SOFTWARE MANUAL

Version: 1.2.1 EN01

Kassow Robots ApS
Oliefrabriksvej 57
DK-2770 Kastrup

info@kassowrobots.com
kassowrobots.com

1 Introduction

1.1 Product Description

The robot consists of the robot arm, a controller cabinet, a teach pendant and the connection cables between those.

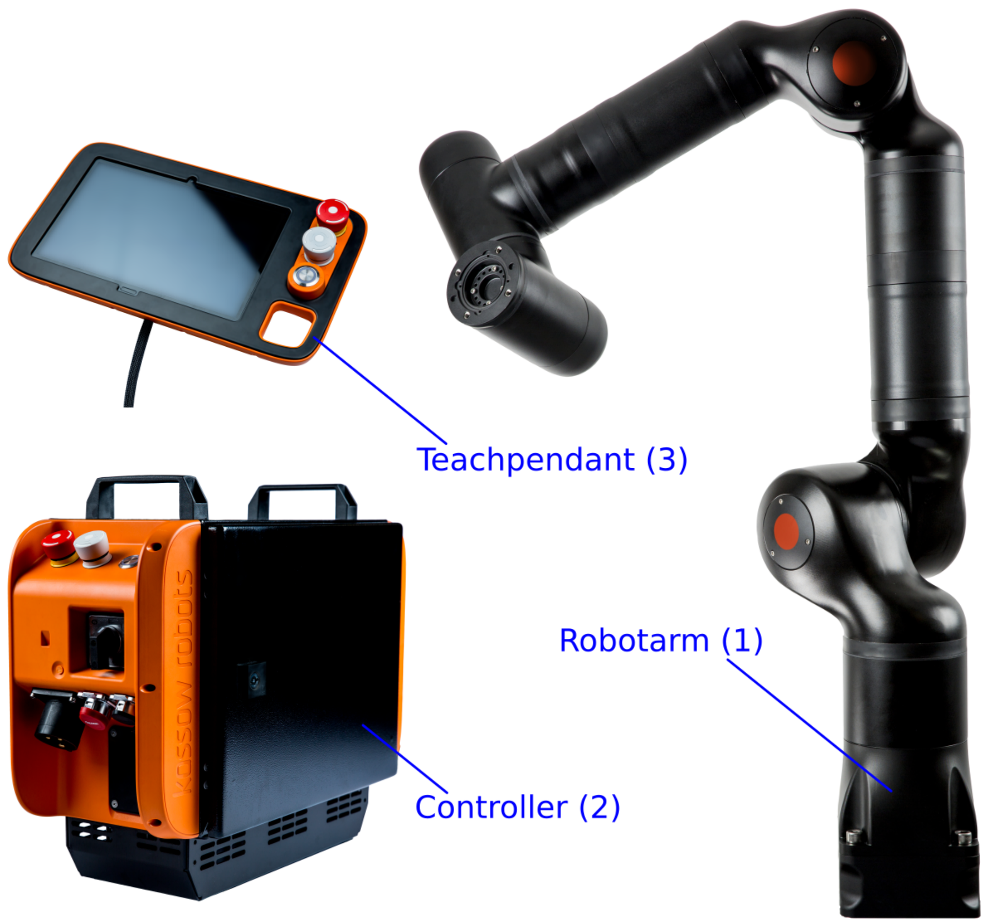


Figure 1: Picture of the Robot arm

1. The robot arm consists of 7 geared servo motors connected mechanically by machined extrusions and castings and connected electrically by a 48V power supply bus and a serial communication bus.

2. The controller cabinet house various power supplies: the computer which coordinated the movements of the 7 geared servo motors; the I/O board, which has a range of safety related

functions and allow various electrical connections; the relays on the robot arm power supply bus; connectors for connecting to wall socket, robot arm and teach pendant and an E-stop button, a Protective stop button and a play/pause/resume toggle button.

3. The teach pendant / HMI which allows the user to program the robot system and which house two buttons, which when pushed allow the user to manually manipulate the robot arm, and an E-stop button, a Protective stop button and a play/pause/resume toggle button.

The wires connecting the system components are: Two wires with connector in both ends between robot arm and controller cabinet; one wire fixed to the teach pendant and with a connector in the other end, to connect teach pendant to controller cabinet and one wire with a 3P CEE female connector in one end and male connector in the other end, to connect wall socket with controller cabinet.

2 Safety

2.1 General

📖	Before the starting to program and operate this robot this instruction manual must be thoroughly read and understood!
⚠	This warning symbol indicates that special precautionary measures must be taken. If the safety precautions are not observed, it may lead to hazardous conditions and result in personal injury or damage to property.
❗	This symbol indicates that the following information is important.

2.1.1 Operation

Always ensure safe operation of the unit. It is the responsibility of the owner to ensure that the personnel, who is operating the unit is familiar with the stopping procedure of the unit. The unit must only be used when all safety precautions have been observed, which means that guards, grids, covers etc. are to be firmly bolted on, etc. All safety prescriptions mentioned in the manual must be observed. Always ensure that the unit is well maintained and in good mechanical condition before any operation. Always inspect the unit before use. Ensure that all safety precautions are meet, before operating the unit; hence, that guard, grids, covers among others are secured and firmly fixed and not damaged. Make sure that no foreign objects (metal parts etc.) enter the unit during operation. Never open electrical cabinet door during operation. The unit generates noise as specified in the specifications. Hearing protection may be used, where it is considered necessary, taking into account factors such as the installation of the unit, the conditions of use of the unit, the characteristics of the workplace (such as, for example, noise absorption, the scattering of noise, noise reflections), noise emissions from other sources (such as, for example, from other machinery), the position of persons with respect to the sources of noise, the duration of exposure und the use of personal protective equipment (hearing protectors).

2.1.2 Training and instructions

For the personnel to protect themselves from machine and workplace hazards, each discipline - operators, cleaning and service personnel etc. - must receive instructions that suit their particular subject area.

- Operators/programmers: Instruction in the safe operation of the system is required and the manual must be accessible at all times.
- Cleaning and Service Personnel: The necessary safe instruction needed to carry out maintenance work, basic technical knowledge.

The unit must be maintained according to the instructions in the instruction manual.

Only trained/instructed and if required also qualified personnel may conduct cleaning, maintenance and repair of the unit. It is the responsibility of the owner to ensure that the personnel, who is conducting the cleaning, maintenance and repair is familiar with the stopping procedure of the unit.

Never attempt to make any repairs, adjustments, or inspections while the unit is operating/ running. Before any maintenance or repair works or cleaning, follow the lock out, tag out (LOTO) instructions below:

- Empty the equipment for material.
- Stop unit and disconnect the supply disconnecting device.
- Tag out the disconnecting device stating “ATTENTION! DO NOT CONNECT. PERSONNEL CARRYING OUT WORK ON THE UNIT”.
- Release the pressure inside the unit (where relevant).
- Ensure that no hazardous energies has been build up.
- Before any work on electric installations ensure that the power is disconnected.

Check the supply disconnecting device or power source and make sure the power is always off before starting any electrical work. Never remove nor disassemble electrical components and/or wires while the unit is running and/or connected to the power grid. Lock the supply disconnecting device if possible or if cannot, make sure everyone in the area knows it is turned off for a reason and leave an easily noticeable note that work on the electricity is ongoing. All electrical installations must be made by a qualified electrician/technician and in compliance with local regulations. The electrical installation must comply with the requirements for the motor and any other preinstalled components. See data sheets. Danger of injuries by rotating elements.

The owner must ensure at least 300 lux are present in the area where normal supervision takes place (front of the unit). During assembly, disassembly, installation, service, maintenance and the like it is advisable to establish temporary extra lighting. Only operate the unit while on the outside with all guards in place. Ensure to consider ergonomic hazards while tightening elements with a relatively high torque value.

2.1.3 Cleaning

Only trained/instructed personnel may conduct cleaning of the unit. It is the responsibility of the owner to ensure that the personnel will receive the training and knowledge they need, in order to become familiar with the unit and able to protect themselves from workplace hazards. It is the responsibility of the owner to ensure that the personnel, who is conducting the cleaning is familiar with the stopping procedure of the unit. Always wear long-sleeved rubber gloves when cleaning the unit. Cleaning may ONLY be done following LOTO instructions in this manual. Kassow Robots ApS will not be held liable for damage due to wrong or inappropriate cleaning methods or use of too harsh chemicals. Be careful not to damage the surfaces nor functionality, when cleaning the equipment. Use only appropriate cleaning methods that do not deteriorate, corrode or discolour the surfaces of the equipment. Ensure that the detergent used does not deteriorate, corrode or discolour the surfaces of the equipment. If in doubt what suitable detergents to use, please consult your supplier of detergents and chemicals for cleaning purposes. Ensure to rinse all detergent residues of the equipment.

2.1.4 Instructions for safe operation

Only the operator that holds the Teach Pendant are allowed to be near the robot when using the Teach Pendant. All other persons shall be an distance of 3m.

2.1.5 Special Situations

If the operator is trapped by the robot arm the operator shall push the robot arm away to release themselves.

Liability

The content panel displays the tool selected via tool bar. There are two content panels (left and right) allowing the user to use two same or different tools at the same time.

The device described in this document is either an industrial robot or a component thereof.

- Manipulator
- Robot Controller
- Teach pendant
- Connecting cables
- Software

The cooperative robot is built using state-of-the-art technology and in accordance with the recognized safety rules. Nevertheless, misuse of the industrial robot may constitute a risk to life and limb or cause damage to the industrial robot and to other material property.

Terms used

Term	Description
RC	**Robot Controller** is the main controlling unit required for each Kassow Robot manipulator.
TP	**Teach Pendant** stands for the portable programming and manual platform providing all necessary user interface and safety controls.
TPUI	The teach pendant **User Interface** is the software user environment running on the manual platform (TP). This provides all software tools necessary for the robot programming, teaching and accessing extension modules.

ESTOP	A stop initiated by any active component of the robot (Joints, IO-Board, Tool-IO, Controller computer). Executive parts of the robot are immediately halted and cut loose of power anytime the ESTOP condition is identified within the system.
PSTOP	A stop initiated by any active component of the robot (Joints, IO-Board, Tool-IO, Controller computer). The executive parts become limited and monitored for energy supplies anytime the PSTOP condition is identified within the system.
Safety mode	Safe and Reduced modes limit individual robot axis moving at maximum speeds of 0.25m/s and 1.0m/s. The Normal mode doesn't provide any limitation upon the link and joint motion.
Back-drive	The KR free drive mode allowing to move the robot manually. The function is controlled by the teaching buttons present at the back side of the TP or at the Tool-IO flange (located at the end of arm).

2.2 Safety Functions

Safety functions evaluate external and internal signals of the whole system which can act immediately to halt the robot or cut him loose from power if necessary.

2.2.1 Emergency STOP

The emergency STOP buttons are present at both Teach pendant and Robot cabinet. These are used to initiate a robot motion or other potentially hazardous situation.

While Emergency buttons provide immediate interaction and safety for the operator, the internal variables are also continuously monitored to avoid any internal damage to system.

The integrity guards check for thr basic life conditions of the system, including temperatures, voltage or currents. Further safety checks keep track with the sensor data consistency.

The emergency STOP always provoke immediate halt of the robot, followed by the power cut to all executive parts of the robot.

2.2.2 Protective STOP

The Protective STOP can be used interactively by the operator to pause and continue the running program.

Internally, the system can also provoke the Protective STOP to apply limit energy guards upon the system when some conditions are met and warn the user about the case. Depending on the type of diverging values, the program can continue its normal operation when the Protective STOP is released.

2.3 Applied norms and regulations

The safety function complies with EN ISO 13849-1:2015.

The emergency stop function is constructed as a category 3, performance level d according to EN 10218-1:2011

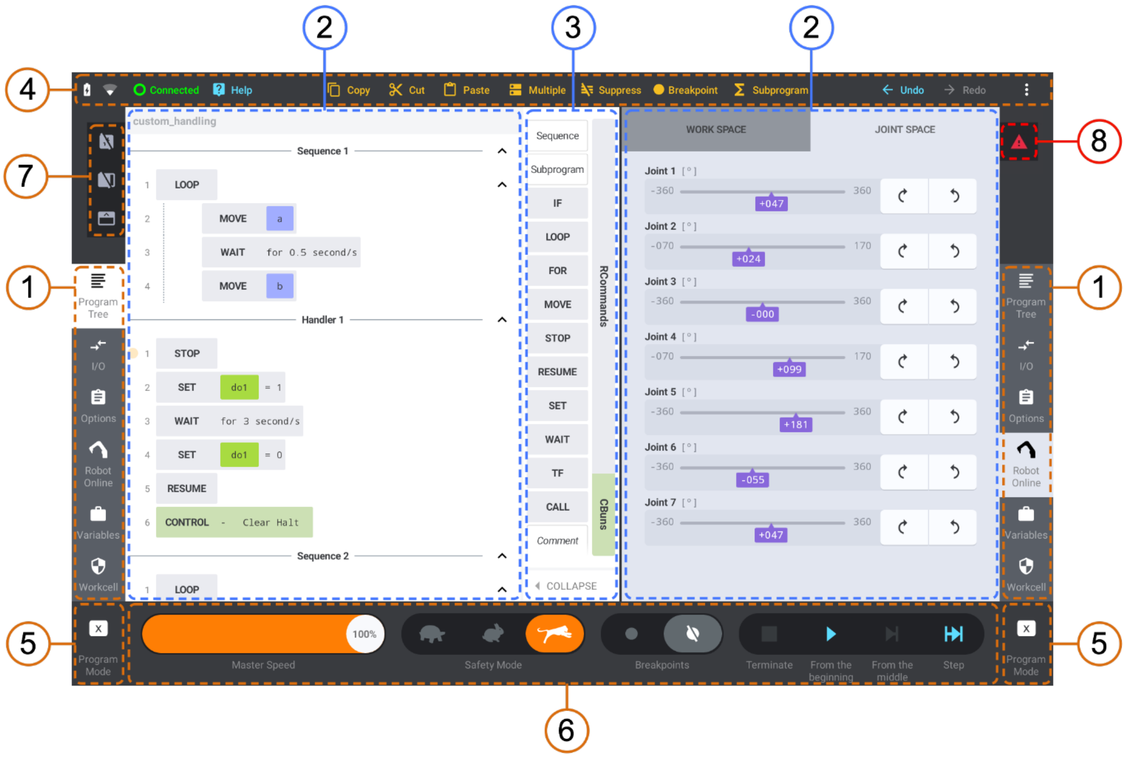
Additional emergency stops added to the robot, must also be constructed as performance level d.according to EN 10218-1:2011

3 User Interface

Kassow Robots graphical user interface is easy to learn, since it comes with familiar design of tablet applications. This intuitive touch-based interface allows users to control robot arm and HW equipment, build and run programs and configure robot installation setup.

3.1 Main Screen

Main screen represents the most important part of the user interface. Main screen comes with two-pane design. This design makes possible to use two different tools at the same time.



