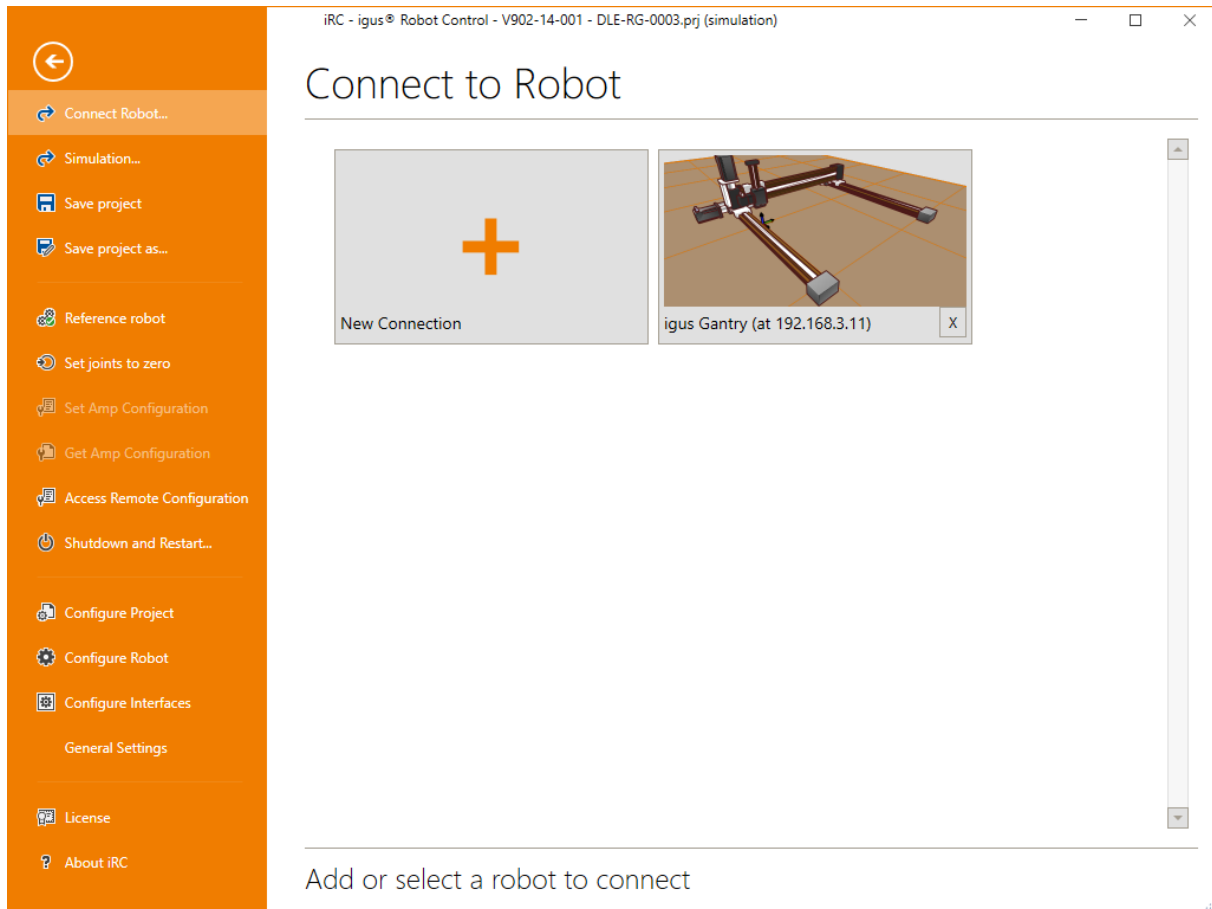
Figure 5: Adding or selecting a connection to a real robot

To disconnect, click "Disconnect" at the left of the ribbon above the 3D view. A simulation of the robot is then started and the button changes to "Connect". Another click reconnects to the previously connected real robot.

6.1.3   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.
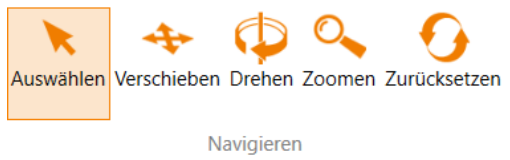
The function of the left mouse button can be changed in the "Scene" tab under "Navigate". The movement options "Select", "Move", "Rotate" and "Zoom" are available. "Reset" returns you to the home screen.

6.2   Connecting the robot

6.2.1   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. 6). Depending on the type of robot the axes must then be referenced.
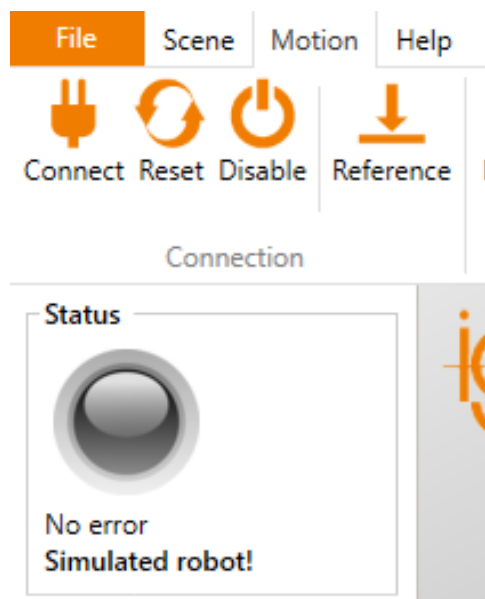
Figure 6: Buttons for connecting to the hardware, resetting errors and activating the motors, referencing and "Status" display.

1. "Connect": Establish the connection to the hardware.
   • This initializes the connection to the robot (usually via Ethernet, in some cases via USB CAN adapter).
   • The "Status" indicator on the left side changes from gray to red or green.
   • Error messages may be 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 6.3.

- The "Status" display becomes 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 joint errors are reset first, so it is not necessary to click "Reset" beforehand.
- The motors are then activated.
- If no errors have occurred, the "Status" display is now green.

6.2.2   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 section 6.4.

6.3   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.

6.3.1   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" (see fig. 6).

3. Click the "Reference" button (Fig. 6) 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 an axis, see Fig. 7. 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.
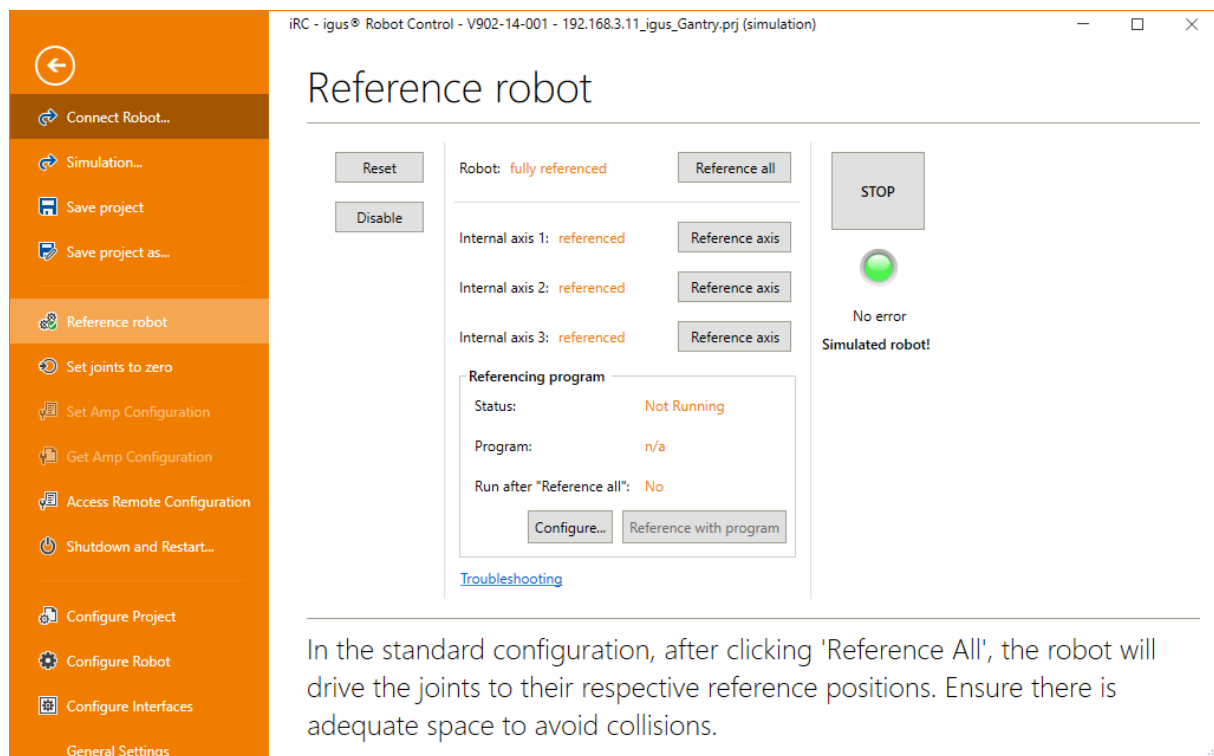
Figure 7: Referencing of the axes.

6.3.2   Referencing program

A referencing program can be defined to improve the precision of a robot referencing via absolute encoders. This is not relevant for robots that use referencing switches. First all axes are referenced normally, then the program is executed to move to a defined position at which the robot is then referenced again.

> As the axis positions before referencing are not reliable, it is not possible to program the referencing sequence, e.g. to avoid obstacles. If necessary, the order of the axes and the delays between the start of referencing individual axes can be adjusted in the robot configuration file so that certain axes can move out of the collision area first

You can find more on this topic in the section 10.2.4 and on our wiki:

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

6.4   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. 8):

- "Input": Connecting a gamepad
- "Motion parameters": Selection of movement modes and speed
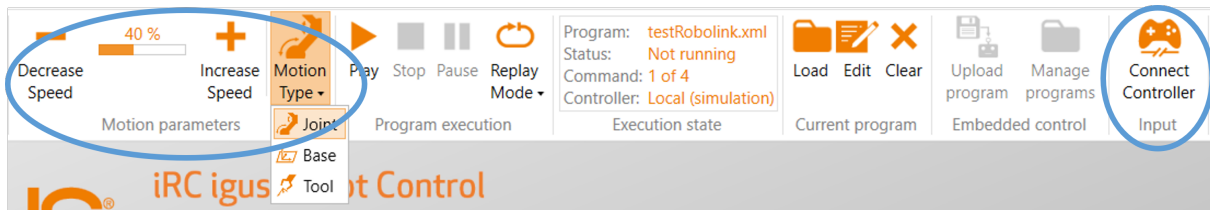
Figure 8: Control panels for moving the robot (highlighted in blue).

6.4.1   Gamepad

A gamepad is possibly the most intuitive way to move the robot. The fig. 9 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) The following button assignment is preset. The direction of the axes depends on the robot type.

Figure 9: Key assignment of the gamepad

1. - top: Change motion mode
   - bottom: Decrease speed

2. - top: 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.
   - bottom: Increase speed

3. - up: select previous statement in program editor
   - down: select next statement in program editor

4. - top: Touch up the current statement in the program editor
   - right: add axis movement to program instruction
   - bottom: Add linear motion program statement
   - left: Remove selected instruction in program editor

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

6.4.2 Software Buttons

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. 8):

- "Axis": Clicking on A1 to A6 moves the corresponding robot axis (if present). E1 - E3 moves the external joints. This may be a linear or a rotational axis (10).
- "Base" (Cartesian mode) moves the robot in straight lines along the X, Y, and Z axes of the currenly active coordinate system (see section 8.3).
- "Tool": (Cartesian mode) moves the robot in X, Y and Z of the current tool coordinate system.
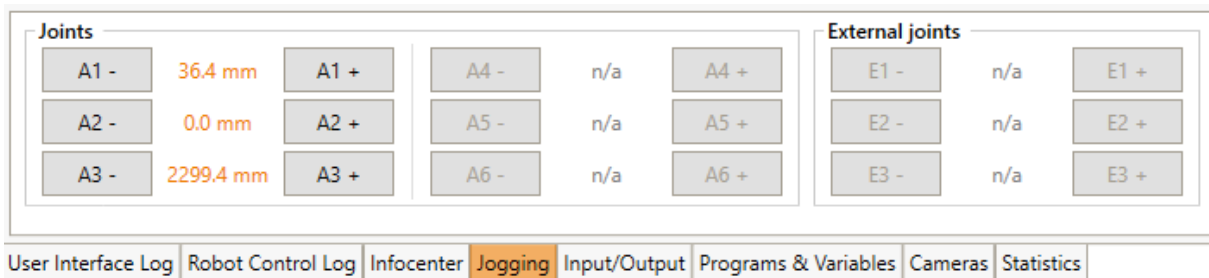
Figure 10: The software buttons for "axis" movements. In both Cartesian modes, the buttons change to X, Y, Z, A, B and C.

6.5   Unreachable Positions and Singularities

In axis mode each axis can move in its full range of motion (provided it does not collide). Some positions may not be reachable in cartesian mode depending on the kinematics. In some cases this is due to mathematical peculiarities, for example small movements with an extended robot arm can lead to fast axis movements or an ambiguous orientation - this is called singularity.

Among others, the following positions are not reachable:

- Out of Reach: the target position is too far away or cannot be reached by the kinematics. For robot arms, the motion stops before the arm is fully extended.
- Center Singularity: Occurs with robot arms when the arm is close to the center axis (rotation axis A1).
- Wrist Singularity: Occurs on 6-axis robot arms when the wrist (last 3 axes) is extended too far.

If the robot comes close to such a position in cartesian mode the motion will be stopped and a possibly running program will be interrupted. The status area on the left of iRC indicates the kinematic error.