Item	Description
1	**Tool Bar** (see 3.1.1) Provides tool selection and switches content of Content Panel.
2	**Content Panel** (see 3.1.2) Displays the selected tool on the screen. Both the left or right half of the screen can be used.
3	**Command Box** (see 3.1.3) Provides list of available program command.

4	**Action Bar** (see 3.1.4) Displays status icon and allows program edit.
5	**Switch Mode Button** Swaps Bottom Bar content.
6	**Bottom Bar** (see 3.1.5) Displays list of variables or program control.
7	**Display Options** (see 3.1.6) Adjusts the program tree view.
8	**Status Icon** (see **Error! Bookmark not defined.**) Indicates various system issues. Clicking the icon shows the issue description.

3.1.1 Tool Bar

The tool bar allows the user to select one of the available tools to be displayed in the related content panel. Two tool bars (left and right) are available in order to provide access to two same or different tools at the same time.

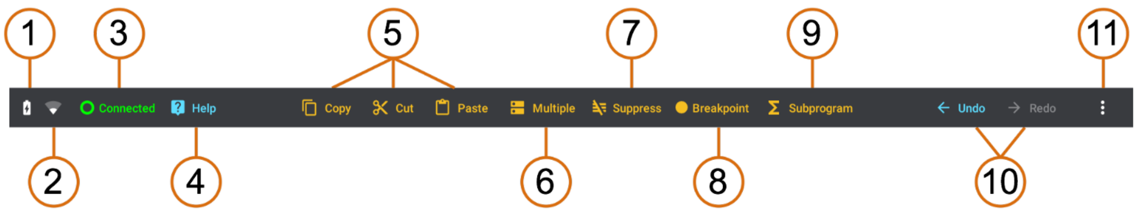
Tool	Description
Program Tree	The interactive view of the actual program listing providing the visual and responsive program alterations (see Basic Components).
I/O	Immediate access to available I/O interfaces (see I/O Interface).
Options	Displays the settings of the selected program command or variable (see Variables).
Robot	The robot jogging interface (workspace/frame, joint, self-motion) (see Robot Online).
Variables	Provides the list of custom (user defined) and system (mapped) variables and allows the user to create new ones (see Variables).
Workcell	Provides robot installation setup (see **Error! Reference source not found.**).

3.1.2 Content Panel

The content panel displays the tool selected via tool bar. There are two content panels (left and right) allowing the user to use two same or different tools at the same time.

3.1.3 Command Box

The command box provides an instant access to basic program building blocks, extensions loaded from



the CBuns or subroutines defined by the user. Those commands can be dragged and dropped into the program tree. The command box is available only when the program tree is shown at least in one of the content panels. Once the command box is available, it can be collapsed in order to provide more space for the content panels (see 3.1.6). See the detailed description of commands in the section 5.2.

3.1.4 Action Bar

The action bar provides tools for program editing such as copy/cut/paste and undo/redo. Moreover, the action bar contains system status indicators (battery, connection) and allows the user to open main settings.

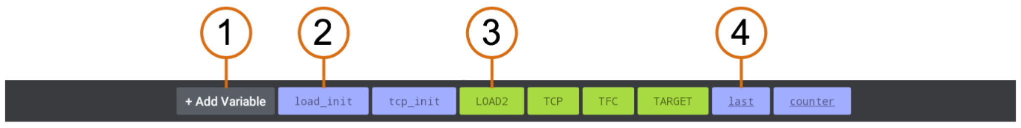
Item	Description
1	**Battery Status** Displays the battery status (level and charging state).
2	**WiFi Connection Status** Indicates if the device is connected to WiFi and what is the signal strength.
3	**Robot Connection Status** Shows the connection status (unreachable, connected, not connected) and allows the user to connect or disconnect manually.
4	**Help mode** When in the interactive help mode, the user can view the relevant help document simply by touching the element on the screen.
5	**Copy/Cut/Paste** Provides the standard copy/cut/paste operations on program commands and variables. Copy/Cut/Paste controls are enabled only when command or variable is selected. Paste button also requires previous Copy/Cut action to be enabled. All the actions are supported also for multiple command selection.
6	**Multiple Selection** Once the switch is on (orange background), the user can select multiple program commands or variables.
7	**Suppress** The suppress button allows the user to suppress selected command or commands in the program tree. Those commands will not be executed. The suppress button is enabled only when a command is selected.
8	**Breakpoint** Places a breakpoint to all selected command/s. If the breakpoints are enabled (see Control Mode), the breakpoint pauses the program execution.
9	**Subprogram** The subprogram button converts selected command/s into a new Subprogram and replaces them with a single CALL command.
10	**Undo/Redo** Undo reverses the latest user action. Redo can restore the Undo changes.
11	**Settings** Opens the settings popup menu.

3.1.5 Bottom Bar

The bottom bar provides two different modes (Program Mode and Control Mode). The user can swap between these two modes via clicking the Switch Mode Button.

Program Mode

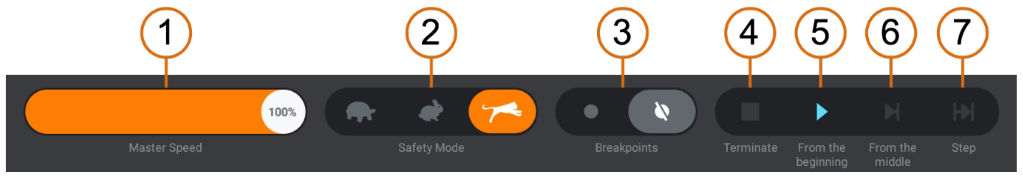
The program mode is recommended during the editation of a program, since it contains list of all custom and system variables. These variables can be dragged and dropped into special fields (options) of selected command or variable. Except the list of variables, the program mode contains also the Add Variable button, which displays the "*Create New Variable*" dialog.



Item	Description
1	**Add Variable Button** Allows the user to create new custom variable.
2	**Custom Variable** Example of custom (user defined) variable – blue color.
3	**System Variable** Example of system (mapped variable) – green color.
4	**Persistent Variable** Example of persistent (user defined) variable – blue color, underline.

Control Mode

The control mode mode allows the user to launch the program and control the execution process (incl. Debugging). Therefore the control mode is recommended during the execution of program.



Item	Description
1	**Master Speed Slider** Sets the speed of robot movement during the program execution (see 6.1.1).
2	**Safety Mode Switch** Sets safety mode: Safe (Turtle), Reduced (Rabbit) or Unlimited (Cheetah) (see 6.1.2).

3	**Breakpoints Switch** Enables/disables the program debugging (see 6.6.4).
4	**Terminate Button** Terminates program execution.
5	**Play/Pause Button** Launches, pauses or resumes program (see 6.2).
6	**From the Middle Button** Not supported yet.
7	**Step Button** Provides an execution of a single program command (see 6.6.3).

3.1.6 Display Options

The Display Options menu provides tools for customization of application appearance such as show/hide program line numbers, show/hide command box or resize program elements (commands, variables).

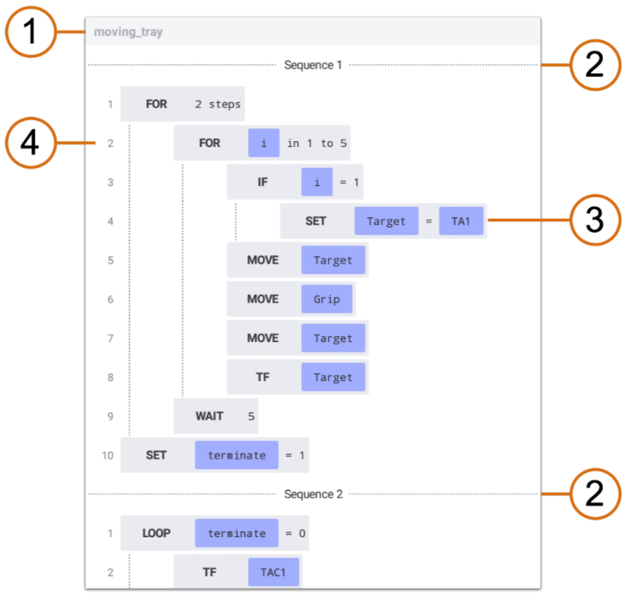
Item	Description
1	**Line Numbers Switch** Shows/Hides line numbers in Program Tree tool (see **Error! Reference source not found.**).
2	**Command Box Switch** Shows/Hides command box. The command box is available only when the program tree is shown at least in one of the content panels.
3	**Element Size Switch** Switches between small and large size of commands and variables.

3.1.7 Status Icon

The status icon (warning sign) is shown whenever some system issue occurs, or the program cannot be executed for some reason. Clicking the icon makes the ”System Info“ dialog appear. This dialog provides the issue description.

3.2 Program Tree

The program tree panel shows the active program which consists of one or more sequences. Every sequence should contain one or more program commands (each command occupies a single line of the sequence). The command indent allows the user to determine whether the commands are siblings or parent and child/children. The program tree supports drag drop mechanism which provides easy way for moving program commands into new positions.



```
moving_tray
Sequence 1
1   FOR 2 steps
2       FOR i in 1 to 5
3           IF i = 1
4               SET Target = TA1
5           MOVE Target
6           MOVE Grip
7           MOVE Target
8           TF Target
9       WAIT 5
10  SET terminate = 1
Sequence 2
1   LOOP terminate = 0
2       TF TAC1
```

Item	Description
1	**Program Name** Name of the saved program or *“untitled“* if not saved yet.
2	**Sequence Header** Represents the beginning of a new sequence (see 5.2.1).
3	**Program Command** Individual program command (see 5.2).
4	**Line Number** Line numbering related to the sequence header.

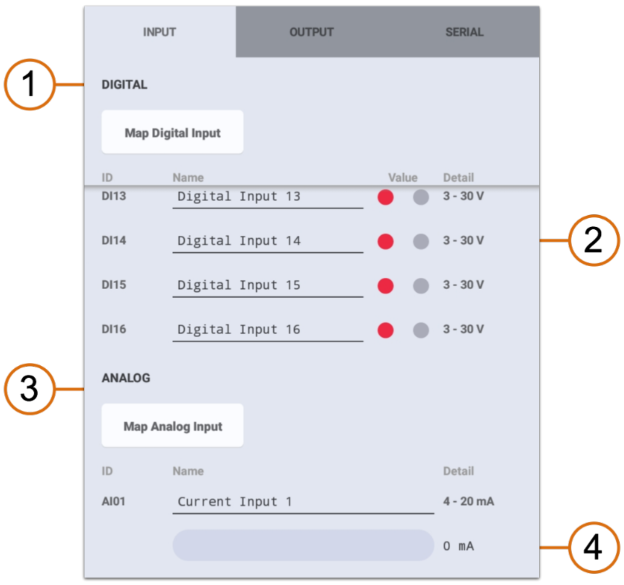
3.3 I/O Interface

Kassow Robots are equipped with two groups of the input/output ports. The one located at the RC cabinet (**IO-Board**), the other one provides interfaces at the end of arm tool flange (**Tool-IO**). Both groups can be controlled and configured from the robot programs, CBuns extension modules or directly jogged within the IO panel.

The IO panel consists of three tabs: INPUT, OUTPUT, SERIAL. The INPUT panel displays the state of digital inputs and values measured by analog inputs. The OUTPUT panel allows the user to control the digital and analog outputs simply by touching the related switch (digital outputs) or moving the slider (analog outputs). The SERIAL panel provides interface for RS232 and RS485.

3.3.1 Input

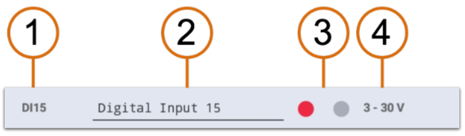
The Input tab provides the interface to access digital and analog inputs of the RC **I/O-board** and the **Tool-I/O**. The input ports are divided into two sections: Digital and Analog. Each section contains list of related input ports. The input ports are available only if the robot is turned on and connected. Note that the content can differ for various versions of I/O-boards or Tool-IOs.



Item	Description
1	**Digital Input Header** Allows the user to map the digital input into system variable. Clicking the Map Digital Input button shows “Map IO Variable“ dialog.
2	**Digital Input Item** Single digital input port interface.
3	**Analog Input Header** Allows the user to map the analog input into system variable. Clicking the Map Analog Input button shows “Map IO Variable“ dialog.
4	**Analog Output Item** Single analog input port interface.

Digital Input Item

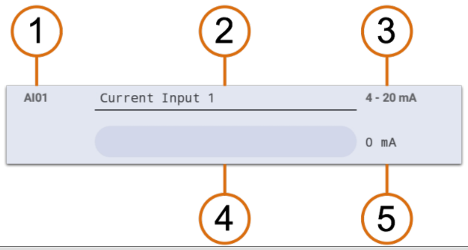
The digital input item represents the single digital input port of the particular IO group. This interface provides instant indicator for the port input value.



Item	Description
1	**ID** Unique port ID generated by I/O board.
2	**Label** Port name generated by I/O board.
3	**Value Indicator** Displays the digital input port value (RED = OFF, GREEN = ON).
4	**Description** Describes value (typically port range and value units).

Analog Input Item

The analog input item visualises a single analog input port of the IO-Board or the Tool-IO, by engaging the bar indicator and the float value with assigned unit (current or voltage).

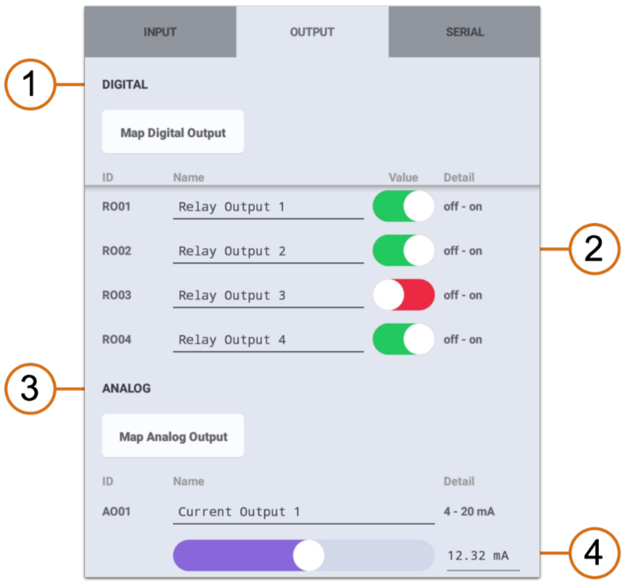


Item	Description
1	**ID** Unique port ID generated by I/O board.
2	**Label** Port name generated by I/O board.
3	**Description** Describes value (typically port range and value units).
4	**Indicator Bar** Shows the value within the port value range.
5	**Value** Displays the current measured port value accompanied by the unit (current or voltage).

3.3.2 Output

The Output tab provides UI for the instant control of the digital and analog outputs supported by the IO-Board or the Tool-IO. Similarly, as for the inputs, the output ports are divided into two sections: Digital and Analog. Each section contains list of accessible output ports.