To avoid singularities joint motion can be used instead of cartesian motion (e.g. linear command). This is especially useful for dynamic target positions, e.g. from a camera

6.6   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".
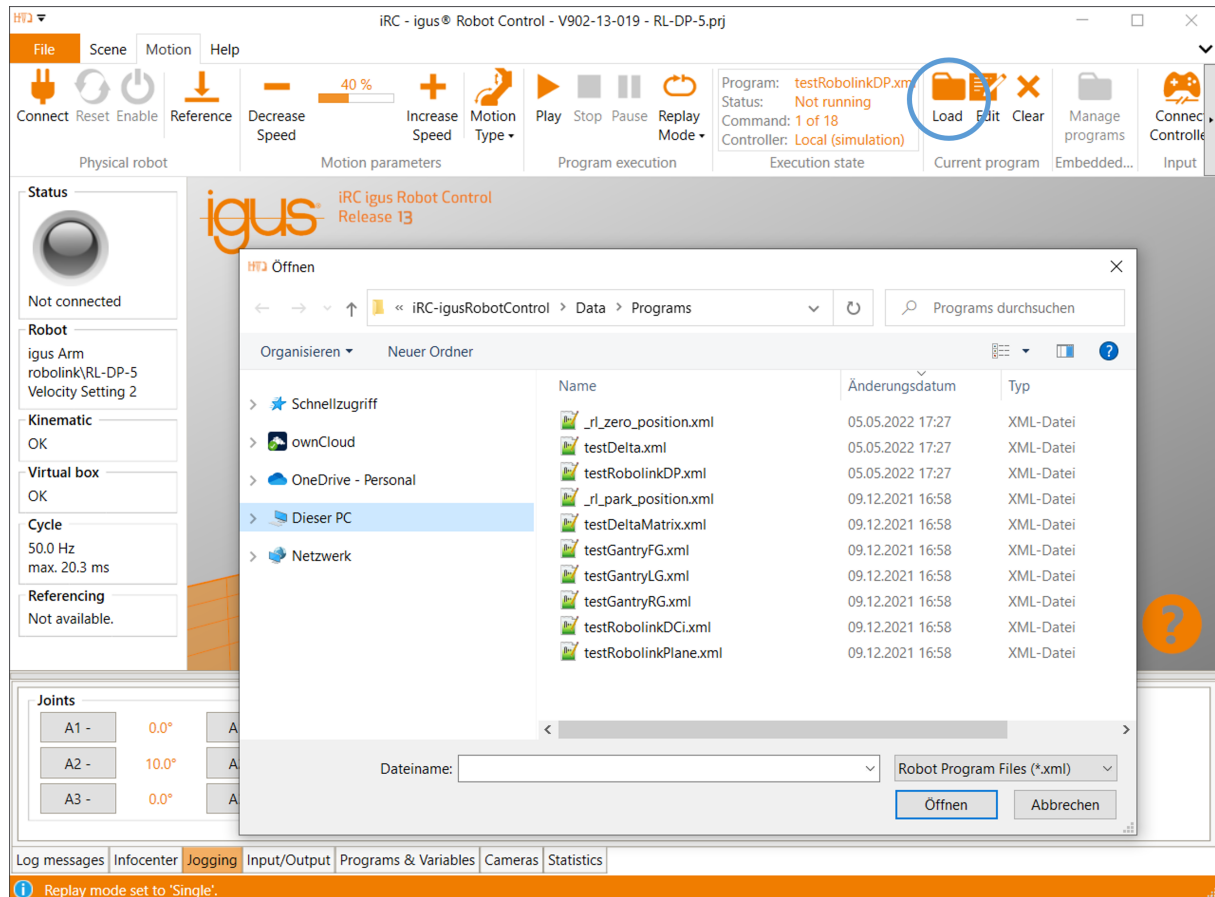
Figure 11: Loading a program (highlighted in blue).

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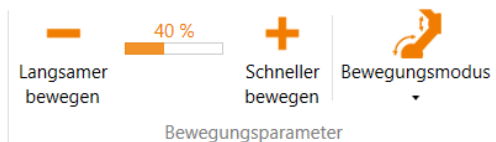
- Before starting an untested program, set the speed to e.g. 20%.
- Be especially attentive during the first complete program run and have the emergency stop key ready.

3. Start the program:
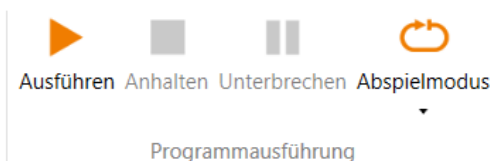
- Click the "Execute" icon in the "Program Execution" button group of the "Motion" tab.

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).

6.7   Digital inputs and outputs

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.
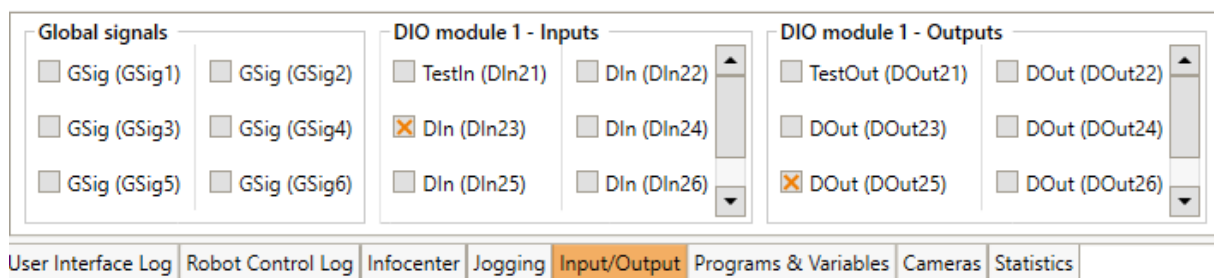
Figure 12: Input/output area of the iRC - igus Robot Control.

The configuration of the inputs and outputs is described in section 9.1.

6.8   Software interfaces

The robot controller provides various interfaces:
- PLC interface for control via the digital inputs and outputs. Especially for easy starting and stopping of programs via a PLC or pushbutton.
- Modbus TCP interface for control via a PLC or PC.
- CRI Ethernet interface for control and configuration via a PLC or PC. This interface offers the widest range of functions, but requires individual implementation.
- ROS interface for operating the robot via the Robot Operating System (`www.ros.org`).
- Interface for object detection cameras.
- Cloud interface for monitoring the robot state.
- App interface to expand the range of functions

See section 10.3 for the configuration of these interfaces.

6.8.1   App Interface

The app interface allows adding new functions to the robot control. The installation of apps is described in section 10.3.5. Apps can extend the graphical user interface, in which case you will find additional tabs above the 3D view. App functions can be integrated into robot programs, e.g. to access peripherals or perform more complex actions. This is described in section 7.10.

Apps and information on creating your own apps can be found under the following link

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

Since apps can execute arbitrary code on the robot controller, you should consider apps to be security-critical and only use apps from trustworthy sources. Errors can lead to damage, malicious apps can, for example, send data to third-party servers or attack the network. It is advisable not to connect robots with apps to the company network or the Internet.

6.9   Updating 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. C:\iRC-igusRobotControl) 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.

7 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.

7.1 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 of iRC.
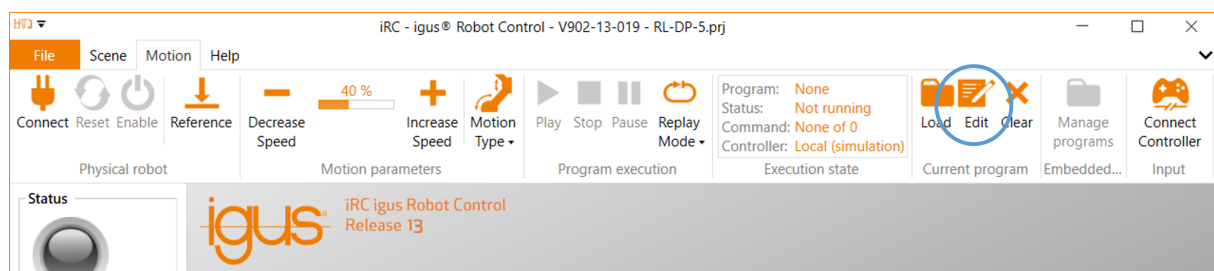
Figure 13: The program editor is opened by clicking "Edit".

The following window opens, here with a short program:
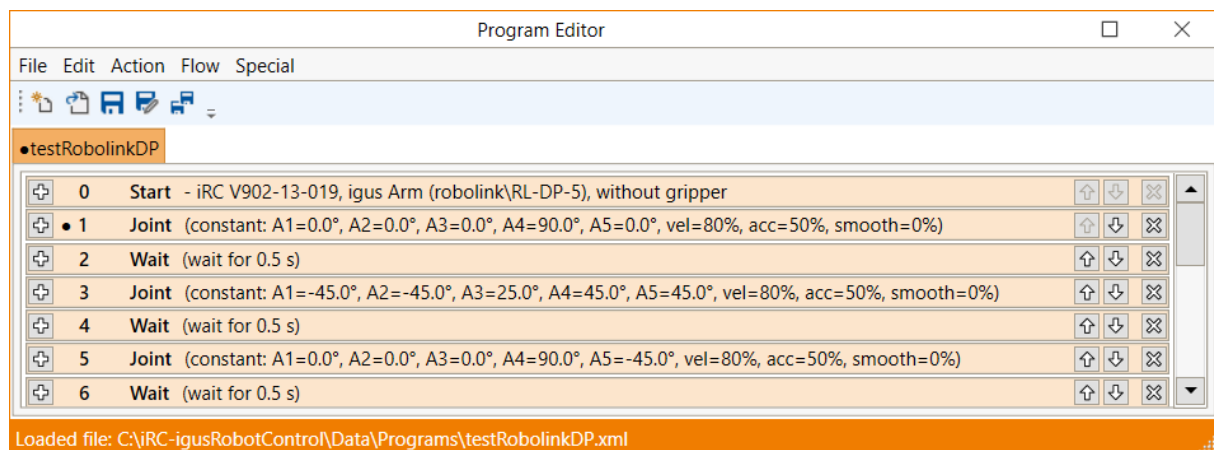
Figure 14: Program editor with a shord programm

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

7.1.1 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. 15).
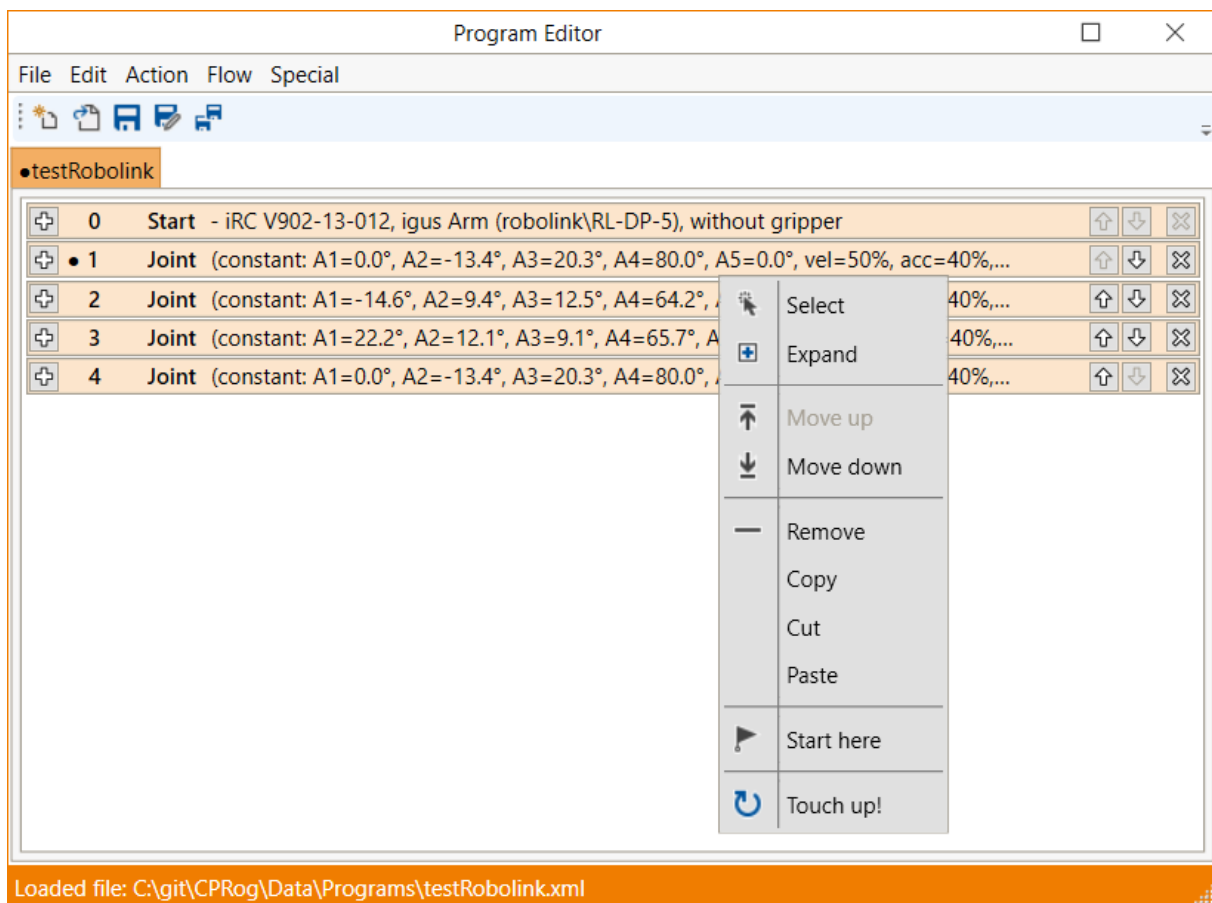
Figure 15: The context menu of a command

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.

7.1.2   Touching Up Positions

Certain commands require position values as parameters. It is often desirable to use the current position of the robot with respect to the current frame of reference. 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. 15).

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

7.1.3   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.
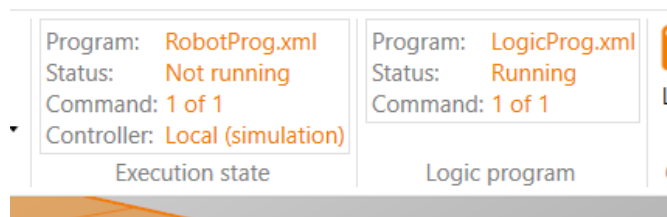
Figure 16: The ribbon at the top of iRC shows the state of the robot and logic program
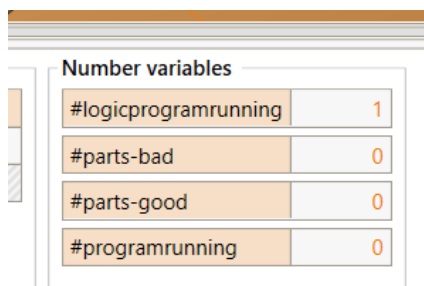
Figure 17: Using the variable #logicprogramrunning the robot program can check if the logic program is running

To select a specific command as the starting point for execution, click on "Start Here" in the context menu (see Fig. 15).

7.2   Robot and Logic Programs

Within a robot program, instructions are always executed one after the other. So it is not possible, for example, to switch a digital output or perform calculations during a movement. If this is necessary a logic program can be used.

A logic program is created like a robot program via the program editor, but must not contain any movement instructions. After it is assigned, it is repeated permanently, even if the actual robot program is not running. To load a program as a logic program, it needs to be transferred to the robot controller first by loading it once like a normal program. After that it can be assigned in the configuration area "File" → "Configure Project" → "Program" → "File logic".

> Programs have a minimum execution time of 500ms. Shorter programs, for example fast running logic programs, may be delayed during the automatic restart. This is especially important if the logic program is used to switch digital outputs or to evaluate the states of the inputs. If a fast repetition of the program is necessary a continuous loop (condition "False") can be created which contains all other instructions of the program

Example: To apply glue a valve is to be opened during movement.

1. Define the motion in the robot program.

2. Before the movement starts, set a global signal (GSig, instruction "Digital output") in the robot program to signal the logic program that it should soon switch the digital output for the valve.

3. Create a logic program with an infinite loop (loop with condition "False"). See info box above.

4. In loop of the logic program, create a condition statement ("IF"), define its condition so that the global signal must be active and the target position is reached. For example "GSig1 and #position.x > 150" if the output should be activated from X=150mm.

5. In the first branch of the condition command set the digital output to activate the valve.

6. Also reset the global signal in the first branch so that the condition is not called again in the next run. In more complex applications the logic program could also set a second global signal to close the valve again after reaching the end target position.

7.3   Comments and Information within the Program

7.3.1   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.
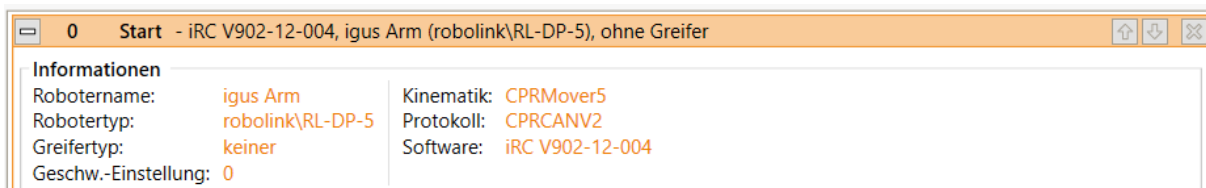
Figure 18: The Start line contains information about the current hardware.

7.3.2   Descriptions

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

7.3.3   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".

7.4   Motion

7.4.1   Abort Conditions

Each motion command can be provided with a abort condition. It is a conditional expression that follows the syntax described in section 7.7.1. 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.

7.4.2   Acceleration and Smoothing

To prevent abrupt movements an axis acceleration (percentage of maximum acceleration) and a smoothing factor can be specified for each motion command. With a smoothing factor of 1-100%, the motion instruction is smoothed with the following instruction, so that, for example, several linear movements form smooth curves instead of stopping at each target point and starting again.
Smoothing is only possible with immediately following motion commands of the same type. E.g. linear and circular movements can be smoothed with each other and axis movements with themselves. If a motion sequence is interrupted by a motion of a different type or by a logic instruction, the smoothing is also interrupted and the robot stops briefly. The maximum number of commands that can be smoothed is limited to 5-20, depending on the robot, in order to prevent calculation pauses in the program sequence. In case of long paths this is shown by regular braking and restarting.

> If your application requires long uninterrupted paths you can increase the limit. Note that this can lead to a short calculation pause before starting a very long paths. More information can be found on our wiki under the keyword "LookAhead".
>
> ```
> https://wiki.cpr-robots.com/index.php/Motion_Smo
> othing
> ```
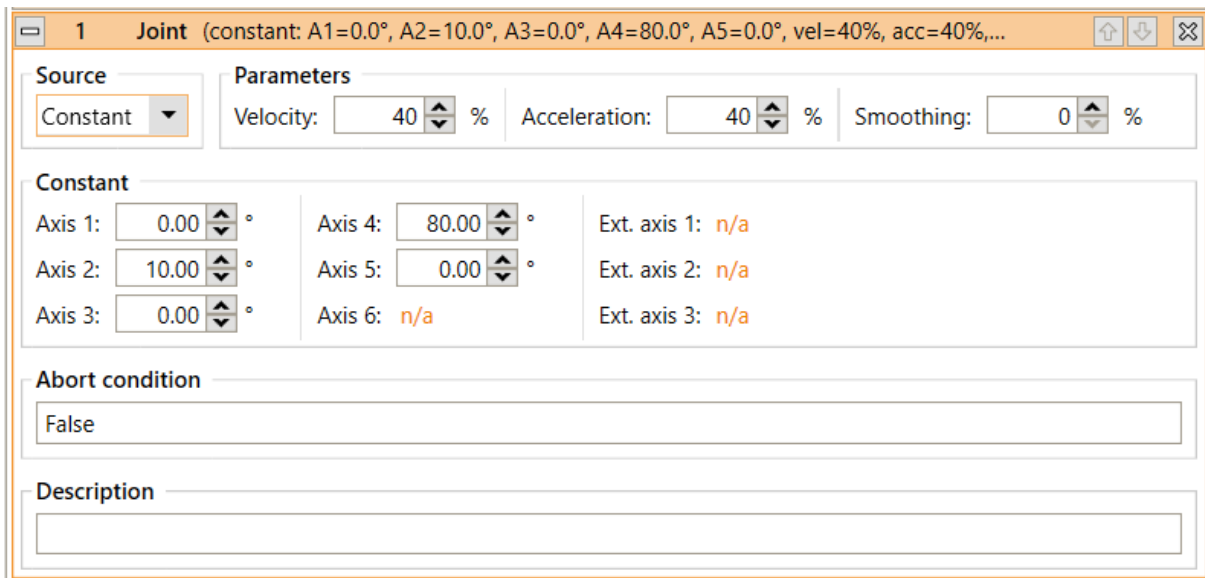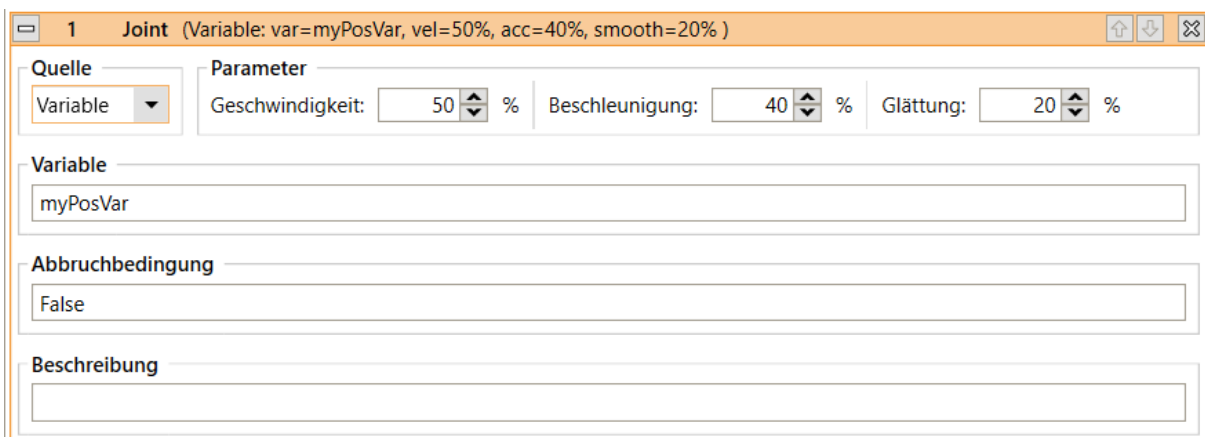
7.4.3   Joint Motion

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.

- "Variable": The target position is taken from the position variable specified in "Variable".
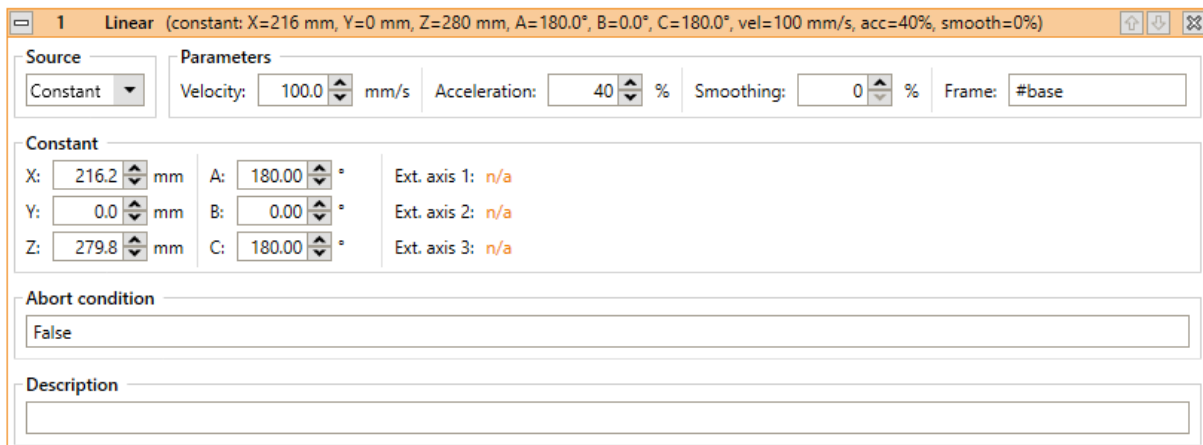
The movement speed is indicated by "speed". It is measured in percent of the maximum allowed motion speed for the respective robot axes.

The joint command can be called in the program editor under the menu items "Action" → "AxisMotion" and "Action" → "VariableMotion" → "AxisMotion".
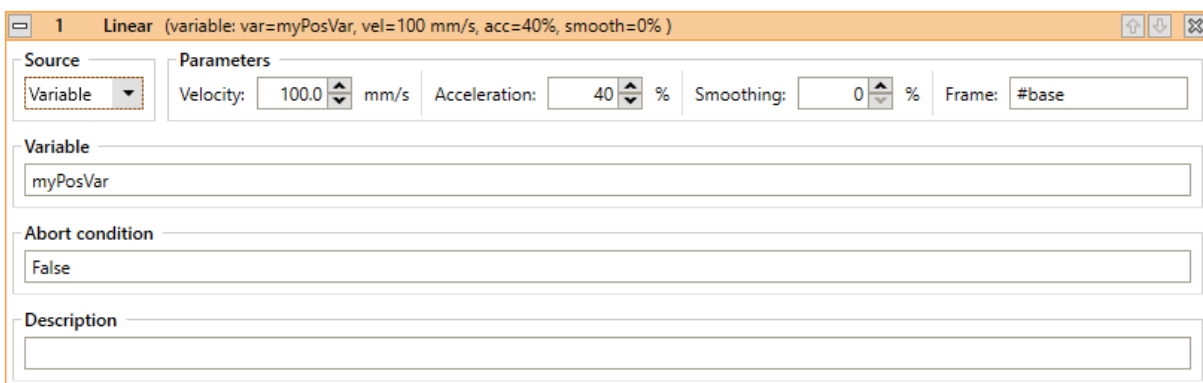
7.4.4   Linear Motion

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.

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

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".

Note that the coordinates of the target position is always with respect to a specified userframe, see section 8 for futher information on user-defined frames of reference.

7.4.5   Joint Motion to Cartesian Position

A linear motion to a position in cartesian space (XYZ) is not always suitable or possible. For example, if the motion should be as fast as possible, but the path does not need to be linear or if a singularity must be passed through on the way there. The "Joint to cart. motion" command can be used for this purpose. Unlike the normal joint motion, it is given a cartesian position. At the start of the motion, the position is converted into the target axis angle and moved to as with the normal joint motion command. The robot moves as fast as the slowest joint allows.