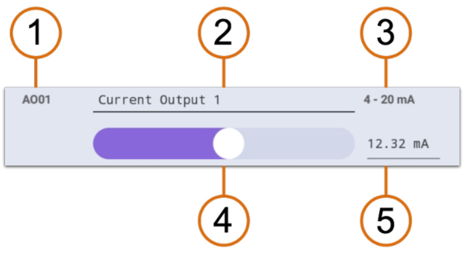
The output ports can be controlled only when the robot is turned on and connected. Note that the content can differ for various versions of the IO device. The Output tab also allows the user to map the output port into a system variable.



Item	Description
1	**Digital Output Header** Allows the user to map the digital output into system variable. Clicking the Map Digital Output button shows “*Map IO Variable*“ dialog.
2	**Digital Output Item** Single digital output port interface.
3	**Analog Output Header** Allows the user to map the analog output into system variable. Clicking the Map Analog Output button shows “*Map IO Variable*“ dialog.
4	**Analog Output Item** Single analog output port interface.

Digital Output Item

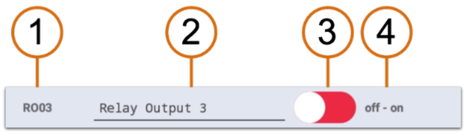
The digital output item represents a single digital output port of I/O board. This interface allows the user to quickly set the digital output value or access the current port value.



Item	Description
1	**ID** Unique port ID generated by I/O board.
2	**Label** Port name generated by I/O board.
3	**Value Switch** Allows the user to set the port value (RED = OFF, GREEN = ON).
4	**Description** Describes value (typically port range and value units).

Analog Output Item

The analog output item represents a single analog output port of I/O board. This interface allows the user to quickly set the analog output value or access the current port value.

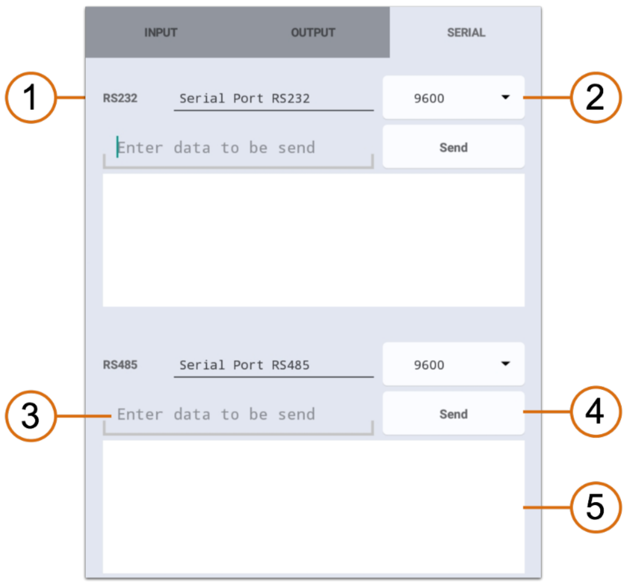


Item	Description
1	**ID** Unique port ID generated by I/O board.
2	**Label** Port name generated by I/O board.
3	**Description** Describes value (typically port range and value units).
4	**Slider Bar** Allows the user to set the output value inside the port output range. The value is being updated continuously as the user drags the slider.
5	**Value Edit Field** Provides the interface for setting the analog output value precisely. The value is updated once the user finishes the editation.

3.3.3 Serial

The Serial tab contains the UI for an instant access to serial interfaces (RS485, RS232) provided by the IO devices. The serial ports can be controller only when the RC is turned on and connected.

The content of this tab may vary, depending on type of the IO device. With the present version of the KSW, the serial ports cannot be mapped into single system variables. But they are accessed indirectly through the IO API from particular device drivers (CBuns).



3.4 Options

The options panel displays the settings of selected program command or variable; therefore the options panel content depends on the selected element and shows different options for each program command type and variable type. If nothing is selected or multiple program commands or variables are selected, options panel is empty.

For more information, please see 5.

3.5 Robot Online

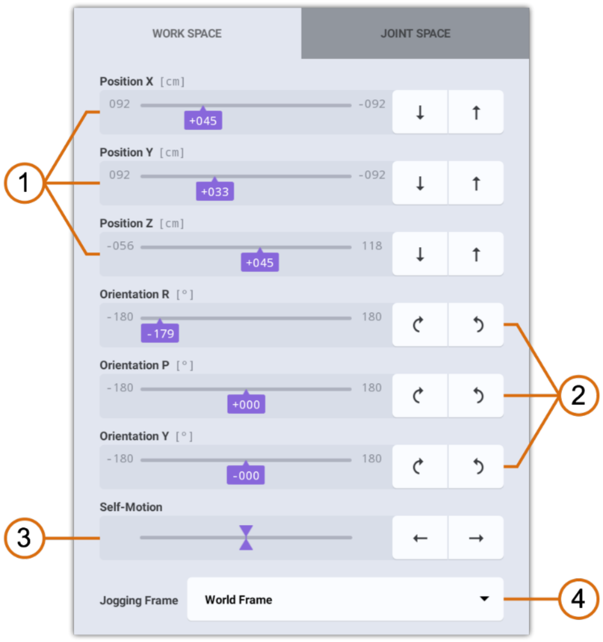
The Robot Online tool provides an interface for jogging the robot. There are two tabs allowing two ways of jogging the robot. The robot can be jogged in Work Space or Joint Space.

3.5.1 Control Modes

All Robot Online buttons (except Self-Motion) support 3 control modes. Continuous robot movement (3 vibrations) is provided when a button is pressed and hold, whereas short button click (1 vibration) leads to a micro fine tuning (0.1 mm in workspace, 0.1° in joint space) and long button click (2 vibrations) leads to a macro fine tuning (1 mm in workspace, 1° in joint space).

3.5.2 Work Space

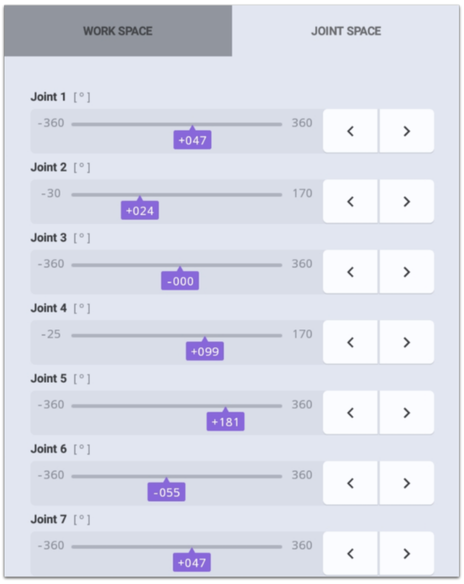
Jogging the robot in work space means, that the end effector (TCP Frame) is moving or rotating along the axes of world coordinate system. The robot is moving along the selected axis as long as the user keeps the button pressed.



Item	Description
1	**TCP Position** Moves the end effector (TCP) along the axes of world coordinate system.
2	**TCP Orientation** Rotates the end effector (TCP) along the axes of world coordinate system.
3	**Self-Motion** Moves the robot through different joint configurations but keeps the same end effector (TCP) position and orientation.
4	**Jogging Frame** Specifies in which frame the jogging should be provided.

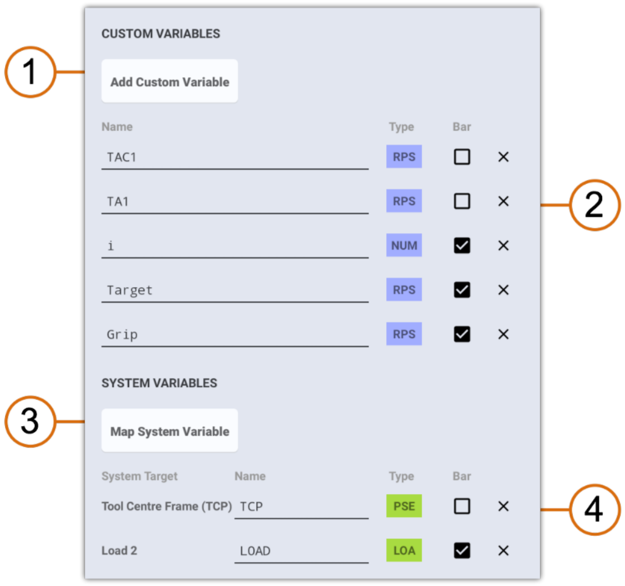
3.5.3 Joint Space

Jogging the robot in joint space means, that each robot joint is being rotated individually along its own axis. The robot is moving along the selected axis as long as the user keeps the button pressed.



3.6 Variables

The Variables tool displays the complete list of custom (user defined) and system (mapped) variables including the variable type. It also allows creation of new custom variables and mapping of system variables as well as their deletion. For information about variables debugging see 6.6.5.



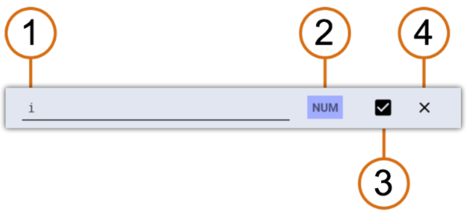
Name	Type	Bar	
TAC1	RPS	☐	×
TA1	RPS	☐	×
i	NUM	☑	×
Target	RPS	☑	×
Grip	RPS	☑	×

System Target	Name	Type	Bar	
Tool Centre Frame (TCP)	TCP	PSE	☐	×
Load 2	LOAD	LOA	☑	×

Item	Description
1	**Custom Variables Header** Allows the user to create new custom variable. Clicking the Add Custom Variable button shows „Create New Variable" dialog.
2	**Custom Variable Item** Single custom variable item.
3	**System Variables Header** Allows the user to map some system value (such as I/O) into system variable. Clicking the Map System Variable button shows "Map System Variable" dialog.
4	**System Variable Item** Single system variable item.

Custom Variable Item

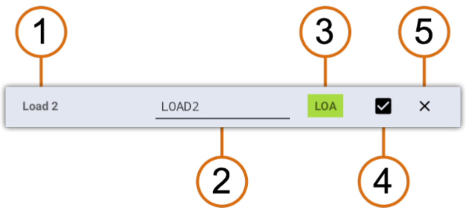
The custom variable item represents a single user defined variable. This interface allows the user to quickly delete the variable, access its type or settings.



Item	Description
1	**Variable Name** Displays user defined variable name (alias).
2	**Variable Type** Determines the variable type: NUM (number), PSE (pose), RPS (pose incl. joints configuration), LOA (load), PTR (pattern) and ARR (array).
3	**Bottom Bar Shortcut** If checked, the variable is listed in Bottom Bar.
4	**Delete Button** Clicking the delete button deletes the selected variable. Beware that this action is permanent.

System Variable Item

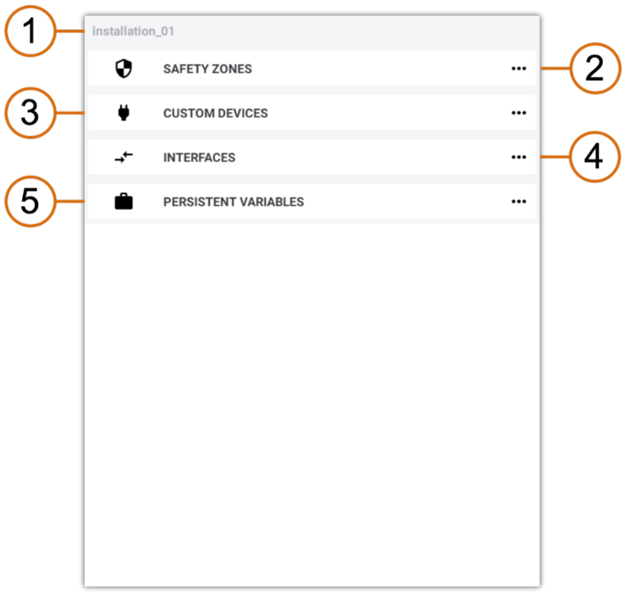
The system variable item represents a single mapped variable. This interface allows the user to quickly delete the variable, access its type or settings.



Item	Description
1	**System Target** Determines the system target which value is being mapped into the system variable.
2	**Variable Name** Displays user defined variable name (alias).
3	**Variable Type** Determines the variable type: NUM (number), PSE (pose), RPS (pose incl. joints configuration), LOA (load).
4	**Bottom Bar Shortcut** If checked, the variable is listed in Bottom Bar.
5	**Delete Button** Clicking the delete button deletes the selected variable.

3.7 Workcell

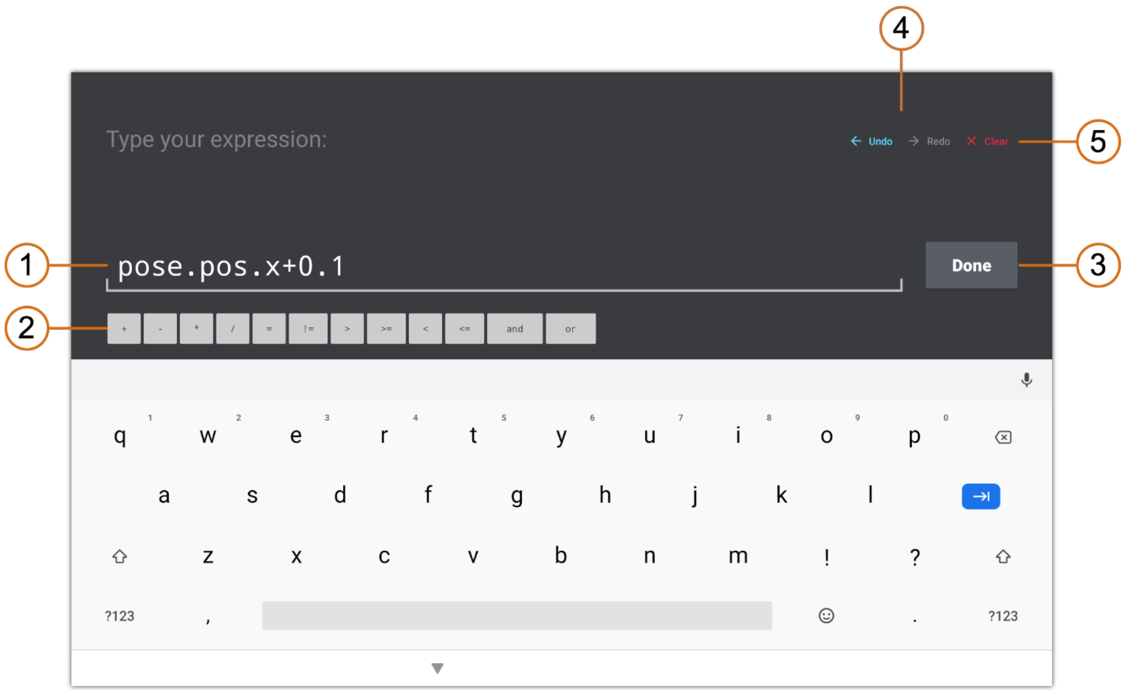
The Workcell tool allows users to configure the robot installation setup (see 7).