7.4.6   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".
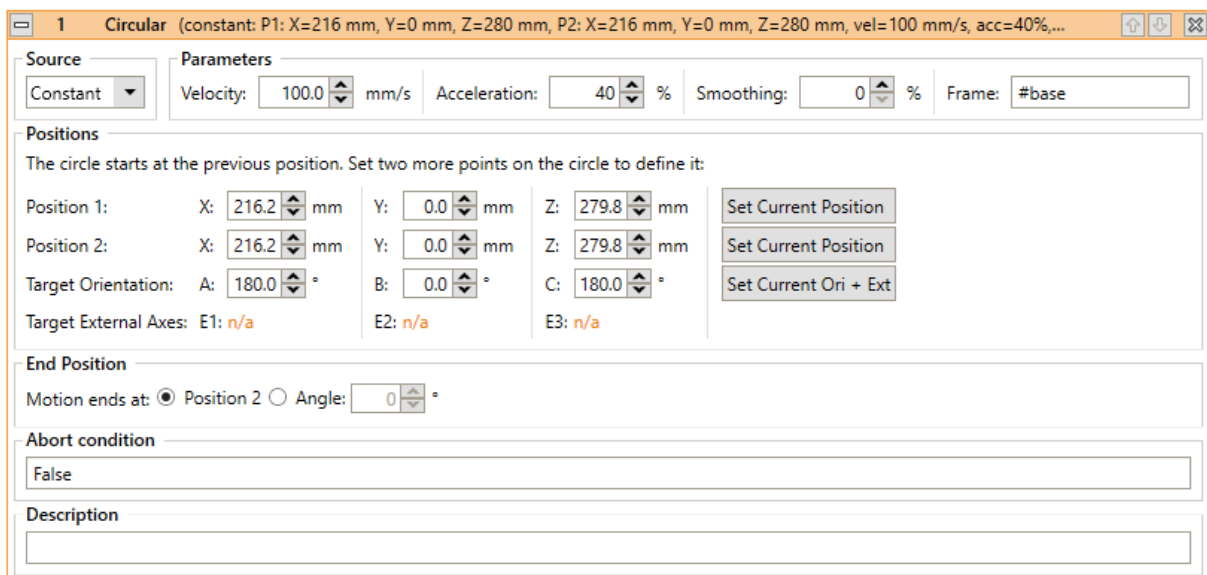
Under "Type" the following modes of relative movement can be selected:
> "Joint": The relative offset is specified in axis coordinates. The motion speed is specified by "Speed" in percent of the maximum allowed motion speed for the respective robot axes.

- "Linear - Base": A linear motion is performed with an offset specified in Cartesian coordinates. The coordinate system used for the offset is the specified frame of reference (see section 8 for futher information on user-defined frames of reference). 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.

7.4.7   Circular Motion

The instruction "Circular Motion" enables movements along a full or partial circular path. It is compatible with linear movements, so that the transition from and to linear movements can be smoothed.

The circular path is defined by three points on the circle, as shown in figure 19. The starting point is defined by the target position of the previous motion command. Position 1 and position 2 are arbitrary points on the circle. The parameter "Motion ends at" defines whether the circle is left at point 2 or after a certain angle. The angle can be before or after point 2. It is also possible to circle several times (angle greater than 360°) or move in the opposite direction (negative angle).
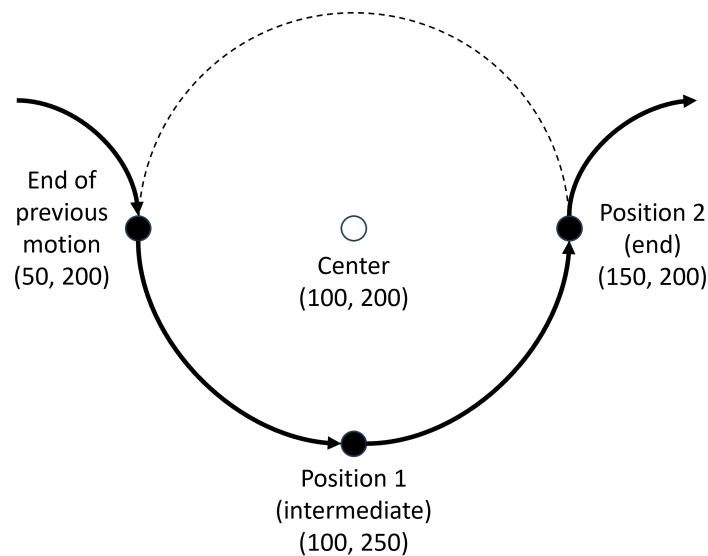
Figure 19: Example of a circle definition. In the plane, the three circle positions can be easily calculated by adding or subtracting the radius (here 50mm) to the center point.

⚠️ The circular motion must not be the first motion instruction in the robot program. If the program starts at an unexpected position, the circle may become larger or smaller than expected.

💡 **Programming a circular motion by calculating the circle points**
For simple circular movements in a plane, it is recommended to calculate the coordinates of the circle center point and add or subtract the radius along the Cartesian axes to calculate the start, intermediate and target position as shown in figure 19. Make a sketch if necessary.
Even if you do not want to move 180°, it is often easiest to use positions 1 and 2 at 0°, 90°, 180° or 270° and to specify the actual angle to be moved using the end of movement parameter. This allows the positions to be calculated as described above without trigonometry.

💡 **Programming a curve path by Teach-In**

1. Move the robot to the start position of the curve and add a linear instruction.

2. Move the robot to any position on the curve, e.g. half way. Add the circle instruction.

3. Move the robot to the end of the curve and click the "Set current position" button in the "Position 2" line.

4. Optional: click the "Set Current Ori + Ext" button if the tool orientation or external axes change during the curve path.

If the tool is to be rotated during the circular movement, the target orientation can be specified. The angles from the start orientation to the target orientation are interpolated linearly. If specified, the positions of the external axes are also calculated.

The positions can be defined as constants via two position variables and are with respect to the given frame of reference (see section 8 for futher information on user-defined frames of reference). In this case, the target orientation and the external axes must be specified in the variable for position 2 (target position).

7.4.8   Set Velocity

The instruction "Set Velocity" instructs an external axis to move at a constant velocity, for example for a conveyor belt. The movement stops only when a movement of 0 units/s is assigned or the program is stopped. This function only affects external axes configured for velocity mode.

7.5   User Frames

As explained in chapter 8, user-defined reference frames (user frames) allow reusing motion sequences on several differently aligned objects. User frames move the origin (0 position), and rotate the XYZ axes.

The motion commands with cartesian target positions contain the parameter "Frame", which specifies the frame to be used. In addition to the user frames, the system-defined frames "#base" and "#tool" are available. "#base" is the standard coordinate system of the robot with the zero point usually at its base. "#tool" is relative to the gripping point of the tool.

7.5.1   Copy User Frame

It is often useful to switch between frames in the program sequence. For example, a motion sequence can be defined with a placeholder frame in a sub program. Before calling the sub program, select the user frame to be used. The "Copy user frame" command copies an existing coordinate system into the placeholder.

The command can be added via the "Special" menu. It contains the parameters "Source" and "Destination". The coordinate system specified at Source is copied to the coordinate system specified at Destination. If it does not yet exist, it is created. Attention! This can overwrite existing coordinate systems!

7.6   Gripper and Digital In-/Outputs

7.6.1   Digital Inputs

The states of the digital inputs can be used in conditions (see section 7.7.1). The first digital input of the first digital I/O module has the number 21 and can be used in conditions via the keyword DIn21.

7.6.2   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".

7.6.3 Global Signals

Global signals are internal flags that can be set in the robot program like digital outputs and evaluated like digital inputs. They can be used for example to store simple state information or to communicate between robot program and logic program. Since global signals can also be read and set via the CRI and Modbus interfaces, robot programs can also use them to communicate with external applications or a PLC. The concept of global signals is similar to coils in Modbus or a boolean in high-level language programming.

In conditions in the robot program, the GSig keyword can be used to query the state of the global signal. For example GSig1 for the first signal. There are 100 global signals available.

7.6.4 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%.

7.7 Program Flow

7.7.1 Conditions

Conditions can be used in if-then-else instructions, loops, and as termination conditions in motion instructions. The conditions can be combinations of digital inputs, global signals, boolean operations and comparisons. Capitalization and spaces between symbols are ignored.

In the simplest case, for example, a condition can check whether a signal is present at digital input 21:

> DIn21

More complex conditions can be constructed using the AND and OR keywords and parentheses. The following condition is met if a signal is present at either input 21 or inputs 22 and 23:

> DIn21 OR (DIn22 AND DIn23).

To negate an expression put an exclamation mark (!) in front of it. This is also possible before parenthesized expressions. The following condition is met if either there is no signal at input 21 or if there is no signal at inputs 22 and 23 together:

> !DIn21 OR !(DIn22 AND DIn23)

Likewise, the state of global signals (see section 7.6.3) can be queried. Global signals are internal flags which can also be used for communication between robot and logic program and external applications or PLC:

> GSig1 AND !DIn21

> The states of the digital outputs cannot be queried in conditions. If necessary, a global signal representing the output can be set after setting the output.

In addition, the values of number and position variables can also be checked. Number variables represent a single number while position variables contain several number components. For position variables, it is therefore always necessary to specify which component is to be compared.

```
mynumbervariable = 5
mynumbervariable < 10
mypositionvariable >= 42
mypositionvariable.X = 123
mypositionsvariable.B >= 90
mypositionvariable.A3 > 300
mypositionvariable.E1 < 500
```

Numeric values can also be compared to each other:

```
mypositionvariable.X > mynumbervariable
mypositionvariable.A1 <= otherpositionvariable.A1
```

The following position components can be used to compare positions:
- cartesian
  - X, Y, Z - position in millimeters
  - A, B, C - orientation in degrees
- axis positions
  - A1 to A6 - robot axes in degrees or millimeters
  - E1 to E3 - additional axes in degrees, millimeters or self-defined unit

In summary, the condition syntax can be described 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	: = "=" | ">" | "<" | ">=" | "<="

7.7.2 Stop

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

7.7.3 Pause

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

7.7.4 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".

7.7.5 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 7.7.1. 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.
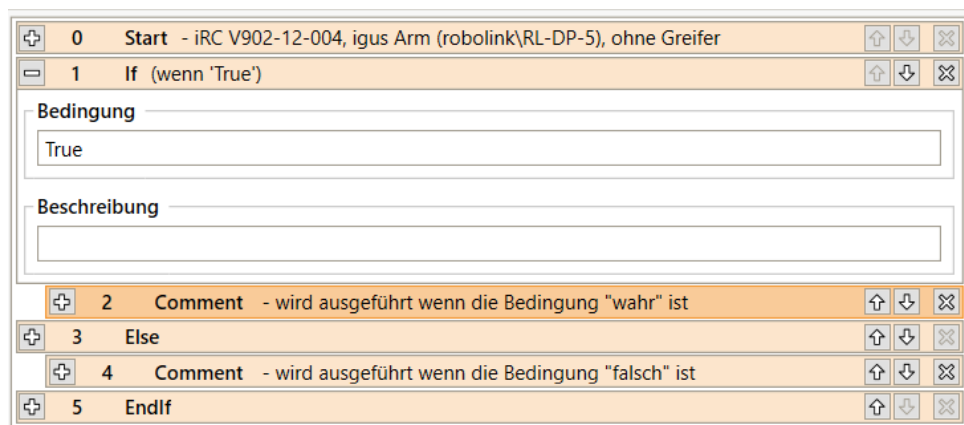
Figure 20: The If statement branches the program flow.

7.7.6 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 7.7.1.
- "counter": The loop will repeat the number of times specified in "repeats".
The loop command is accessible through the menu items "Flow" → "Loop".

7.7.7 Matrices / Palettizing

The matrix instructions calculate positions aligned to a grid, e.g. as gripping or depositing position for palletizing tasks. Figure 21 shows a motion pattern that can be executed by using the raster instructions.
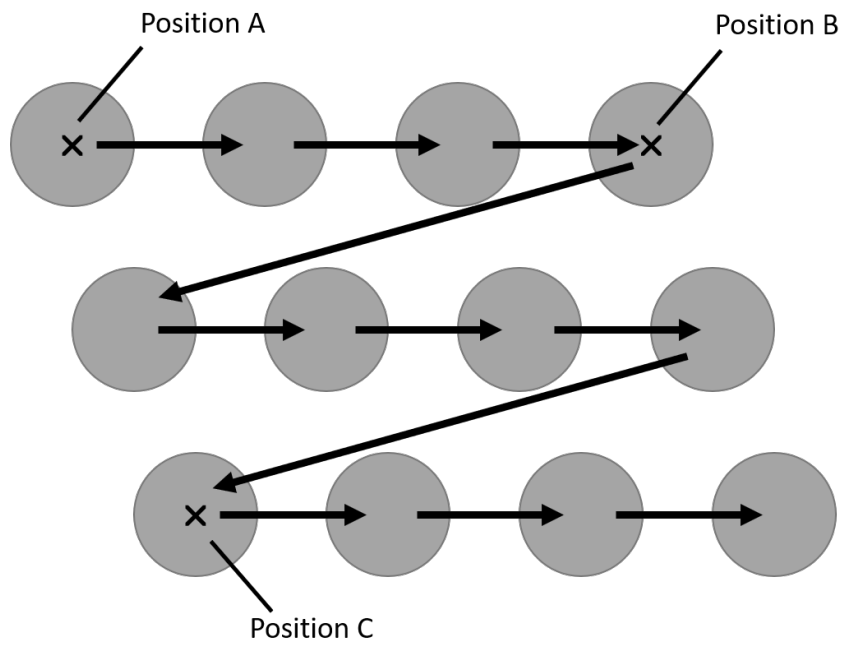
Figure 21: The matrix movement is from point A to B, then offset in direction C
.

iRC provides two approaches to program this:
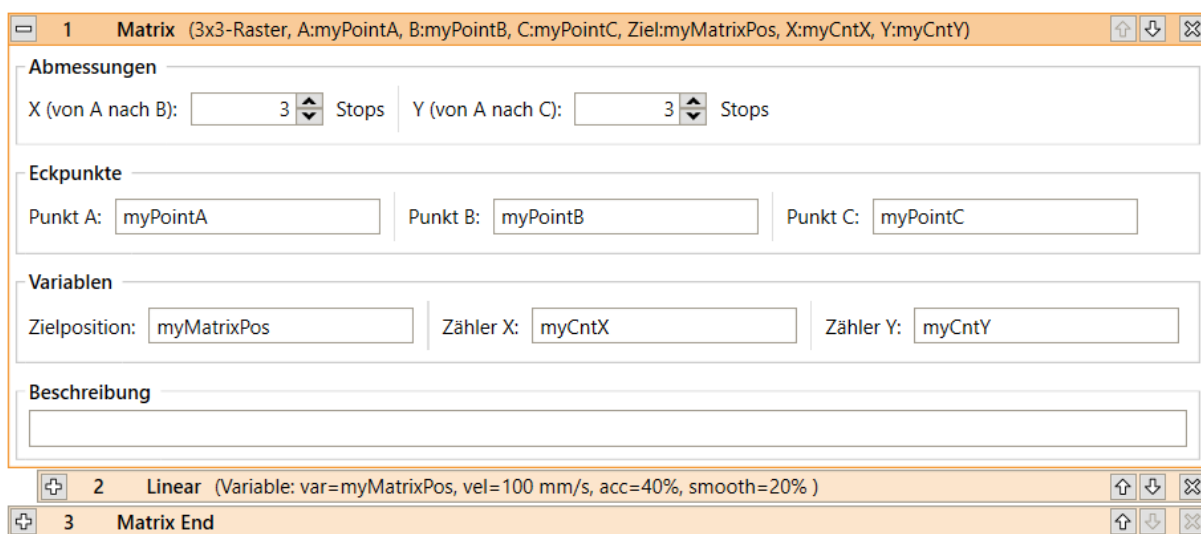
Figure 22: Definition of a matrix loop

Statement type	Description
Matrix loop	Executes a statement block for each raster position.
Matrix definition and raster query	Allows to calculate arbitrary raster positions based on an index variable.