Item	Description
1	**Workcell Name** Name of the saved workcell or “*untitled*“ if not saved yet.
2	**Safety Zones** Provides list of safety zones and their geometries (see 7.1).
3	**Custom Devices** Provides access to custom devices (see 7.2).
4	**Interfaces** Allows users to configure network interfaces (see 7.3).
5	**Persistent Variables** Provides list of persistent variables (see 7.4).

3.8 Expression Builder

Some of the commands (such as IF, LOOP, SET, WAIT and MOVE) require an expression to be defined. Such expression can be defined via expression builder.



Item	Description
1	**Expression** Displays the expression and allows its edit via built-in keyboard. Also contains syntax validity check (invalid expression parts are marked in red).
2	**Hint Bar** Displays variables, members and operators that match the current cursor context.
3	**Done Button** Returns the user to the main screen.
4	**Undo/Redo** Provides undo/redo on selected expression.
5	**Clear** Clears the whole expression.

3.9 Teaching

It is of a common practice in the world of robot programming to jog or free-drive the robot manually and define poses and related motions based on the physical coordinates of the EOAT (end of arm tool) and context of the intended application. The basic way how to read and store the actual TCP or the robot configuration in the reusable Pose variable is provided by the TPUIO, e.g. when the new Pose variable is created/modified by the user.

To give the user more freedom and release his hands from the TP the recent software version implements a set of supportive *teaching functions*. Invoking the teaching function is simple, it requires to double click of the teaching button, either the one placed at the Tool-IO of the robot arm, or the one at the back side of the TP.

Based on the context of the TPUI, the teaching function provides an automated action as stated in the following table.

User Context	Teaching Function
Move command selected (with the empty target pose)	Creates the new Pose variable (labelled “P#number”) having the actual robot TCP/JointConfiguration (JCF) and assigns it to to the selected MOVE command target.
Move command selected (target pose already set)	Duplicates the selected MOVE command and place it after the actual selection. Creates the new Pose variable (labelled “P#number”) having the actual robot TCP/JCF and assigns it to the freshly created MOVE as target pose.
Non-array variable selected	Creates the new Pose variable (labelled “P#number”) having the actual robot TCP/JCF.
Array variable selected (array contains Pose typed column)	Takes the first column of the Pose type and define the actual robot TCP/JCF in the first empty row (or adds a new row at the end).
Array variable selected (array doesn’t contain Pose typed column)	The new Pose column is added to the array definition and fills-in its first row by the actual robot TCP/JCF.

Teaching functions are quite an efficient tool helpful in situations, where the row of move commands or poses are expected and defined from the physical robot configurations (teached). In this case the user can eliminate any extra interaction with the TPUI and focus on the robot arm positioning with the semi-automated program alteration provided by the teach button double clicks.

4 Storage

4.1 Locations

Kassow Robots control software allows users to configure where files are stored (Programs, Workcells and CBun Installers). Some of the storage locations are available by default (Tablet and Robot Storage), some require specific HW (USB Drive) and some have to be configured in advance (Google Drive).

Note: Available storage locations depend on specific storage operation.

4.1.1 Tablet Storage

Built-in tablet (teach pendant) storage is designed for storing programs and workcells while the robot is offline (turned off). Ie. tablet storage allows users to manage their programs and workcells without the need to turn on the robot.

4.1.2 Robot Storage

Built-in robot (cabinet) storage is the default location for storing programs and workcells while the robot is online (turned on). Robot storage also contains CBun installers that are provided by Kassow Robots.

4.1.3 USB Drive

USB Drive allows users to attach their own USB Flash Drive devices. Such storage can be used for storing programs and workcells and for installation of CBuns provided by 3rd party.

Note: USB Flash Drive devices are only mounted if they contain one of the following filesystems: exFAT, VFAT, EXT2, EXT3, EXT4 or HFS+.

4.1.4 Google Drive

Google Drive provides cloud storage for programs, workcells and 3rd party CBuns Installers. Google Drive storage has to be configured in advance and requires internet access via WiFi.

4.2 Programs and Workcells

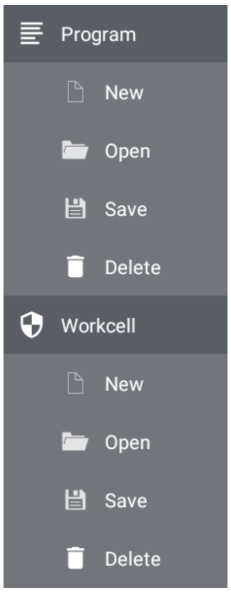
Both programs and workcells can be saved on Tablet Storage, Robot Storage, USB Drive or Google Drive. While files saved on Tablet and Robot Storage are associated with a particular robot, saving a file on USB Drive or Google Drive allows its access on any other robot.

Note: Regular saving of your program and workcell can help prevent data loss.

4.2.1 New File

To create new program or workcell:

1. Open Popup menu
2. Click **New**
3. Confirm by clicking **Yes**

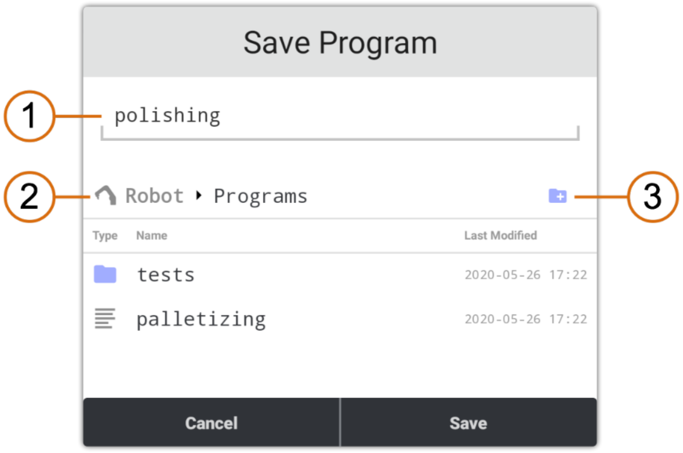


Warning: Any unsaved changes will be lost.

4.2.2 Save File

To save your program or workcell:

1. Open Popup menu
2. Click **Save**
3. Confirm by clicking **Yes**
4. Enter name, select location and click **Save**



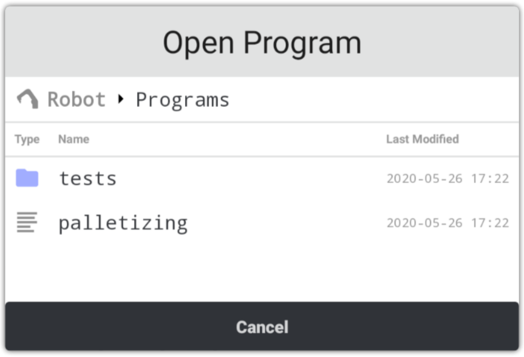
Item	Description
1	**File Name** Unique file identification in selected folder. File with same name will be overwritten.
2	**Storage Navigation** Displays selected storage location. Clicking the icon provides navigation through available storage locations.

3	**New Folder** Allows users to create new folder inside the selected folder.

4.2.3 Open File

To open your program or workcell:

1. Open Popup menu ⋮
2. Click **Open**
3. Select location and click the file to be opened



Open Program

Robot ▸ Programs

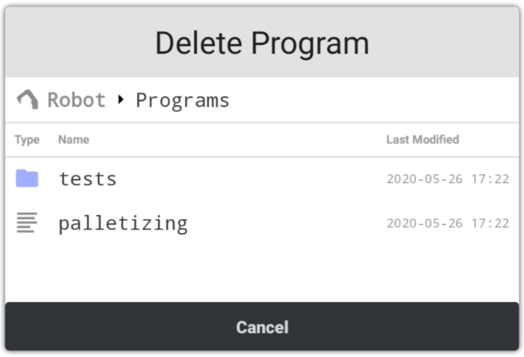
Type	Name	Last Modified
📁	tests	2020-05-26 17:22
≡	palletizing	2020-05-26 17:22

Warning: Any unsaved changes will be lost.

4.2.4 Delete File

To delete your program or workcell:

1. Open Popup menu ⋮
2. Click **Delete**
3. Select location and click the file to be deleted



Delete Program

Robot ▸ Programs

Type	Name	Last Modified
📁	tests	2020-05-26 17:22
≡	palletizing	2020-05-26 17:22

5 Basic Components

The Programming chapter describes the process of program development and the execution of such program by robot. At the beginning the program variables and commands are described as the basic element for program development. Later the control of program execution is explained as well as program debugging. In the end, the program storage is introduced to allow the user storing and opening his/her programs.

5.1 Variables

Variables are one of the basic elements of each programming language. There are two different types of program variables. Both variable types can contain different data type.

Variables Type

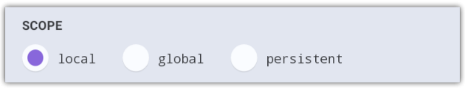
- Custom Variable
- System Variable

Data Type

- Number
- Pose
- Load
- Array
- GridPattern

5.1.1 Custom Variables

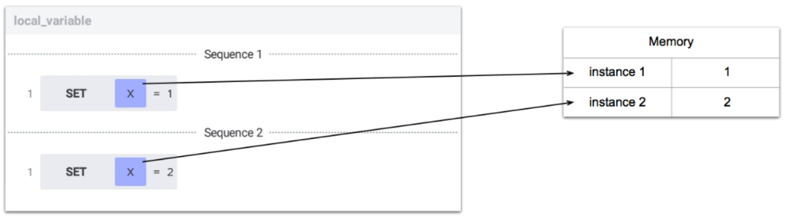
The custom variables are the ones defined by user. That means that the data type is selected by the user and also the default value is declared by user. The value of a custom variable is permanent unless the user changes it via Options panel or SET program command (see 5.2.7). Unlike the system variables the user can define the scope of custom variables via the variable options tool. The custom variable scope can be set as Local, Global or Persistent.


SCOPE
[✓ local] global persistent

Local Scope

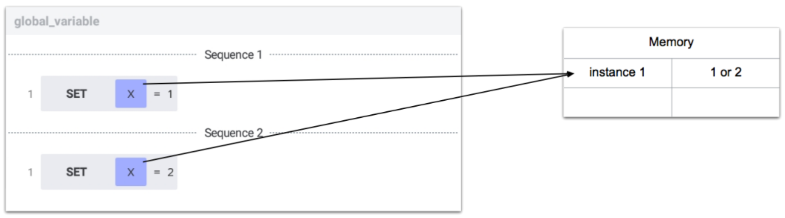
Local variable exists in each sequence individually. Therefore, the value of local variable is not shared across the program sequences. This means, that the local variable can hold different value in every sequence and the instances of such local variable are fully independent.

Local variables must be used inside a FOR loop, otherwise the program execution could be inconsistent and unpredictable. It is also prohibited to use local variable as a part of some global variable (such as reference in pose variable). This type of variable is stored as a part of Program.



Global Scope

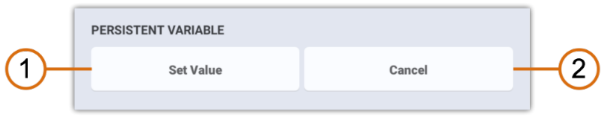
There is just single instance of global variable across the all sequences. This means that the value set by one sequence can be read by other sequence and vice versa. Since each sequence can change the value asynchronously, the result is less predictable than for local variable. On the other hand, global variables can be used for program synchronization, since one sequence can wait for value set by other sequence.



This type of variable is stored as a part of Program.

Persistent Scope

Persistent variable is similar to a global variable (ie. there is just single instance across all sequence), but unlike the global variable, the persistent variable exists outside the program execution and its value is recovered after system restart. Persistent variables are always stored as a part of Workcell.

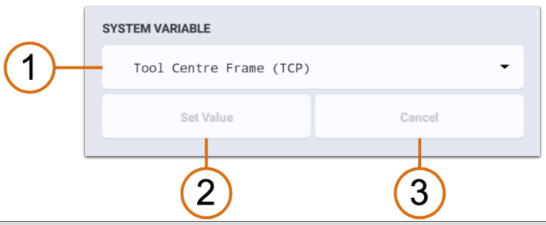


Item	Description
1	**Set Value** Updates the persistent value from local changes (if not applied yet).
2	**Cancel** Reverts the local value, ie. current persistent value is shown.

5.1.2 System Variables

The system variables are created by mapping some real system property (such as system loads, frames or I/O) into variable. The system variable data type is defined by the mapped system property as well as the default value. The variable scope cannot be set, since the mapped value is always the same in every sequence.

Options tool of each system variables displays the section called SYSTEM VARIABLE. This section allows the user to select the mapped system value (such as TCP), confirm the set value action or cancel the value editation.



Item	Description
1	**System Property Spinner** Allows the user to select the system property mapped by this variable.
2	**Set Value Button** Updates the system property to the user defined value. This button is enabled only for ouput system variable (see bellow).
3	**Cancel Button** Cancels the value editation. The user defined values are deleted, and current system value is shown instead.

There are two types of system variables: Input and Output. The type depends on the properties of mapped system value and determines wether the variable value can be set by user.

Input System Variable

The input system variable maps the system value that behaves as an input (such as input port). This means that the value can be only read but cannot be set either via Options tool or SET command.

Output System Variable

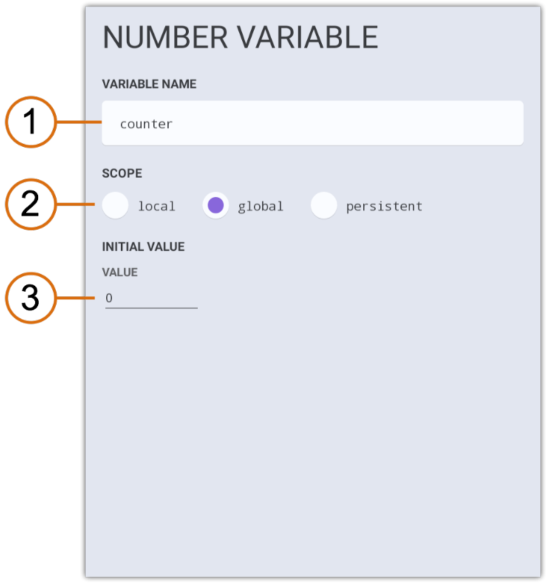
The output system variable mapes the system value that behaves as an output (such as output port). This means that the value can be read and also write either via Options tool or SET command.

5.1.3 Number

The number data type stores numeric value. The number data type is quite complex type, since it can store integer (positive and negative whole number) or floating-point number (real number). When used in expression, the number can behave also as a Boolean data type (0 = False, True otherwise).

Options Panel

The custom variable that stores the Number data type has the following options panel. System variable does not contain SCOPE section but contains SYSTEM VARIABLE section. Persistent variable involves PERSISTENT VARIABLE section.


NUMBER VARIABLE

VARIABLE NAME

counter

SCOPE

local [✓ global] persistent

INITIAL VALUE

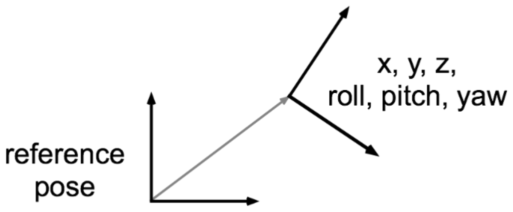
VALUE

0

Item	Description
1	**Variable Name** Displays the user defined variable name (alias). Clicking the field opens the *“Rename Existing Variable“* dialog.
2	**Variable Scope** Determines wether the variable is Local, Global or Persistent (see 5.1.1).
3	**Initial Value** Allows the user to set the default numeric value that will be used at the beginning of program execution (Custom Variable – Local or Global scope) or read/write the current value (Custom Variable – Persistent Scope, System Variable).

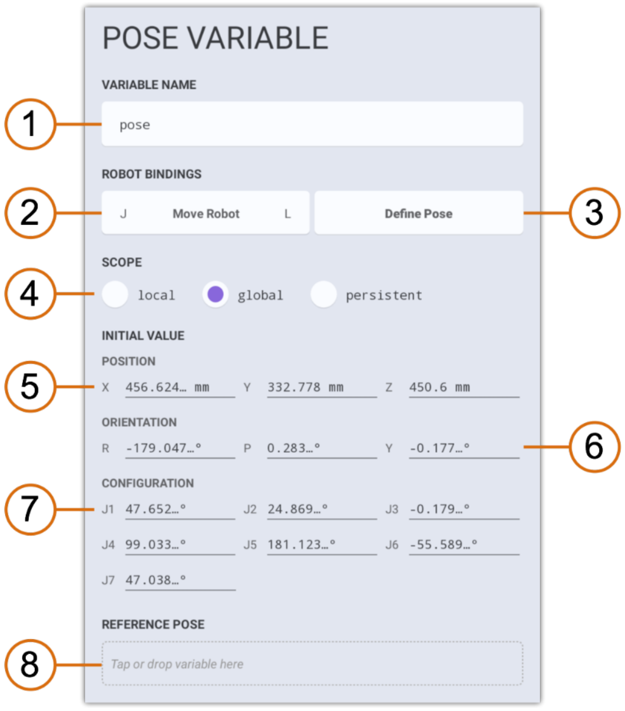
5.1.4 Pose

The pose data type stores position (x, y, z), rotation (roll, pitch, yaw), joints configuration (optional) and reference pose variable (optional). The pose data type provides great functionality since it can represent a robot pose (incl. joints configuration), a single frame or a chain of multiple frames.



Options Panel

The custom variable that stores the Pose data type has the following options panel. System variable does not contain SCOPE section but contains SYSTEM VARIABLE section.



POSE VARIABLE

VARIABLE NAME
pose

ROBOT BINDINGS
J Move Robot L | Define Pose

SCOPE
local [✓ global] persistent

INITIAL VALUE
POSITION
X 456.624… mm Y 332.778 mm Z 450.6 mm

ORIENTATION
R -179.047…° P 0.283…° Y -0.177…°

CONFIGURATION
J1 47.652…° J2 24.869…° J3 -0.179…°
J4 99.033…° J5 181.123…° J6 -55.589…°
J7 47.038…°

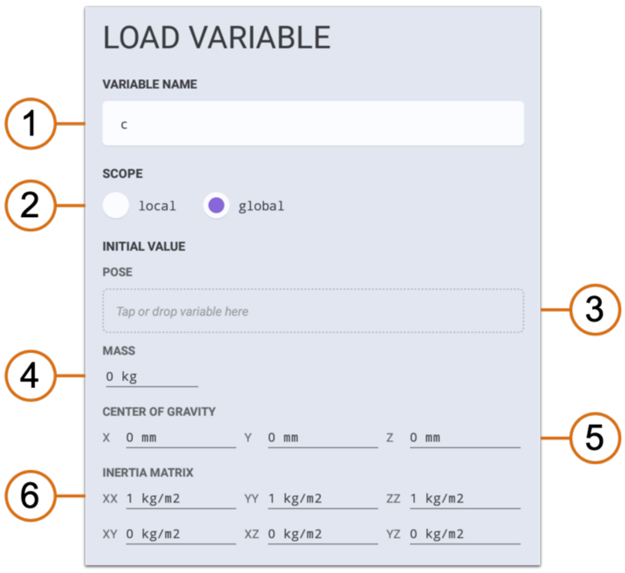
REFERENCE POSE
Tap or drop variable here

Item	Description
1	**Variable Name** Displays the user defined variable name (alias). Clicking the field opens the *“Rename Existing Variable“* dialog.
2	**Move Robot to Pose Button** Pressing the button makes the robot move toward the specified pose either in Joint Space (J) or Work Space (L).
3	**Define Pose Button** Clicking the button opens the *“Define Pose Dialog“*, that allows the user to store current robot pose into the variable.

4	**Variable Scope** Determines whether the variable is Local, Global or Persistent (see 5.1.1).
5	**Initial Position** Represents frame position (x, y, z) in World Frame or Reference Pose coordinate system (if used).
6	**Initial Orientation** Represents frame orientation (roll, pitch, yaw) in World Frame or Reference Pose coordinate system (if used).
7	**Initial Configuration (Optional)** Stores joints configuration that can be used for movement in Joint Space.
8	**Reference Pose (Optional)** Allows the user to set a custom reference coordinate system. Supports drag drop mechanism and *"Create New Variable"* dialog.

5.1.5 Load

The load data type represents a rigid body defined by its mass, center of gravity (x, y, z) and inertia matrix (Ixx, Iyy, Izz, Ixy, Ixz, Iyz) that can be attached to a pose (optional). The load data type expresses that some object is attached to a robot part.


LOAD VARIABLE

VARIABLE NAME: c

SCOPE: local [✓ global]

INITIAL VALUE

POSE: Tap or drop variable here

MASS: 0 kg

CENTER OF GRAVITY: X 0 mm, Y 0 mm, Z 0 mm

INERTIA MATRIX: XX 1 kg/m2, YY 1 kg/m2, ZZ 1 kg/m2, XY 0 kg/m2, XZ 0 kg/m2, YZ 0 kg/m2

Options Panel

The custom variable that stores the Load data type has the following options panel. System variable does not contain SCOPE section but SYSTEM VARIABLE.

Item	Description
1	**Variable Name** Displays the user defined variable name (alias). Clicking the field opens the *"Rename Existing Variable"* dialog.

2	**Variable Scope** Determines whether the variable is Local, Global or Persistent (see 5.1.1).
3	**Initial Pose** (Optional) Allows the user to attach the load to a selected pose. Supports drag&drop mechanism. Clicking the field opens *"Create New Variable"* dialog.
4	**Initial Mass** Represents mass of the load.
5	**Initial Centre of Gravity** Represents position (x, y, z) of centre of gravity inside the World Frame or Pose coordinate system (if used).
6	**Initial Inertia Matrix** Allows the user to set the members of inertia matrix.

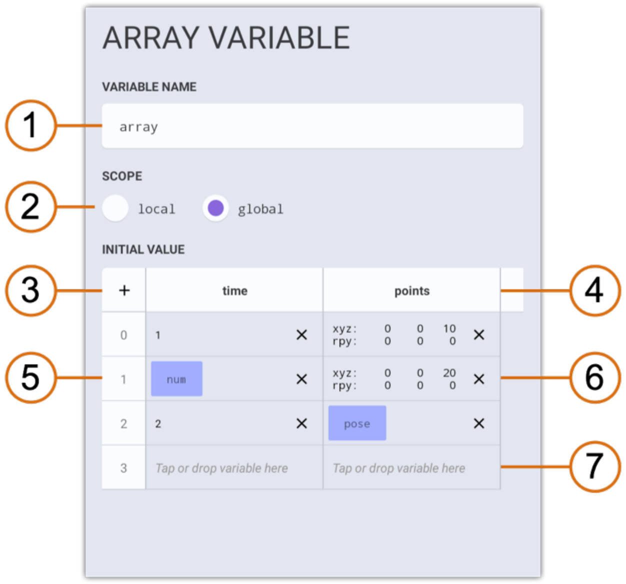
5.1.6 Array

The array data allows to store multiple values in tabular form. Each array should consist of one or more columns, where each column has a specific data type (note that multiple columns can have same data type).

This means that the user can store multiple Poses, Loads, Numbers or GridPatterns in single variable without the need to create many different variables. Array variable can be automatically iterated by FOR command.

Options Panel

The custom variable that stores the Array data type has the following options panel.


ARRAY VARIABLE

VARIABLE NAME: array

SCOPE: local [✓ global]

INITIAL VALUE

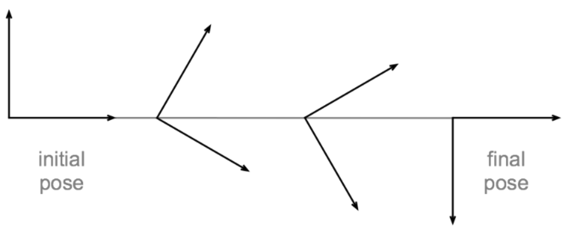
+	time	points
0	1 ✕	xyz: 0 0 10 rpy: 0 0 0 ✕
1	num ✕	xyz: 0 0 20 rpy: 0 0 0 ✕
2	2 ✕	pose ✕
3	*Tap or drop variable here*	*Tap or drop variable here*

Item	Description
1	**Variable Name** Displays the user defined variable name (alias). Clicking the field opens the “*Rename Existing Variable*“ dialog.
2	**Variable Scope** Determines whether the variable is Local, Global or Persistent (see 5.1.1).
3	**Add Column Button** Allows the user to add a new column and specify its name and data type. Note that the column is added at the end of the array.
4	**Column Header** Shows user defined column name.
5	**Row Header** Shows automatically generated row index.
6	**Cell Content** Represents cell content. Clicking the content shows its value in separate options panel.
7	**Empty Cell** Represents cell without content. Supports drag drop of variables. Clicking the empty cell creates new content with default value (or actual robot pose in case of Pose variable).

5.1.7 Grid Pattern

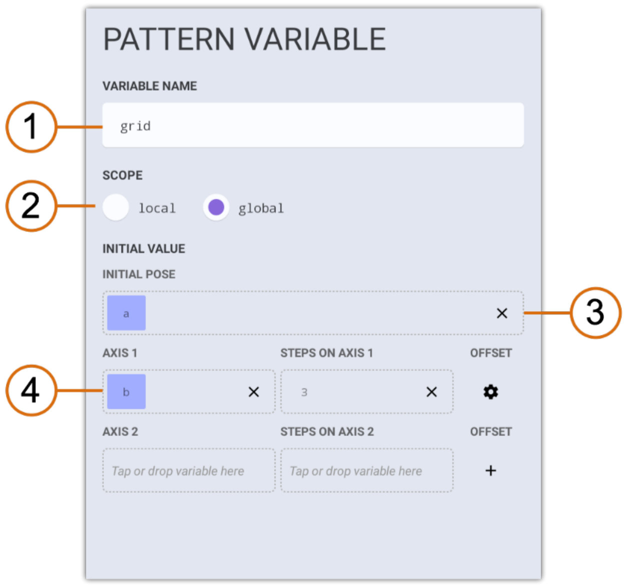
The grid pattern data type generates poses in 1D (1 axis defined), 2D (2 axes defined) or 3D (3 axes defined) grid. This means that the user can generate a lot of poses in a regular pattern using just few poses. Each grid pattern stores initial pose (origin) and up to 3 axes. Pattern variable can be automatically iterated by FOR command.

The following example shows the behaviour of 1D grid pattern (single axis) with grid divided into 4 steps.



Options Panel

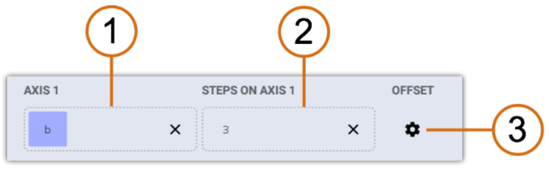
The custom variable that stores the Grid Pattern data type has the following options panel.



Item	Description
1	**Variable Name** Displays the user defined variable name (alias). Clicking the field opens the „Rename Existing Variable“ dialog.
2	**Variable Scope** Determines whether the variable is Local, Global or Persistent (see 5.1.1).
3	**Initial Pose** Determines the origin of Grid Pattern.
4	**Axis Definition** Allows the user to define a single axis. Each axis is defined by its final pose, number of steps and optional offset.

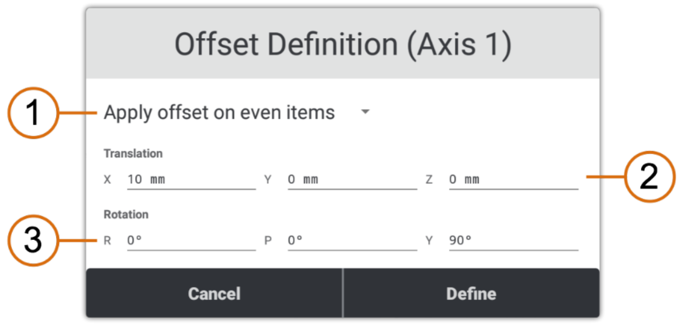
Axis Definition

Each grid axis must define final pose and number of steps. The offset definition is optional.



Item	Description
1	**Final Pose** Determines the final pose of related axis. Supports drag drop of variables (Pose).

2	**Number of Steps** Allows the user to define the number of steps. Supports drag&drop of variables (Number).
3	**Offset** Opens the “Offset Definition” dialog.



Apply offset on even items

Translation

X 10 mm Y 0 mm Z 0 mm

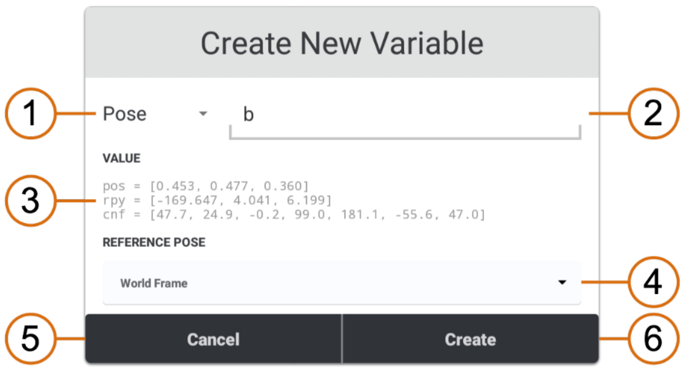
Rotation

R 0° P 0° Y 90°

Item	Description
1	**Offset Applicationm** Allows the user to define whether the offset should be applied always, never or on even or odd items.
2	**Translation** Defines the translation of specified items.
3	**Rotation** Defines the rotation of specified items.