The matrix loop is suitable for simple use cases where the positions are processed strictly in their order. The loop starts with the first position and is left after the last position. The matrix definition and query is more flexible, several grids can be used at the same time and the positions can be retrieved in any order, but the index variable must be counted by additional logic. This allows e.g. to start with a partially filled palette or to skip positions.

The matrix loop can be added via the menu item "Program flow" → "Loop" → "Raster". Matrix definition and matrix query are located in the menu "Special commands" → "Raster".

The position variables specified as "Point A", "Point B" and "Point C" define the corners of the area covered by the raster loop (see figure 21). The number of steps to be taken is specified 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.

Figure 22 shows a matrix loop. 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.

The matrix definition and query instructions (see fig. 23) use the same parameters: The dimensions and positions are specified in the raster definition, and the index and output variables are specified in the raster query. Different rasters are identified by a name.

7.7.8 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".

⊞ • 1	**Matrix Definition** (name=myMatrix, dimensions=3x3, A=myPointA, B=myPointB, C=myPointC)	⬆ ⬇ ✖
⊞ 2	**Loop** (repeat until 'myIndex < 10')	⬆ ⬇ ✖
⊞ 3	**Matrix Query** (name=myMatrix index=myIndex, target=myMatrixPos, X=myCntX, Y=myCntY)	⬆ ⬇ ✖
⊞ 4	**Math** (add 1 to myindex)	⬆ ⬇ ✖
⊞ 5	**Linear** (variable: var=mymatrixpos, vel=100 mm/s, acc=40%, smooth=0%)	⬆ ⬇ ✖
⊞ 6	**EndLoop**	⬆ ⬇ ✖
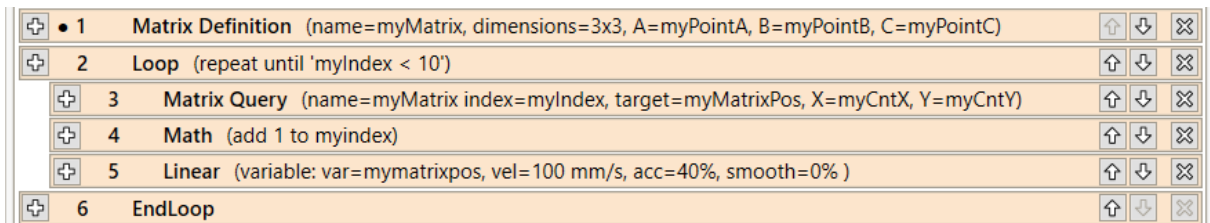
Figure 23: Usage of matrix definition and matrix query commands. The definition of variables is omitted.
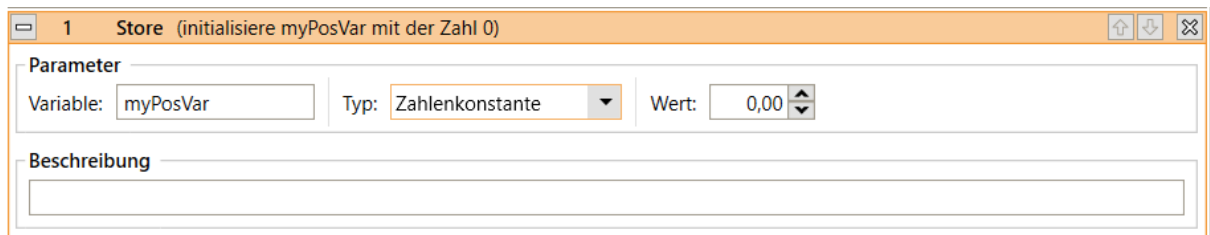
Figure 24: Definition of a number variable.

7.8 Variables and Variable Access

Robot programs 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.
  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.

7.8.1 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 24).
- "PositionConstant":
  A position variable is initialized with the constants specified in "Cartesian Position", "Joint Position" and "External Joints" (see fig. 25). 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.

7.8.2 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.
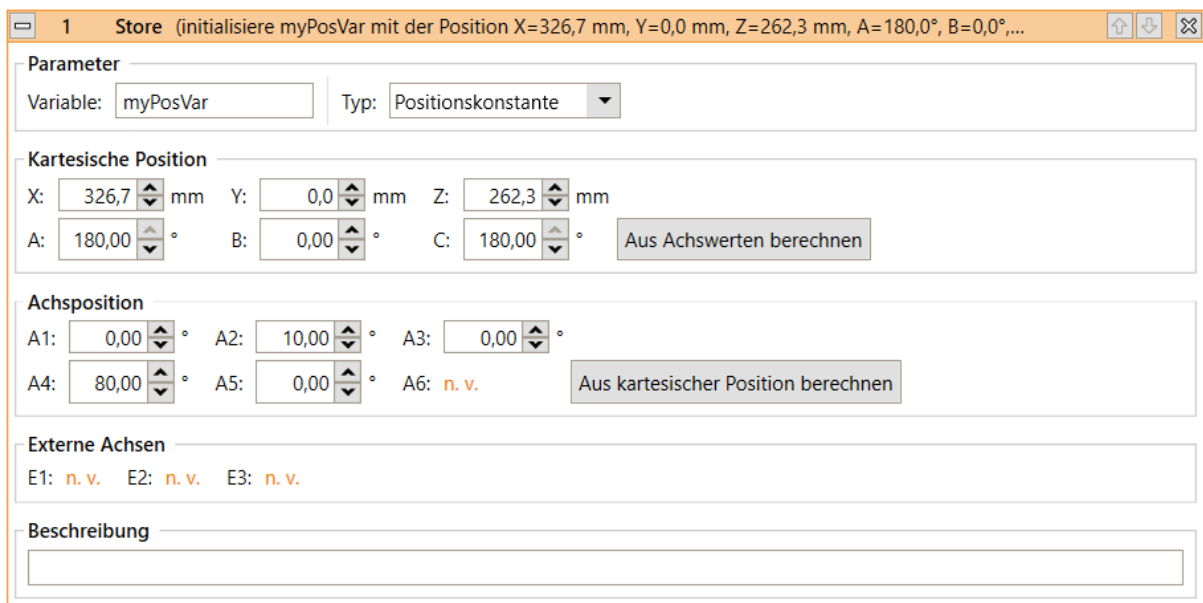
Figure 25: Definition of a position variable.

- #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 "#"

7.8.3   Accessing Elements

Position variables contain the following elements:
- Position: x, y, z
- Orientation: a, b, c
- 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".

7.8.4   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	Multiplication	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						

7.8.5   Observing Variables

You can observe the current values of all defined variables in iRC in the "Programs and Variables" tab in the status area.
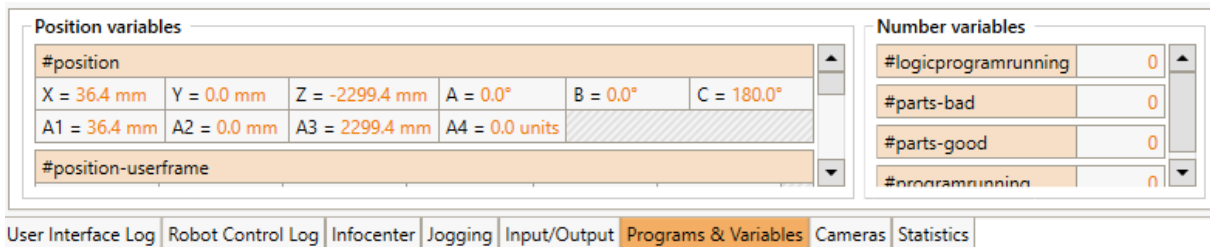
Figure 26: The values of the variables are displayed in the info area.

7.9   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 10.3.6). The program instruction can be added via the menu item "Special Commands" → "Camera".
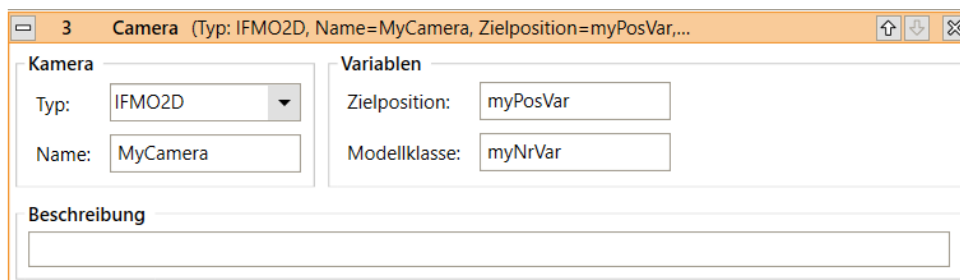
Figure 27: Camera command in the program editor.

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.

7.10   App Function

The app command enables the integration of app functions into the program flow. If at least one app that defines functions is active, it can be selected in the command. If the app defines function parameters, these can then be specified in the instruction.

When the robot reaches the instruction, it sends the function call including parameters to the app and waits until the app confirms completion. In the meantime, the app can, for example, access data from the robot or connected peripherals, perform calculations, log data or send it to a server. It can also set variables or global signals to interact with the following program instructions.

8 Frames of reference

When the robot is moved in cartesian mode (see 6.4.2) all coordinates describing the motion are with respect to a specific frame of reference. iRC allows to change the frame of reference that is used. Two predefined frames of reference ("#base" and "#tool") are provided and additional user-defined frames of reference (Userframes) can be chosen. All frames of reference used in iRC are orthonormal.

8.1 Predefined frames of reference

The predefined frames of reference are always available. They are marked by a # character as first letter of their names. They can neither be changed nor deleted.

8.1.1 The base frame

The base frame of reference is named "#base". It is the "natural" reference frame of the robot. Its exact position depends on the type of robot, typically its origin is at the base of the robot.

8.1.2 The tool frame

The tool frame ("#tool") has its origin at the TCP of the robot. Therefore this frame of reference will always be moving with the tool of the robot.

8.2 User-defined frames of reference

User-defined frames of reference (short "userframes") provide a way to move the robot relative to a frame of reference that has been provided by the user. A userframe is uniquely defined by providing three points in space (A, B and C):

- Point A designates the origin of the reference frame.
- Point B always lies in positive x direction from the origin and thereby defines the direction of the x axis which always points from A towards B.
- Point C always lies in positive y direction of the origin. The y axis always lies inside of the plane spanned by the x axis and point C and is perpendicular to the x axis. The direction of the y axis is uniquely defined by those properties.
- The direction of the z axis is defined as the (unique) direction which is perpendicular to both, the y and x axes such that a right-handed coordinate system is created.

8.2.1 Creating a new userframe

The first step in the creation of a new userframe consists of starting the userframe-wizard via the project configuration section. See 10.2.6. When the wizard has been startet follow these steps:

1. Move the robot into a pose, such that the TCP is located where point A is to be defined and then click on "use current". The chosen coordinates will now be displayed next to "Point A (origin)".

2. Move the robot into a pose, such that the TCP is located where point B is to be defined and then click on "use current". The chosen coordinates will now be displayed next to "Point B (pos. x from origin)".

3. Move the robot into a pose, such that the TCP is located where point C is to be defined and then click on "use current". The chosen coordinates will now be displayed next to "Point C (pos. y from origin)".
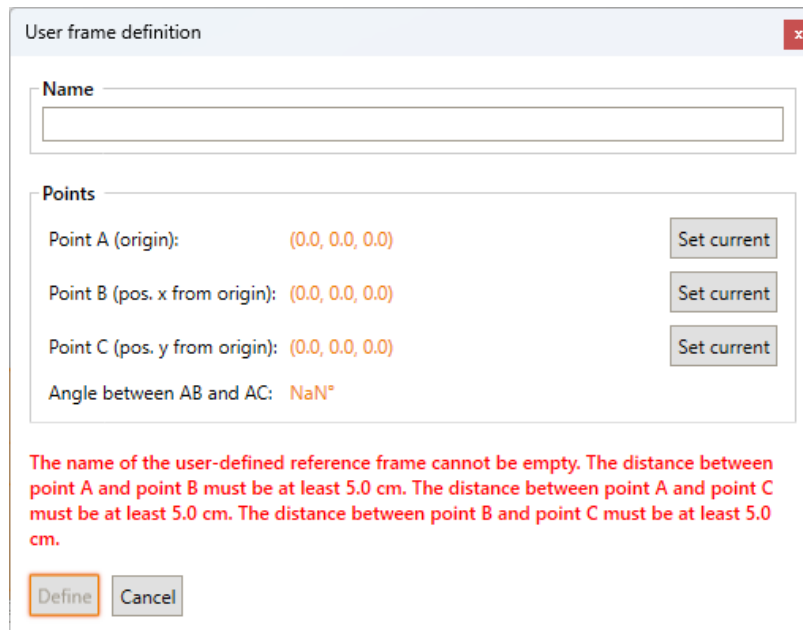
Figure 28: The userframe creation wizard.