5.1.8 Create New Variable

There are 3 ways to create a new custom variable. The user can create a variable via Bottom Bar (Program Mode), Variables view or drag drop boxes included in Options panel of some commands or variables. In all cases the ”*Create New Variable*“ dialog appears.



Pose b

VALUE

```
pos = [0.453, 0.477, 0.360]
rpy = [-169.647, 4.041, 6.199]
cnf = [47.7, 24.9, -0.2, 99.0, 181.1, -55.6, 47.0]
```

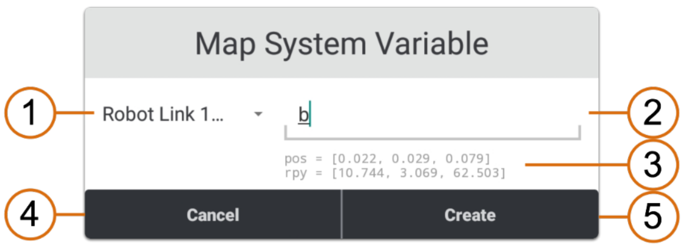
REFERENCE POSE

World Frame

Item	Description
1	**Data Type Spinner** Selects the variable data type (Number, Pose, Load). More information can be found in 5.1.3.
2	**Variable Name** Allows the user to define the variable name (alias).
3	**Default Value** Displays default (initial) value of each variable data type. If the robot is turned on and connected, the default pose value is obtained from real robot pose.
4	**Reference Pose Spinner** (only for Pose data type) Allows the user to express the real robot pose coordinates in selected reference coordinate system. Available only if the robot is turned on and connected.
5	**Cancel Button** Interrupts the process of creation a new variable.
6	**Create Button** Creates new custom variable with selected data type and name. The Create button is enabled only when some unique variable name is entered.

5.1.9 Map System Variable

There are 2 ways to map a new system variable. The user can map a system variable via Variables view or I/O tool. In first case the "*Map System Variable*" dialog appears allowing the user to map all system values. As for the I/O tool, the "*Map I/O Variable*" dialog appears that allows the user to map input or output ports. System Variables can be mapped only if the robot is turned on and connected.



```
pos = [0.022, 0.029, 0.079]
rpy = [10.744, 3.069, 62.503]
```

Item	Description
1	**System Value Spinner** Selects the system value that will be mapped into this variable. More information can be found in 5.1.2.
2	**Variable Name** Allows the user to define the variable name (alias).
3	**Default Value**

	Displays default (initial) value of each system value.
4	**Cancel Button** Interrupts the process of mapping a new system variable.
5	**Create Button** Maps the selected system value into a variable. The Create button is enabled only when some unique variable name is entered.

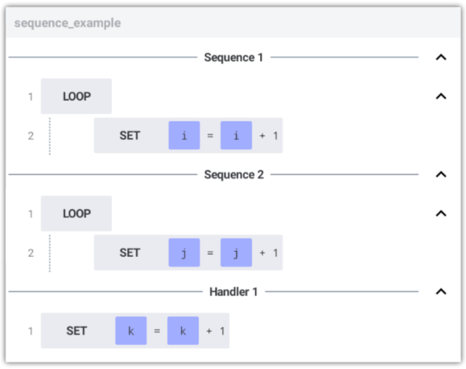
5.2 Basic commands

Commands are the basic elements for creation of a program tree. All commands are available in the Command Box (see 3.1.3). The detailed description of each command can be found below.

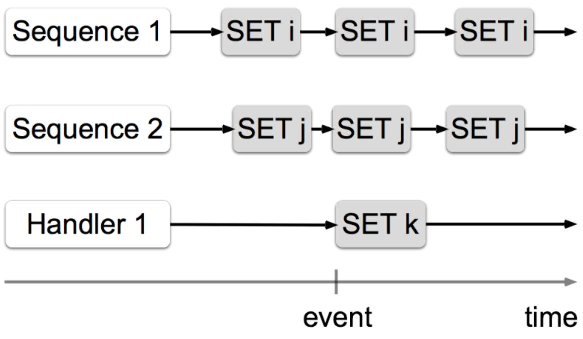
5.2.1 Sequence

The Sequence command is the root command that encapsulates a block of commands to be executed. The sequence execution starts either on program start, or anytime the selected event occurs. Multiple sequences are executed simultaneously and asynchronously and therefore independently. There must always be at least one sequence. The maximum number of sequences included in program tree is also limited to 10.

Example

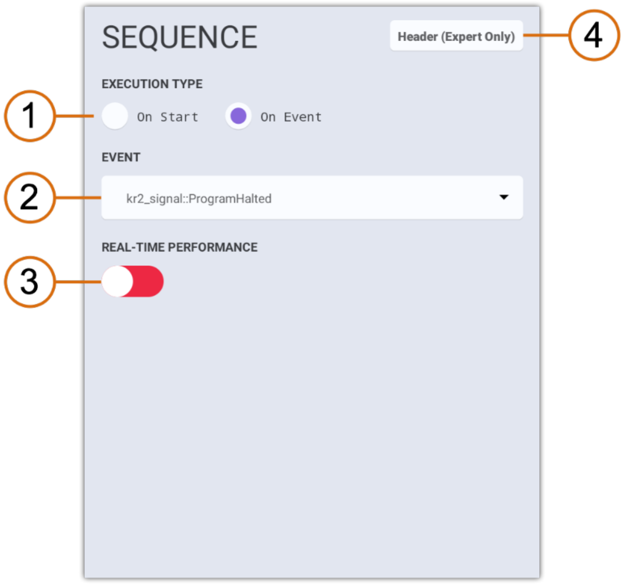


```
sequence_example
Sequence 1
1 LOOP
2     SET i = i + 1
Sequence 2
1 LOOP
2     SET j = j + 1
Handler 1
1 SET k = k + 1
```



Options Panel

The Sequence command cannot be configured so the Options panel is empty.



SEQUENCE

Header (Expert Only)

EXECUTION TYPE

On Start [✓ On Event]

EVENT

kr2_signal::ProgramHalted

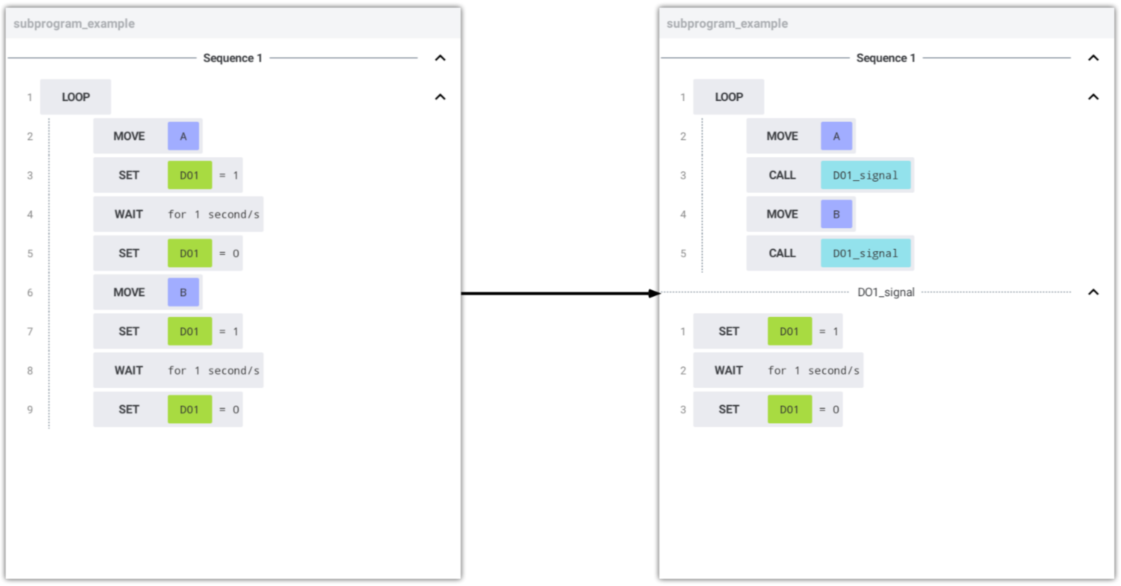
REAL-TIME PERFORMANCE

Item	Description
1	**Execution Type** Determines whether the sequence execution starts on program start or on event.
2	**Event Selection** Specifies the event that will be handled by this sequence. Available only for On Event execution type.
3	**Real-Time Performance** Determines whether the sequence execution is real-time
4	**Header** Specifies which C++ header should be included. This option is intended only for expert users.

5.2.2 Subprogram

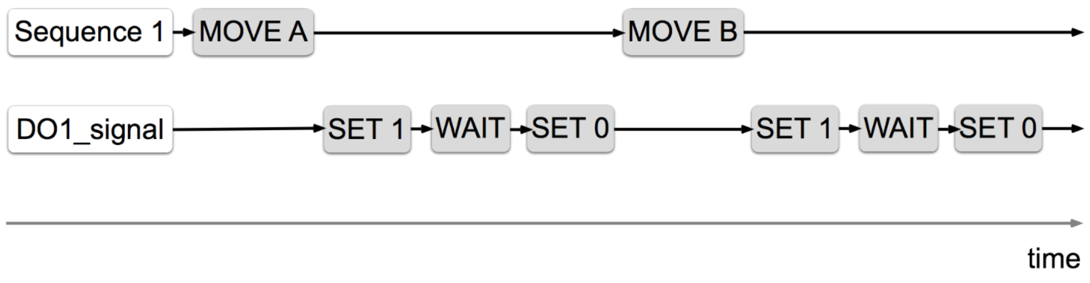
The Subprogram command is the root command that encapsulates a block of commands to be included in one or more locations in Sequence. The Subprogram execution is invoked by CALL command.

Example



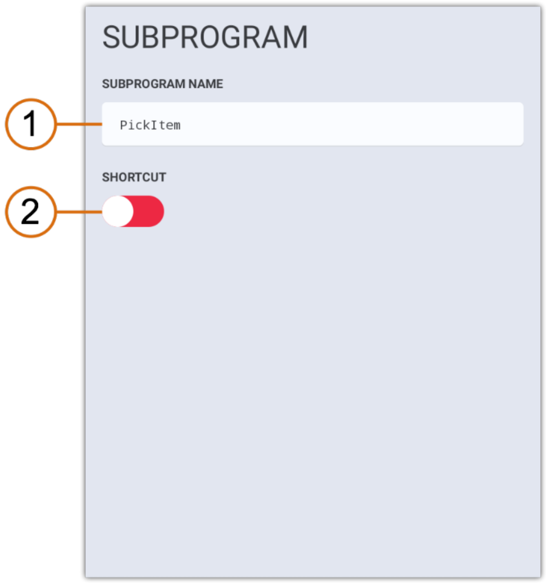
```
subprogram_example
Sequence 1
LOOP
    MOVE A
    SET DO1 = 1
    WAIT for 1 second/s
    SET DO1 = 0
    MOVE B
    SET DO1 = 1
    WAIT for 1 second/s
    SET DO1 = 0
```

```
subprogram_example
Sequence 1
LOOP
    MOVE A
    CALL DO1_signal
    MOVE B
    CALL DO1_signal
DO1_signal
SET DO1 = 1
WAIT for 1 second/s
SET DO1 = 0
```



Options Panel

The Subprogram command has the following Options panel.


SUBPROGRAM

SUBPROGRAM NAME

PickItem

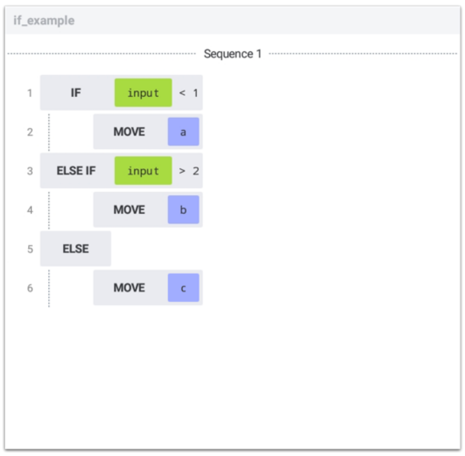
SHORTCUT

Item	Description
1	**Subprogram Name** Allows the user to specify the subprogram name.
2	**Shortcut** Determines whether the subprogram should be listed in Commands Box.

5.2.3 IF

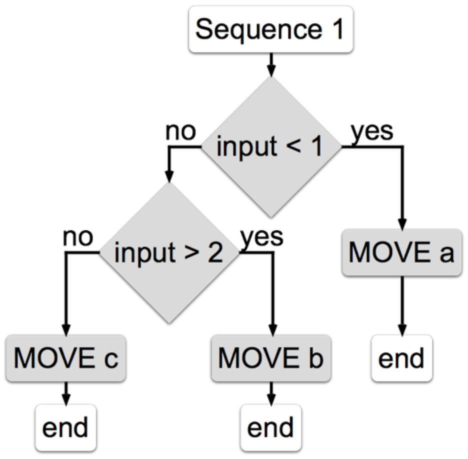
The IF command is one of the control flow commands, which allows the user to control the flow of program execution.

Example


if_example

Sequence 1

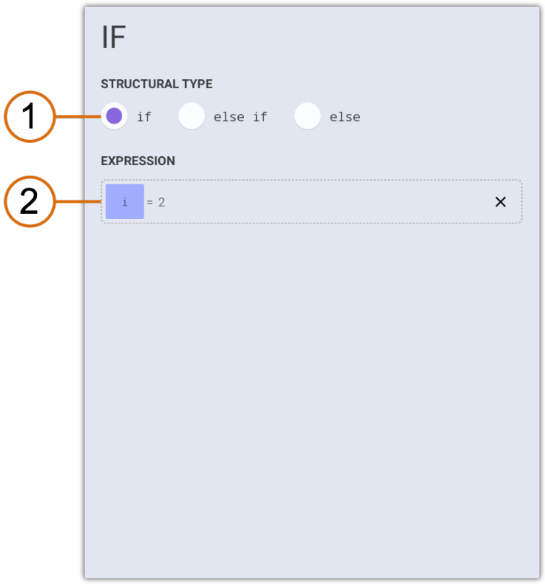
```
1 IF input < 1
2     MOVE a
3 ELSE IF input > 2
4     MOVE b
5 ELSE
6     MOVE c
```



The IF command evaluates user defined condition (expression) and if the condition is met, the nested block of command will be executed. The IF command can support three different modes: IF, ELSE IF and ELSE. Whereas IF command can be used alone, ELSE IF and ELSE can be used only with previous IF command.

Options Panel

The IF command has the following Options panel.

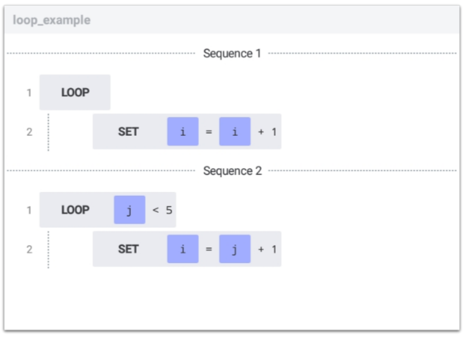


Item	Description
1	**Structural Type** Specifies whether the command behaves as a new IF command or extends the previous IF command as ELSE IF or ELSE command. In case there is not any IF command prior ELSE IF or ELSE command, it behaves as standard IF command.
2	**Expression** An expression is evaluated once the IF command is called. If the result of evaluation is true (or non-zero number), the nested block of commands is executed, otherwise the commands are skipped Note that the expression is not available for ELSE mode. Supports drag drop of variables.

5.2.4 LOOP

The LOOP command is another control flow command that allows the user to repeat block a of nested commands forever or while the user condition evaluated as true. The LOOP command also supports synchronization at chosen frequency.

Example

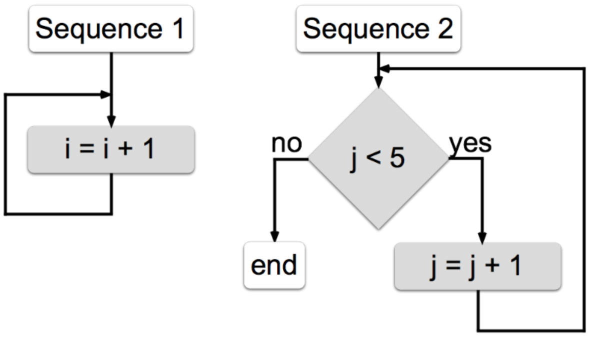

loop_example

Sequence 1

1 LOOP
2 SET i = i + 1

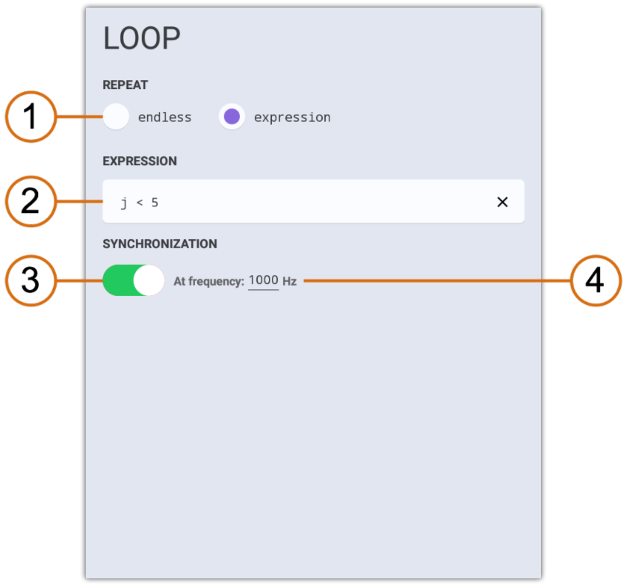
Sequence 2

1 LOOP j < 5
2 SET i = j + 1



Options Panel

The LOOP command has the following Options panel.



LOOP

REPEAT

endless [✓ expression]

EXPRESSION

j < 5

SYNCHRONIZATION

At frequency: 1000 Hz

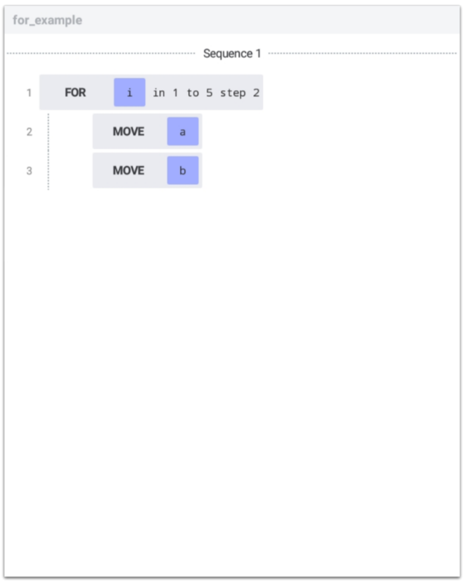
Item	Description
1	**Repeat Condition** Determines whether the loop is endless (repeated forever), or based on expression, i.e.. executed as long as the result of expression evaluation is true (or non zero number).
2	**Expression** An expression is evaluated on every LOOP cycle (at the beginning of cycle). If the condition is met, the nested block of commands is executed, otherwise the loop is terminated. Note that the expression is available only if repeat – expression is selected.

3	**Synchronization Switch** Enables the loop synchronization. When the synchronization is disabled, the loop is executed at maximum possible frequency. This can have a negative impact on performance and should be used only by experienced users.
4	**Synchronization Frequency** Specifies the loop synchronization frequency up to 1000 Hz. This field is available only when Synchronization Switch is ON.

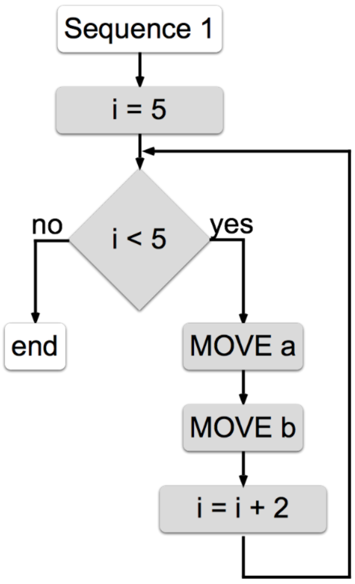
5.2.5 FOR

The FOR command is another control flow command that allows the user to repeat block of nested commands for a specific number of iterations. The FOR command provides wide spectrum of options, since it can store the iteration result inside the iteration variable, start and stop the iteration at certain value, or use custom increment. FOR command can be also used for automatic iteration through Pattern of Array variables.

Example

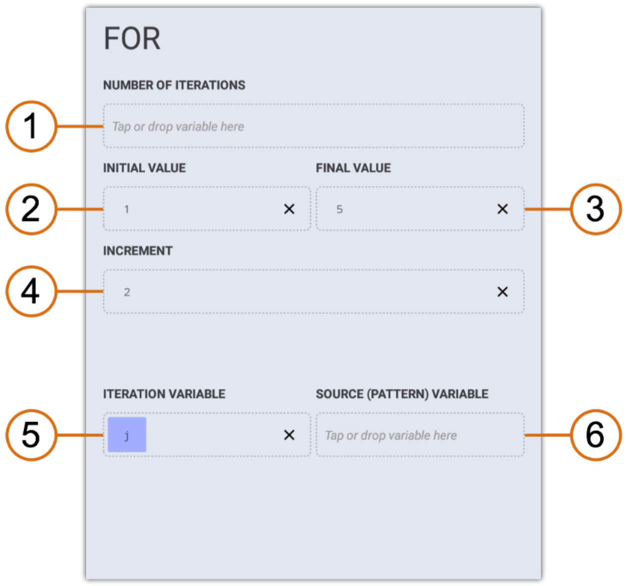


```
FOR i in 1 to 5 step 2
    MOVE a
    MOVE b
```



Options Panel

The FOR command has the following Options panel.



Item	Description
1	**Number of Iterations** Determines how many times the for loop should be executed. Supports drag drop of variables (Number) and constant values.
2	**Initial Value** Allows the user to set the initial value (0 by default). Supports drag drop of variables (Number) and constant values.
3	**Final Value** Allows the user to set the final value. Supports drag drop of variables (Number) and constant values.
4	**Increment** Sets the iteration increment (default 1). Supports drag drop of variables (Number) and constant values.
5	**Iteration Variable** Stores the output of iteration process. Supports drag drop of variables (Number and Local scope). Note that the iteration variable data type has to be compatible with the Source Variable data type.
6	**Source Variable** Source Variable provides the output value of FOR cycle based on the current iteration index. By default (when empty), the Source Variable is a continuous sequence of numbers (0, 1, 2, 3, etc.). Supports drag drop of variables (Pattern and Array).

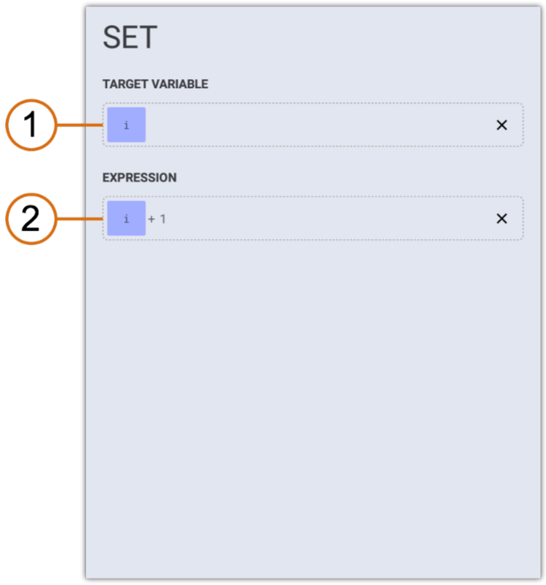
5.2.6 MOVE, STOP, RESUME

Motion commands are the basic building blocks for any robot application. Because of the complexity of this group of control commands there the whole subchapter 5.3 is dedicated to it.

5.2.7 SET

The SET command assigns new value (result of the expression) into the Target Variable. This allows the user to change the value stored in a variable during the program execution. If the target variable is system variable (output type), the SET command influences also the mapped value (it can for example activate the digital output on I/O board).

Options Panel

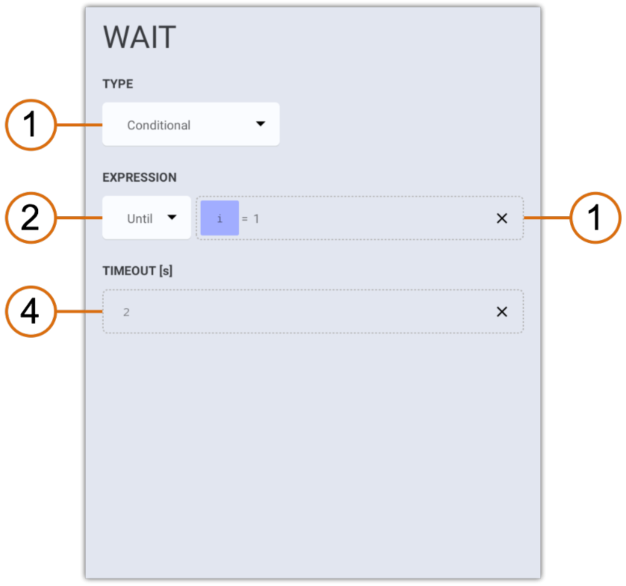


Item	Description
1	**Target Variable** Stores the result of the expression, i.e. represents left side of assignment equation. Supports drag drop of variables (any data type) and clicking the field opens expression builder.
2	**Expression** An expression is evaluated once the SET command is called. The result of evaluation is assigned into the Target Variable. Beware that the result of the evaluation must have the same data type as the Target Variable. Supports drag drop of variables.

5.2.8 WAIT

The WAIT command makes the program to sleep either for a specific amount of time (*Time Instant*) or until the condition is met (*Conditional*). In both cases the expression is necessary to specify the amount of time or condition. The *Conditional* option together with shared variables provides great way for synchronization of multiple sequences.

Options Panel



Item	Description
1	**Type Spinner** Determines whether the WAIT command should sleep for specific amount of time (Time Instant) or until the condition is met (Conditional).
2	**Condition** Determines whether the WAIT Conditional command should wait Until expression becomes true or While expression is true. This setting is available only if *Conditional* type is selected.
3	**Expression** An expression is evaluated continuously in a loop once the WAIT command is called. When the condition is met, i.e.. the result of the evaluation is true (or non-zero number), the WAIT command is unblocked, and any next command will be executed. Supports drag drop of variables.
4	**Timeout** The WAIT Conditional command can be unblocked also if specified timeout occurs.

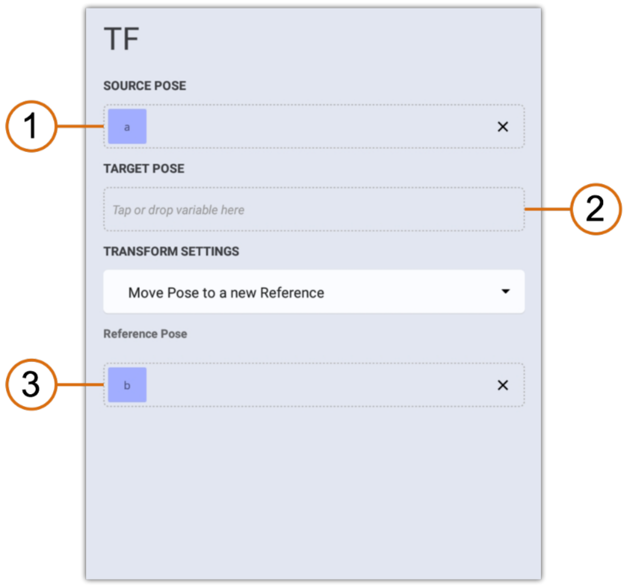
5.2.9 TF

The TF command transforms the pose and stores the new pose in the Source Pose variable or into Target Pose variable (if used). There are 3 transform types supported: Move Pose to a new Reference, Transform Pose (translate, rotate or both) and Convert Pose to a new Reference.

Move Pose to a new Reference

This type of transformation changes the reference pose of Source Pose and stores the result in Target Pose (if available) or overrides the Source Pose (if Target Pose not available). The Source Pose coordinates (x, y, z, roll, pitch, yaw) do not change although they are expressed in the coordinate system of Reference Pose.

Move Pose to a new Reference Options Panel



TF

SOURCE POSE

a

TARGET POSE

Tap or drop variable here

TRANSFORM SETTINGS

Move Pose to a new Reference

Reference Pose

b

Item	Description
1	**Source Pose** Stores the source coordinates to be used in the Reference Pose coordinate system. If *Target Pose* is not available, the result is stored into Source Pose. Supports drag drop of variables (pose data type). Clicking the field opens “*Create New Variable”* dialog.
2	**Target Pose** (Optional) Determines the output variable. The result of transformation is stored in the *Target Pose* (if available). Supports drag drop of variables (pose data type). Clicking the field opens *‘Create New Variable”* dialog.
3	**Reference Pose** Allows the user to set the new reference pose. Supports drag drop of variables (pose data type). Clicking the field opens *“Create New Variable”* dialog.

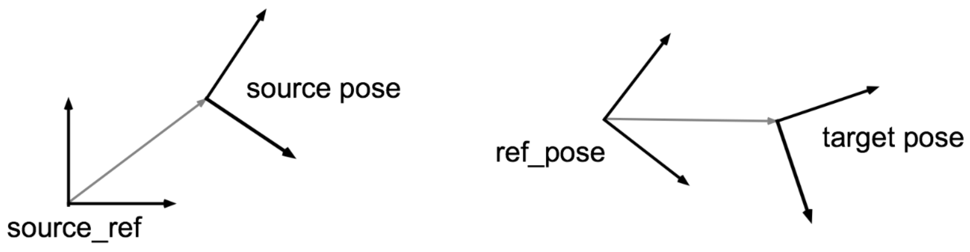
Move Pose to a new Reference Example

The following example shows the typical usage of a Move Pose to a new Reference function. The target pose coordinates (x, y, z, roll, pitch, yaw) are the same as source, but the target reference pose is changed.