4. Enter the name of the new userframe into the textbox under "Name" and then click on "Define" in order to complete the creation of the new userframe.

> Please note that in iRC any frame of reference is orthonormal. It is usually not possible to define the points A, B and C such that they exactly define an orthogonal coordinate system. Because of this any newly defined userframe will be orthonormalized by iRC in a way that produces a frame of reference that is as fitting as possible to the given points A,B,C while being orthonormal.

8.3   Changing frames
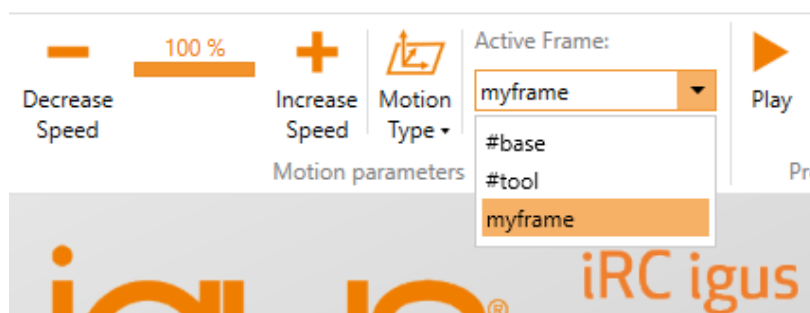
Figure 29: The dropdown-menu for frame selection.

The currently active frame of reference is shown in the toolbar of iRC (see figure 29) and can be changed via the dropdown-menu there. Please note that viewing and selecting frames is only available if at least one userframe hast been defined before.
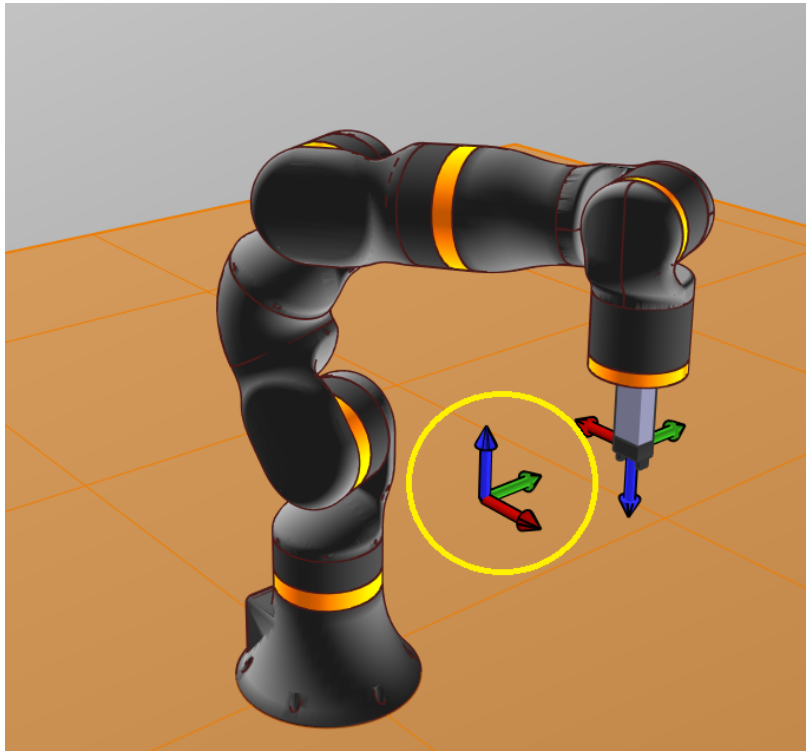
Figure 30: iRC shows the currently active frame of reference using a colored coordinate cross.

To jog the robot with respect to a userframe one hast to put iRC in cartesian mode first (see 6.4.2). In cartesian mode iRC will show the position and orientation of the currently selected frame of reference by drawing a colored coordinate cross (see figure 31).
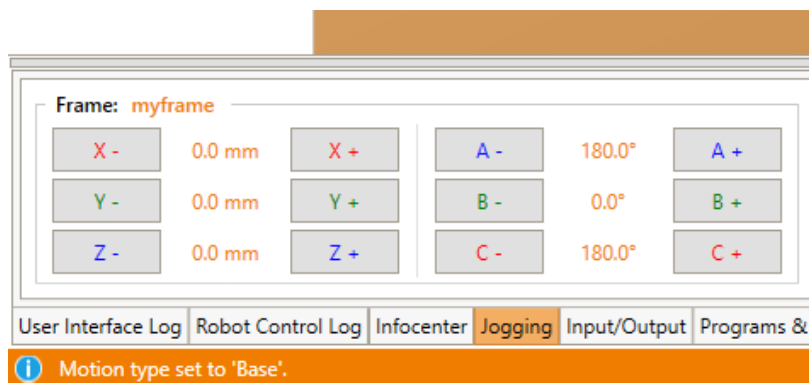
Figure 31: The software buttons for motion in cartesian mode.

In cartesian mode the jog buttons under the 3D view are showing the currently active frame of reference and the shown position is with respect to that frame.

9 Hardware Configuration

In order to use additional hardware such as switches, PLCs, actuators, axis brakes or additional axes, these must first be connected and configured. This chapter explains how to configure the hardware related to axes and input/output. Further settings concerning the behavior of the robot and the software interfaces (including cameras) can be found in chapter 10.

9.1 In-/Outputs

The ReBeL has digital inputs and outputs in the base and on the arm behind joint A4:

- Base: 7 digital inputs and outputs each. They are addressed by the names DIn21 to DIn27 or DOut21 to DOut27.
- Arm: 2 digital inputs and outputs each. They are addressed via the names DIn31 and DIn32 or DOut31 and DOut32.
- Global signals: These are virtual inputs/outputs. They can be set and read via robot programs, the CRI interface or the Modbus interface. Up to 100 global signals can be used.

The status of the DIO is visible via the DIN rail inputs/outputs tab, where the outputs can also be switched manually. The inputs and outputs can be configured in the project configuration area under "Inputs/Outputs".

9.1.1 Electrical Integration

The easiest way to connect switches, actuators or PLCs is via digital inputs and outputs. The ReBeL has two integrated DIO modules; 7 inputs and outputs are available at the base, 2 inputs and outputs each are available at the arm for tools. The socket on the arm is supplied with 6 or 8 poles, check which variant is installed on your robot.

The inputs and outputs are not galvanically isolated from the robot controller, the voltage is provided by the robot's power supply. A maximum of 500mA may flow through all inputs and outputs. Devices that require a different voltage or higher currents or where there is a risk of larger voltage or current fluctuations should be connected via relays, for example.
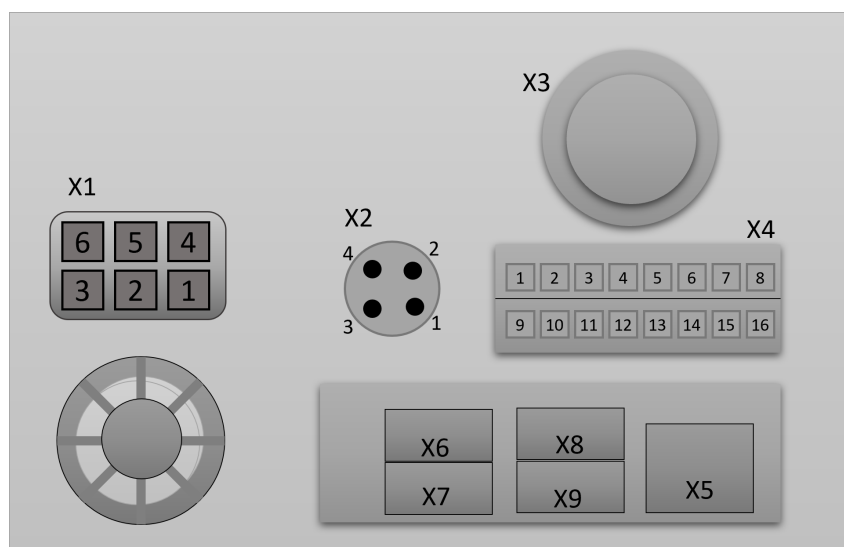
Figure 32: Switches, sockets and their pins at the base of the robot

Socket	Name	Plug
X1	Power supply	Molex 39012065
X2	Emergency stop	M8 4-pin female
X3	Soft on/off button with status light	
X4	Digital inputs	PhoenixContact 1844633
X5	Ethernet, default IP 192.168.3.11	
X6 - X9	USB sockets, not used	

Table 8: Names of the sockets and switches on the base (see fig. 32)

The status light at X3 can show the following states:
- Green: no error, axes enabled.
- Red: error or axes not enabled. The error code is displayed in iRC.
- Orange: integrated robot controller not yet connected (startup), please wait.

Socket	Pin	Description
X1	1-3	Power supply 24V
	4-6	Power supply GND
X2	1	Emergency stop CH1-Out (24V)
	2	Emergency stop CH1-In
	3	Emergency stop CH2-Out (24V)
	4	Emergency stop CH2-In
X4	1	24V for digital inputs
	2-8	Digital inputs DIn21 - DIn27
	9	GND for digital outputs
	10 - 16	Digital outputs DOut21 - DOut27

Table 9: Pin assignment of the sockets on the base (see fig. 32)

Figure 33: 8 pin DIO socket at the robot arm
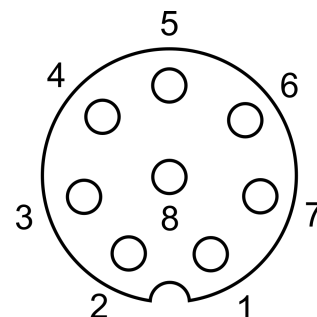
Figure 34: Pins of the 8 pin DIO socket at the arm

Pin	Description	Cable color
1	n.cv.	white
2	n.c.	brown
3	DIn32	green
4	DIn31	yellow
5	+24VDC (max 500mA)	gray
6	DOut32	pink
7	DOut31	blue
8	GND	red

Table 10: Assignment of the 8 pin DIO socket at the arm (see fig. 34)

Figure 35: 6 pin DIO socket at the robot arm
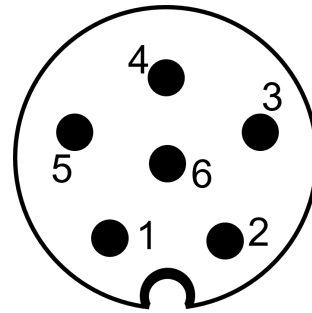
Figure 36: Pins of the 6 pin DIO socket at the arm

Pin	Description	Cable color (according to binder-Kabel)
1	+24VDC (max 500mA)	brown
2	DIn32	white
3	GND	blue
4	DIn31	black
5	DOut32	gray
6	DOut31	pink

Table 11: Assignment of the 6 pin DIO socket at the arm (see fig. 36)

9.1.2   Software Configuration

The software and integrated control are preconfigured for the two DIO modules of the ReBeL.
You can enter a descriptive name for each input and output. This is only relevant for the display, but
not as a name in program conditions. The reset state indicates which state an output changes to on
reset, the error state is assumed if an error occurs.

9.1.3   Connect sensors and buttons to the base

- The input signal (positive +24V) must be connected to an input pin X4 pins 2-8. X4 pin1 can be used as power supply e.g. for switches.
- The status of the inputs can be monitored in the register "Input/Output" below in iRC.
- A robot program can request inputs and react to them, e.g. with an if-then-else instruction

9.1.4   connect actuators to base

- The actuator (relay etc.) is supplied with +24V via a free DOut pins 10-16 if the DOut is set to true. X4 pin 9 can be used as GND.
- You can set the outputs manually in the register "Input / Output" below in iRC.
- A robot program can set the state of the outputs with the digital-out instruction.

9.1.5   connect sensors and buttons to the arm

- The sensor signal (positive +24V) must be connected to an input pin pin 2 or pin 4. Pin1 can be used as power supply e.g. for switches.
- The status of the inputs can be monitored in the register "Input/Output" below in iRC.
- A robot program can request inputs and react to them, e.g. with an if-then-else instruction

9.1.6   connect actuators to arm

- The actuator (relay etc.) is then supplied with +24V via a free DOut pin 5 or pin 6 provided the DOut is set to true. Pin 3 can be used as GND.
- You can set the outputs manually in the register "Input / Output" below in iRC.
- A robot program can set the state of the outputs with the digital-out instruction.

Translated with www.DeepL.com/Translator (free version)

9.2   Motor Brake

Electromagnetic brakes are installed in the ReBeL to prevent the axles from sagging. By applying a voltage, the axles are released. Without voltage, they fall into the braking state by springs.
The brakes of the ReBeL are controlled by the motor electronics, therefore no additional configuration is necessary. The settings in the configuration area of iRC have no influence.

9.3   Motor control configuration

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" → "Download firmware parameters" or "Upload firmware parameters". After downloading, the configuration set 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_P
arameter_Configuration
```

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

10 Software 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. 37). 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".
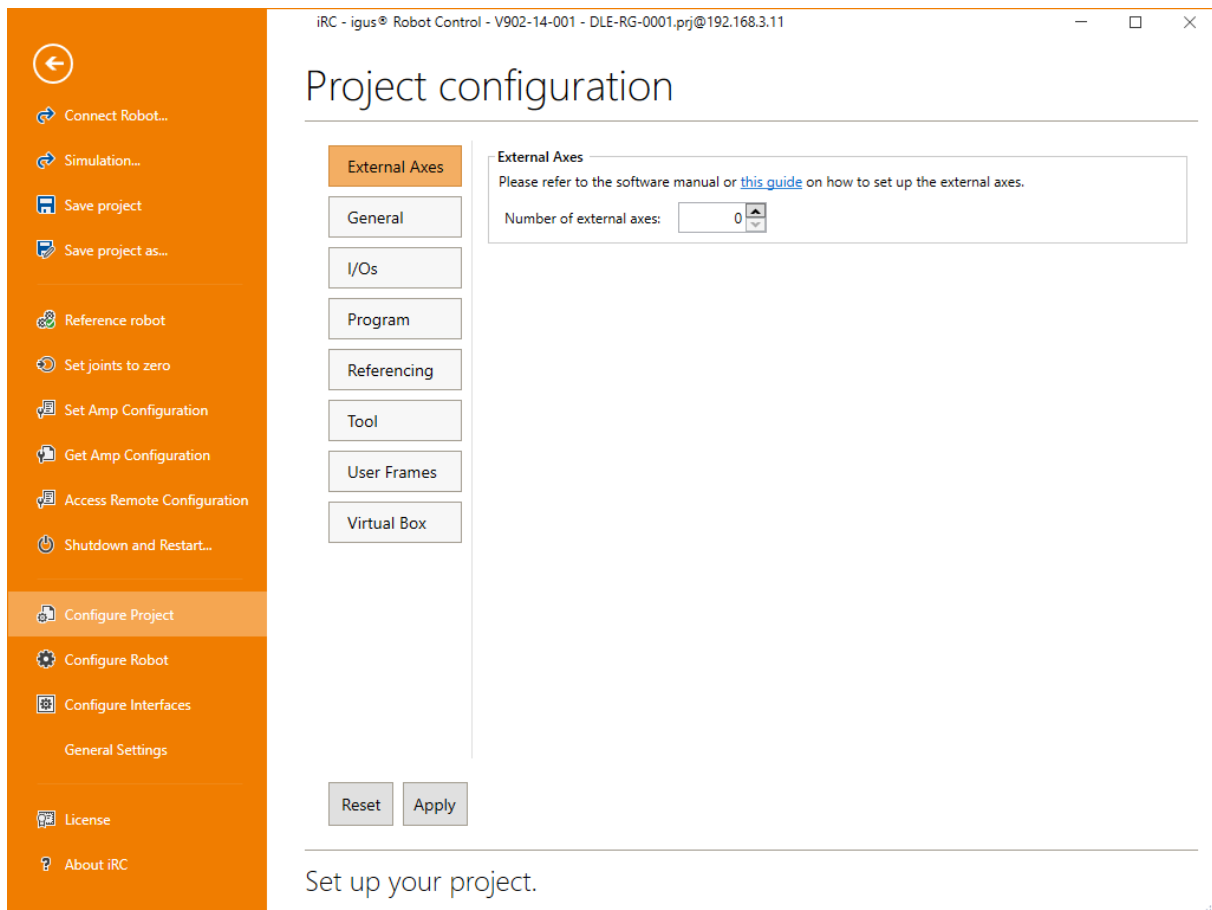
Figure 37: The project configuration area.

More specific settings can be made via the project, robot, and tool configuration files (see section 10.4). The settings of the axis modules can be accessed and changed via "Get/Set Amp Configuration" (see section 9.3).

> ⚠️ 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!

> ℹ️ Some changes to the integrated robot controller are only applied after a restart. Wait at least 30 seconds after the transfer and restart the robot.

10.1   General Settings

In the general settings, you will find robot-independent settings for iRC, such as the language of the user interface.

10.2   Project Configuration

10.2.1   General

Here you can name the robot and specify the author of the configuration. The robot name is used in the list of real robots to help you distinguish your robots. You can optionally enter the configuration author to indicate who configured the robot or who is responsible for it.

10.2.2   Inputs/Outputs

Here the number of input/output modules and the behavior of the inputs/outputs can be set. The basic inputs/outputs are only relevant for robots of the Mover series, all other robots are configured via the "DIN-Rail inputs/outputs" area. The global signals are internal flags for communication between robot programs and PLC.
For each digital output can be specified which state is assumed at reset and in case of error.

10.2.3   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.

10.2.4   Referencing

The referencing program is useful to improve the accuracy of robots that reference via an absolute encoder. Robots that reference via a referencing switch or by simply setting axis positions to 0 do not benefit from this. If a referencing program is defined, all axes are referenced first, then the program is executed to move to a user-defined position at which all axes are referenced again.

> The robot must already be referenced before the referencing program can be started. The referencing program cannot be used to define a referencing sequence or to avoid an obstacle between referencing multiple axes

> If you are using a referencing program, observe the referencing and have the emergency stop ready. The robot will move. If there is an error in the first referencing, it may move to an unexpected position

As a referencing program, simply create a normal robot program. In the simplest case it only contains one joint command. In order to compensate for gear play, a position should be chosen at which all axes are under a light load, i.e. the robot should not be in the candle position, for example.
Set the box "Run after Reference all" to run the referencing program during normal referencing (also via the PLC interface, Modbus or CRI). If the box is not set, it can only be executed via the button in the referencing area or via the corresponding CRI function.

10.2.5   Tool

The mounted tool can be defined here. Changing the tool requires reloading the project or restarting the integrated control.

New tools can be defined as configuration file in the directory "Data/Tools". New and changed tools are not automatically synchronized with the integrated control. To commit changes, open the "File" → "Tool Configuration" area, click the "Add" button in the "Tool Configuration" area and select the new tool configuration file.

10.2.6   User Frames

The list of user-defined frames of reference can be edited here. In order to define a new user-defined frame of reference, click on "Create" which will start the wizard for the creation of new user-defined frames of reference. Existing user-defined frames of reference can be deleted by selecting a frame from the list followed by a click on "Delete". You can find further information concerning user-defined frames of reference in section 8.
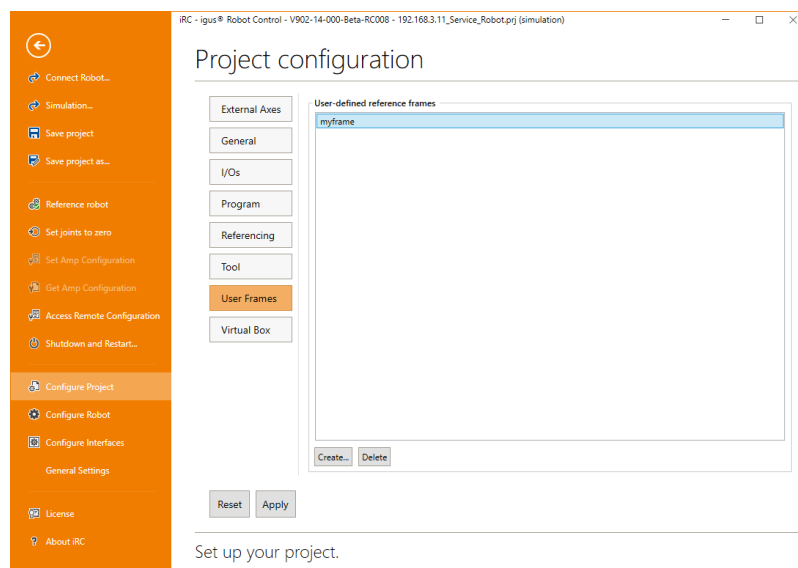
Figure 38: Userframes configuration.

10.2.7   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.

10.3   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.

10.3.1   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:

Parameter	Meaning
Enable	Executes Reset and Enable
Request reference	Starts the referencing of all axes
Play	Starts the execution of the loaded program if no program is running, continues a paused program or stops the execution of a running program
Pause	Pauses a running program or continues a paused program
Alt Start	One-button control: Executes the following functions in succession when the button is pressed several times: Reset, Activate, Reference (if not yet done), Start program
Alt Stop	One-button control: Executes the following functions in succession when the button is pressed several times: Pause, Stop, Reset
Shutdown	Shuts down the control computer
Start Platform Mission	Starts the mission of the mobile platform
Add joint command	Add-Axis statement to the current position in the program editor
Add linear command	Add-Linear statement to the current position in the program editor

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

The following output functions are supported:

Parameter	Meaning
No fault	Output is active when the robot is in fault-free state
Fault	Output is active when the robot is in fault state
Robot is referenced	Output is active when all axes are referenced
Program running	Output is active when a program is running
No program running	Output is active when no program is running
Platform mission running	The mobile platform is executing a mission

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".

10.3.2   Program Selection via Digital Inputs

Robot programs can be loaded and started via digital inputs or global signals. This is useful, for example, if a program is to be selected from a given selection via pushbuttons or the CRI-GSig instruction. The configuration for this can be found in the configuration area of the PLC interface in the section "Program Trigger".
The following parameters can be specified for each program trigger:

Parameter	Meaning
Active	Enables the program trigger
Type	Type of input signal: digital input (DIn) or global signal (GSig)
Number	Number of the input. E.g. 21 for DIn21 or GSig21
Program Type	Robot program or platform mission
Program File	Program or mission file. Must be in the programs or missions directory

10.3.3   Modbus

With the Modbus TCP interface, for example, PLCs can send data and instructions to the robot controller and receive status information. For more information on the use and licensing of this interface, see section 11.
The Modbus interface can be enabled through the interface configuration. Unlike the other configuration parameters, these settings are not mirrored with the connected system to prevent the configuration of the integrated controller and the control software on the Windows PC from colliding. For this reason, if a robot controller is connected, the parameters displayed refer to it; if none is connected, they refer to the PC software.
The following parameters can be configured:

Parameter	Meaning
Enabled	activates the Modbus server
Port	TCP port of the Modbus server
Max connections	Maximum number of simultaneous connections to the server (integrated control only)

10.3.4   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. From version 14 this interface is alway active at the robot and in simulation and does not require any configuration.
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`

> 💡 Note the active/passive system of the CRI: At any time only one CRI client is able to control the robot (active), all further clients can only observe its state (passive). This includes the graphic user interface of iRC. The first client that connects to the robot control becomes active. To manually become active an application can send the CRI command SetActive (see the CRI documentation), iRC becomes active by clicking "Reset"

10.3.5   App Interface

The app interface enables the robot control to be extended with your own or third-party software components. These components can access data from the controller and define functions that can be called from robot programs. It is also possible to integrate a graphical user interface in iRC to interact with the app or to configure it.

> 💡 Apps and information on programming your own apps can be found here:
>
> https://wiki.cpr-robots.com/index.php/Apps_for_t
> he_Robot_Control

To add, update, activate, deactivate or remove apps, open the app configuration area under "File" → "Configure Interfaces" → "Apps". Click on "Install app..." and select a zip archive that contains an app. The installation may take a few minutes, after which it should appear in the list. The "Update App..." button updates important files but retains the app settings. In the list you will find information about the status of the app.

If the app defines a graphical user interface, you will find this in the main view of iRC in the tabs at the top. Optionally, you can also display an app next to the 3D view.

To call an app function in a program, add it in the program editor under "Special commands" → "App function". Select the app, the desired function and enter the displayed parameters if necessary. Depending on the app, these can be numerical values, coordinates, variable names, character strings or any other data.

10.3.6   Camera Interface

The camera interface enables the use of object recognition and video cameras. Object recognition cameras detect the position and class of objects and transmit this data to the robot controller; image data is optional. If the camera provides positions as pixel coordinates, the robot controller calculates the corresponding positions in the robot coordinate system. Video cameras only provide images and can therefore only be used to observe the work area, but not for object recognition.

Currently, the robot controller supports the following camera types:

- ifm object detection cameras of the O2D200 and O2D500 series and cameras that can emulate their TCP/IP protocol.
- USB video cameras (e.g. webcams and industrial cameras supporting USB video class (UVC))

The cameras can be added to the robot controller via the "Cameras" section of the interface configuration. There, a distinction is made between cameras on the PC and the integrated controller. If the robot is controlled by an integrated controller, then the cameras must be configured there.

The following sections describe the configuration of both camera types.

> Not every integrated controller supports USB video cameras. The USB camera section below explains compatibility.

Camera Interface for Object Recognition

For object recognition only the ifm O2D200 and O2D500 cameras are currently supported as well as cameras that can emulate their TCP/IP protocol (as described here: `https://wiki.cpr-robots.com/index.php/Remote_Variable_Access#Protocol`).

> 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 enable the camera to communicate with the robot controller, first set the following parameters via the camera manufacturer's software:

- ifm O2D200
  - Format: ASCII
  - protocol version: V2 or V3
  - object detail output: on
  - start string: start or star
  - stop string: stop
  - Separator character: #
  - image output: any
  - image format: Windows Bitmap
- ifm O2D500
  - Protocol version: V3
  - For "contour presence control" mode:
    * Model result: on
    * ROI results: on
    * object results: any
    * start: start or star
    * separator: #
    * end: stop
    * number of objects: 1
  - For custom mode:
    * Use the provided protocol preset

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:

Parameter	Meaning
Image enabled	If this field is activated, the robot controller regularly requests the current camera image if the camera does not send it automatically. Supported are images in the format "Windows Bitmap" (O2D200) and "JPEG" (O2D500).
Image enabled	If this field is activated, the robot controller regularly requests the current camera image if the camera does not send it automatically. Supported are images in the format "Windows Bitmap".
Name	Name of 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 "Coordinate transformation" area determine the processing of the position data supplied by the camera. Depending on the camera settings, these are transferred as image coordinates (pixel position) or robot coordinates (in mm). Image coordinates must be transformed into robot coordinates by the robot controller; the values in the "Geometry" area are used for this purpose. See figure 39.

Parameter	Meaning
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

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

Parameter	Meaning
X, Y, Z	Object position XY in pixels (0-640 or 0-480) or XYZ in mm, depending on the setting "Source coordinate type"
Orientation	Rotation in degrees
Model class	Class of detected object, the value -1 means no object is recognized

After configuration, the detected and calculated values can be observed in the status area. Images received from the camera are displayed here.
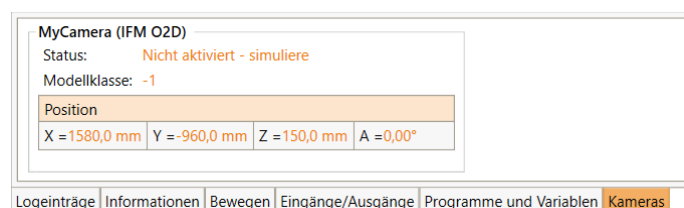
Figure 40: Camera status area
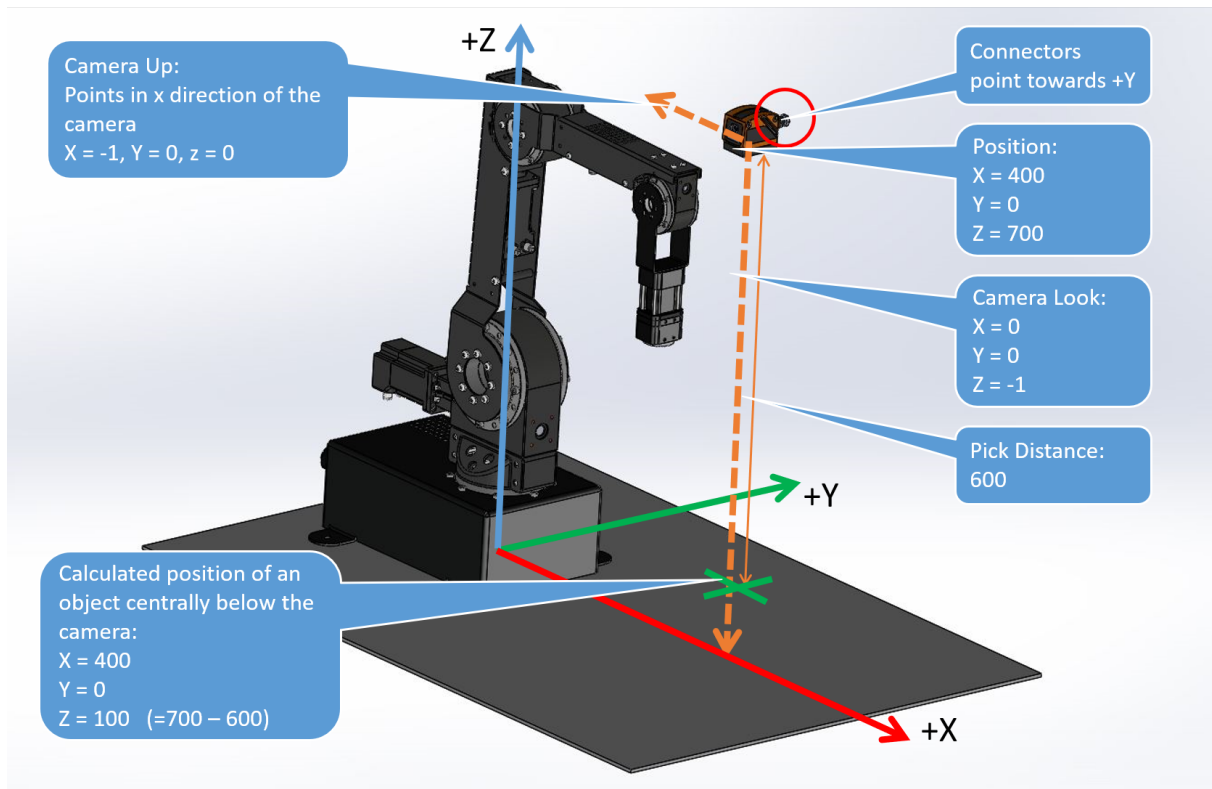
Figure 39: Measuring the camera position and orientation.

Note that O2D500 type cameras in "Contour Presence Control" mode do not provide model IDs after completing the setup wizard. With only one object type this should not be a problem, but if you want to distinguish objects an alternative protocol preset with model IDs can be selected. See the link in the previous note for more info

After making changes to the configuration of the camera via the manufacturer's software (e.g. changes to the model definition), click "Apply" once in the camera configuration area of iRC. This will reconnect the camera. Otherwise, the camera may continue to operate with the old configuration until it is rebooted

Camera Interface for USB Video

USB video class (UVC) cameras can be used to observe the work area, for example webcams or industrial USB video cameras. The robot controller forwards the received images via the following interfaces:

- CRI interface: The image is transmitted via the Ethernet interface to iRC and can be observed there in the camera status area (see fig. 40).
- Cloud interface: When this is enabled and the camera is assigned the image is transmitted to the RobotDimension cloud. The image can be monitored via the website (see section 10.3.8).

> **Compatibility:**
> The USB camera interface is only supported by integrated controllers based on the Raspberry Pi. Older robot controllers controlled by a Phytec module do not support USB cameras. The Windows software iRC does not support this camera interface either, here USB cameras can be assigned directly in the cloud configuration area (see section 10.3.8).

Connect the camera to one of the USB ports on the control module. In the camera configuration area of iRC, add a USB type camera, give it a unique name and select the device number. You may need to try different numbers in the range 0-10. The camera status area displays the connection status and, if successful, the camera image. To transfer the image to the cloud, the camera must be configured using the specified name in the cloud configuration area.

> The camera number may change after restarting the controller or if another camera has been connected. Restart the controller after configuration and correct the number if necessary.

10.3.7   Network

The WiFi interface of the integrated controller can be activated and configured via the network configuration area. When the configuration area is opened, it is automatically checked whether the function is supported. Phytec-based robot controllers generally do not support WiFi, newer controllers may require a software update.

To change the configuration, the "Change" box must first be set. Then the mode can be selected: "Disabled" disables the WiFi interface, "Access Point" opens a new WiFi network, "Infrastructure" lets the controller connect to an existing network. For an internet connection the latter is needed. Then specify the parameters (depending on the mode, only the required ones are displayed):

Parameter	Meaning
SSID	WiFi network name
Password	WiFi password
IP address	WiFi IP address of the controller. You can use this to connect to the robot. Is optional in infrastructure mode, in which case it is assigned automatically.
Router IP address	The IP address of the router in infrastructure mode, which provides the route to the Internet. This entry is optional, if empty it is assigned automatically.

The settings are usually applied immediately. Only when activating and deactivating the interface is it necessary to restart the controller. When the interface is activated, it may be blocked after the restart (similar to the airplane mode of mobile devices), in which case the configuration area shows a button that will enable the interface.

10.3.8   Cloud

The cloud interface enables remote monitoring of the robot via RobotDimension. After activation and login, the robot sends basic status information and camera images to the online service, from whose website they can be retrieved. Controlling the robot via the Internet is not possible.

```
https://www.robotdimension.com
```

To use the cloud interface, a controller with Internet connection is required, for example an integrated controller as described in section 10.3.7 or a PC with iRC, which is connected to a robot. In addition, an account with RobotDimension is required, there a separate robot password must be set in addition to the online password.

The cloud interface can be activated via the interface configuration in iRC. To enter the user information, the field "Set Credentials" must be set.

Parameter	Meaning
User name	login email address at RobotDimension
Password	Robot password at RobotDimension
Client ID	To distinguish multiple robots, if empty it will be randomly generated

The following data can optionally be sent to the cloud, it is for information only:

Parameter	Meaning
Robot name	Identifying name of the robot
Robot owner	Owner or person in charge of the robot

The images from up to two cameras can be sent to the cloud, for example to monitor the work area and the environment. For a robot connected to the cloud via iRC, USB cameras connected to the PC can be selected via the device number. If successful, a preview image is displayed. In the case of a robot with an integrated controller that connects to the cloud independently, USB and object detection cameras can be configured in the camera configuration area and assigned here for image transmission based on the name specified there.

10.3.9   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
```

10.4   Access to configuration files

The most important settings can and should be made via the graphical user interface as described above. For more specific changes, the configuration files can be changed manually.

After manual changes, the robot controller must always be restarted. This can be triggered via iRC in the "File" → "Switch off and restart..." section or by briefly switching off the electronics.

10.4.1   Access via iRC

The project, robot and tool configuration files can be downloaded via iRC, changed in a simple text editor and uploaded back to the robot control. To do this, open the "File" → "Access Remote Configuration" section. Find the section for the configuration file you want to download, click "Load..." and save the file to a location where you can find it again.
To change the file, we recommend using a simple text editor such as Notepad or Notepad++.
To copy the file back to the robot, click on the corresponding button "Write..." in iRC.

10.4.2   SFTP Access

The "Secure Shell File Transfer Protocol" (SFTP) allows access to the files of the embedded robot control, for example via a graphical file explorer (SFTP client) such as FileZilla. Changing system files via this is not possible due to access rights, use an SSH command line for this (see section 10.4.3).

Illustrated instructions are available here:

    https://wiki.cpr-robots.com/index.php/FTP_and_p
    utty_Access
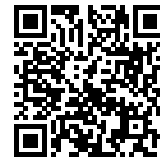
SFTP is the SSH File Transfer Protocol. It can be used to transfer or adjust files from one computer to another. FileZilla is a free FTP program.

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. By default 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 robot control software called RobotControl. It is located in the ~/RobotControl folder.

- The directory structure inside the RobotControl folder is similar to 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 or Notepad++.

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.

10.4.3   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_p
utty_Access
```
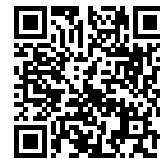
An SSH client such as PuTTY can be used to connect to the integrated controller and work on a command line. In this way, you can, for example, view the live output of the robot controller or change files.

> ⚠️ Linux knowledge is required to work on the command line! Errors may result in the system having to be reset to the factory settings

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. By default 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 RobotControl directory, whose content is similar to the iRC directory in Windows.

7. Robot files are located in ~/RobotControl/Data/Robots/<category>/<robot type>/<robot type>.xml.

8. The project file where the robot file is referenced are located in ~/RobotControl/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 RobotControl log output live in the terminal (this is especially useful after changes in the files), RobotControl must first be terminated: killall RobotControl

12. Then navigate to the directory ~/RobotControl and start ./RobotControl

13. The process can be terminated by pressing Ctrl+C.

14. After restarting the controller RobotControl starts automatically.

10.5   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.

11 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.

11.1 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.

11.2 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_Ser
ver
```

For including the library please contact the Siemens support.

```
https://support.industry.siemens.com/cs/ww/de/vi
ew/37364723
```

11.2.1 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.

		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.

11.2.2   Inserting the Robot Communication FB

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

Signal	Data Type	Explanation
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.

Signal	Data Type	Explanation
Enabled	Bool	Robot is enabled
Referenced	Bool	Robot is referenced
ProgrammRunning	Bool	The robot program is running

11.2.3   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.

11.3   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.

Memory 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 24: Use of the Modbus address sections

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

Memory 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 25: 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 11.3.5.
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 26: Definition of complex data types

> 💡 Position values (X, Y, Z, A, B, C, joint angles) are given as 32 bit values where necessary and therefore occupy 2 registers. The first register contains the lower bits, the second register the upper bits. For negative values both registers must be set accordingly.
> The following link shows an example on how to calculate these values and move the robot to a target position.
>
> https://wiki.cpr-robots.com/index.php/Moving_Robots_via_Modbus
>

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

11.3.1   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.

Read access to 1-bit data is possible both as coils and as discrete inputs so that fewer Modbus messages have to be sent for reading both. Reading values that were previously written as coil via discrete inputs should be avoided. 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 33)
60	⊓	Are all axes referenced? / reference
61-66	⊓	Is robot axis referenced? / reference
67-69	⊓	Is external axis referenced? / reference

Addresses	Type	Description
70-72	⌐	Is gripper axis referenced? / reference
73	⌐	Set all axes to 0
74	⌐	Start the referencing program, then "reference all". Must be referenced and motors enabled.
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 27: Assignment of coils and discrete inputs

11.3.2 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 11.3.5.

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

Addresses	Type	Unit	Description
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

Addresses	Type	Unit	Description
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. 11.3.4)

Table 28: Assignment of the Input Registers

11.3.3   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	int32	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	Velocity for MoveTo (percent or mm/s)
181-186	int32	0.1	Target velocity of ext. axes in velocity mode
187	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

Addresses	Type	Unit	Description
456-711	int16	0.1	Position variables mb_pos_w1 - mb_pos_w16 (see sec. 11.3.4)

Table 29: Assignment of the Holding Registers

11.3.4 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 30: Program variables for communication via Modbus

> ℹ️ 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

Value	Meaning

Table 31: Conversion type

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

11.3.5 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 32: 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 33: Operation mode

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

Value	Meaning
0xFFFF	Invalid

Table 34: Jog Mode

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

Table 35: RunState

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

Table 36: 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 37: Reason for last stop/pause of program

12   Maintenance

12.1   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).

13   Troubleshooting

13.1   Frequently Asked Questions

Antworten zu häufig gestellten Fragen finden Sie in unserem Wiki:

```
https://wiki.cpr-robots.com
```

13.2   Error Codes and Solutions

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

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

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

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

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

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

13.3   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 9.3).
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_Sof
tware_ModuleCtrl
```

13.4   Support Contact

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

- igus Support landingpage:

  ```
  https://www.igus.eu/info/igus-low-cost-automatio
  n
  ```

- 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