Input	Value

Source Pose	x = 50 mm, y = 50 mm, z = 0 mm
	roll = 0°, pitch = 0°, yaw = 45°
	reference pose = source_ref

Target Pose	Any pose variable
Reference Pose	ref_pose



Input	Value

Source Pose	x = 50 mm, y = 50 mm, z = 0 mm
	roll = 0°, pitch = 0°, yaw = 45°
	reference pose = source_ref

Target Pose	x = 50 mm, y = 50 mm, z = 0 mm
	roll = 0°, pitch = 0°, yaw = 45°
	reference pose = ref_pose

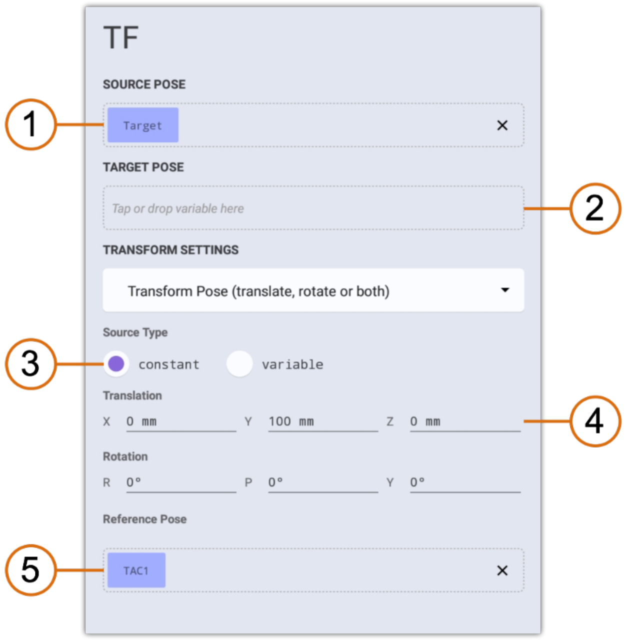
Reference Pose	ref_pose

Transform Pose (translate, rotate or both)

This type of transformation transforms the pose by constant value (x, y, z, roll, pitch, yaw) or using a content of pose variable. If the Reference Pose is not available, the transformation is provided in the coordinate system of Source Pose reference (in case of Constant transformation) or in the coordinate system of Transform Variable (in case of Variable transformation). Otherwise, the transformation is provided in the Reference Pose coordinate system.

Transform Pose (Constant) Options Panel

The Transform Pose (Constant) TF configuration has the following Options panel.



Item	Description
1	**Source Pose** Stores the source coordinates to be transformed using the constant translation and rotation. If Target Pose is not available, the result is stored in the Source Pose. Supports drag drop of variables (pose data type). Clicking the field opens *“Create New Variable”* dialog.
2	**Target Pose** (Optional) Determines the output variable. The result of transformation is stored in the Target Pose (if available). Supports drag drop of variables (pose data type). Clicking the field opens *“Create New Variable”* dialog.
3	**Source Type** Determines whether the transformation is defined by constant translation and rotation or pose variable. This example shows the Constant Source Type case.
4	**Constant Pose** Defines the translation (x, y, z) and rotation (roll, pitch, yaw) to be applied on the source pose in its reference coordinate system or in the coordinate system of reference pose (if available).
5	**Reference Pose** (Optional) Allows the user to set the reference pose of the transformation. Supports drag drop of variables (pose data type). Clicking the field opens *“Create New Variable“* dialog.

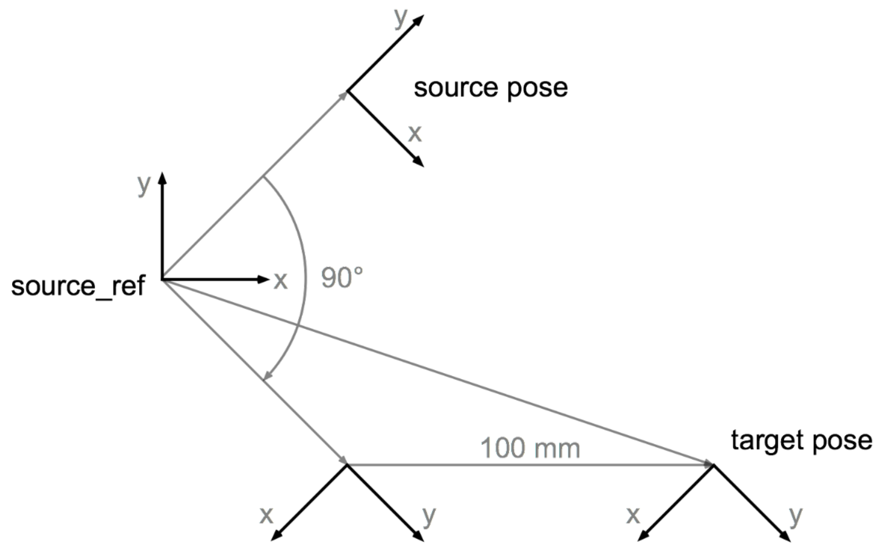
Transform Pose (Constant) Example 1

The following example shows one of the usages of Transform Pose (Constant) function. The target pose coordinates are transformed (translated and rotated) in the coordinate system that is defined by the source pose reference. This coordinate system is used if reference pose is not available.

Input	Value

Source Pose	x = 50 mm, y = 50 mm, z = 0 mm
	roll = 0°, pitch = 0°, yaw = 45°
	reference pose = source_ref

Target Pose	Any pose variable
Source Type	Constant
Transla-tion	x = 100 mm, y = 0 mm, z = 0 mm
Rotation	roll = 0°, pitch = 0°, yaw = 90°
Reference Pose	None



Output	Value

Source Pose	x = 50 mm, y = 50 mm, z = 0 mm
	roll = 0°, pitch = 0°, yaw = 45°
	reference pose = source_ref

Target Pose	x = 150 mm, y = -50 mm, z = 0 mm
	roll = 0°, pitch = 0°, yaw = 135°
	reference pose = source_ref

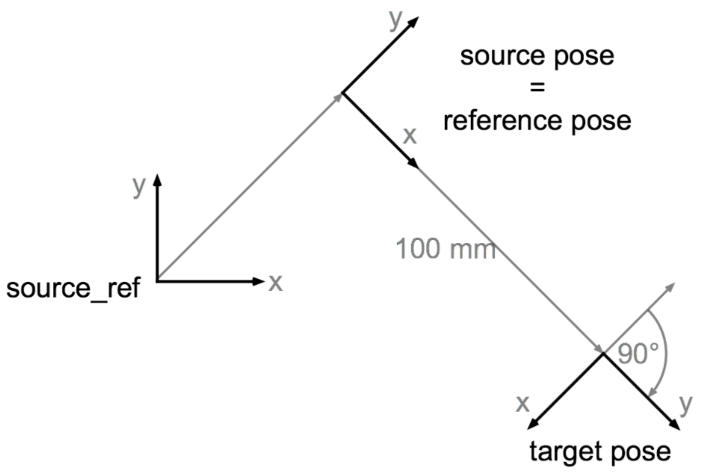
Transform Pose (Constant) Example 2

The following example shows another usage of Transform Pose (Constant) function. The target pose coordinates are transformed (translated and rotated) in the coordinate system of the Reference Pose. This time the Source Pose is used also as a Reference Pose which leads to a useful transformation.

Input	Value

Source Pose	x = 50 mm, y = 50 mm, z = 0 mm
	roll = 0°, pitch = 0°, yaw = 45°
	reference pose = source_ref

Target Pose	Any pose variable
Source Type	Constant
Transla-tion	x = 100 mm, y = 0 mm, z = 0 mm
Rotation	roll = 0°, pitch = 0°, yaw = 90°
Reference Pose	Source Pose

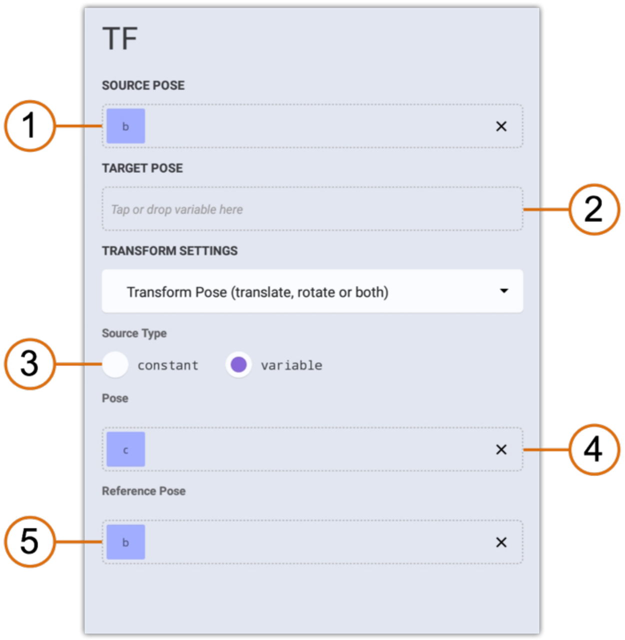


Output	Value

Source Pose	x = 50 mm, y = 50 mm, z = 0 mm
	roll = 0°, pitch = 0°, yaw = 45°
	reference pose = source_ref

Target Pose	x = 100 mm, y = 0 mm, z = 0 mm
	roll = 0°, pitch = 0°, yaw = °
	reference pose = Source Pose

Transform Pose (Variable) Options Panel



TF

SOURCE POSE

b ×

TARGET POSE

Tap or drop variable here

TRANSFORM SETTINGS

Transform Pose (translate, rotate or both)

Source Type

constant [✓ variable]

Pose

c ×

Reference Pose

b ×

Item	Description
1	**Source Pose** Stores the source coordinates to be transformed using the transformation variable. If Target Pose is not available, the result is stored in the Source Pose. Supports drag drop of variables (pose data type). Clicking the field opens *“Create New Variable”* dialog.
2	**Target Pose** (Optional) Determines the output variable. The result of transformation is stored in the Target Pose (if available). Supports drag drop of variables (pose data type). Clicking the field opens *“Create New Variable”* dialog.
3	**Source Type** Determines whether the transformation is defined by constant translation and rotation or pose variable. This example shows the Variable case.
4	**Transformation Variable** Defines the transformation (translation, rotation) to be applied on the source pose in the transformation pose reference coordinate system or in the coordinate system of reference pose (if available).
5	**Reference Pose** (Optional) Allows the user to set the reference pose of the transformation. Supports drag drop of variables (pose data type). Clicking the field opens *“Create New Variable”* dialog.

Transform Pose (Variable) Example 1

The following example shows one of the usages of Transform Pose (Variable) function. The target pose coordinates are transformed (translated and rotated) in the coordinate system that is defined by the transformation pose reference. This coordinate system is used if reference pose is not available.

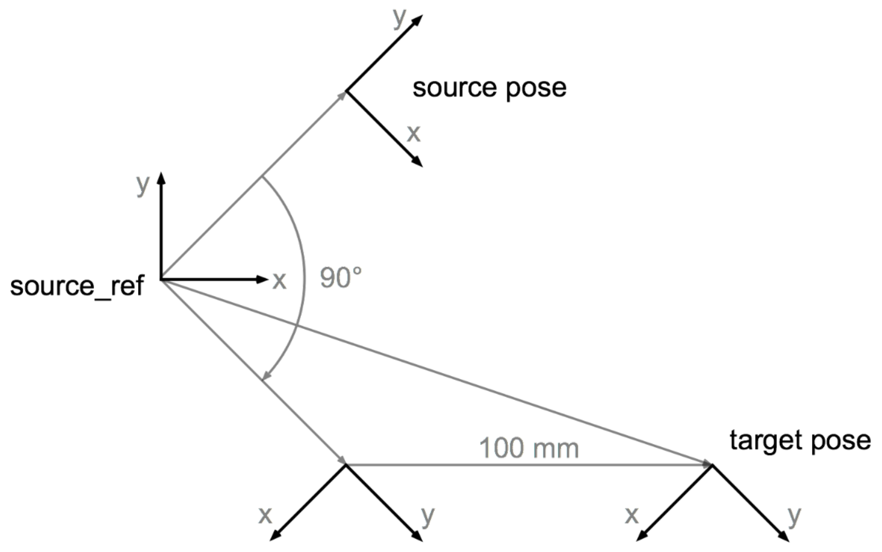
Input	Value

Source Pose	x = 50 mm, y = 50 mm, z = 0 mm
	roll = 0°, pitch = 0°, yaw = 45°
	reference pose = source_ref

Target Pose	Any pose variable
Source Type	Variable

Transfor-mation Pose	x = 100 mm, y = 0 mm, z = 0 mm
	roll = 0°, pitch = 0°, yaw = 90°
	reference pose = source_ref

Reference Pose	None



Output	Value

Source Pose	x = 50 mm, y = 50 mm, z = 0 mm
	roll = 0°, pitch = 0°, yaw = 45°
	reference pose = source_ref

Target Pose	x = 150 mm, y = -50 mm, z = 0 mm
	roll = 0°, pitch = 0°, yaw = 135°
	reference pose = source_ref

Transform Pose (Variable) Example 2

The following example shows another usage of Transform Pose (Variable) function. The target pose coordinates are transformed (translated and rotated) in the coordinate system of the Reference Pose. This time the Source Pose is used also as a Reference Pose which leads to a useful transformation.

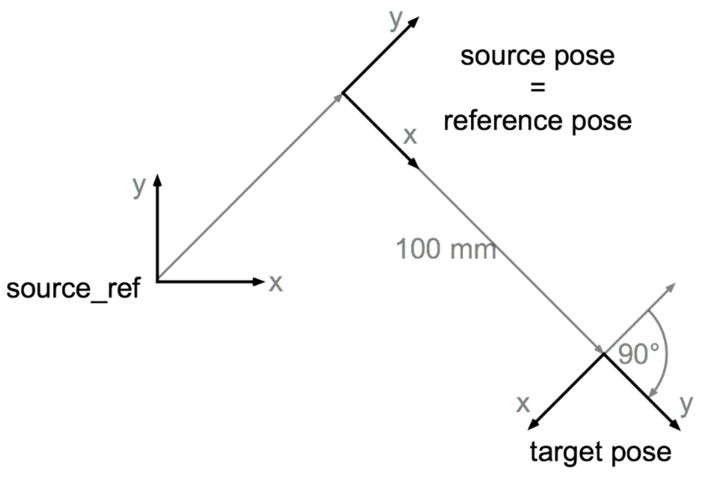
Input	Value

Source Pose	x = 50 mm, y = 50 mm, z = 0 mm
	roll = 0°, pitch = 0°, yaw = 45°
	reference pose = source_ref

Target Pose	Any pose variable
Source Type	Variable

Transfor-mation Pose	x = 100 mm, y = 0 mm, z = 0 mm
	roll = 0°, pitch = 0°, yaw = 90°
	reference pose = source_ref

Reference Pose	Source Pose



Output	Value

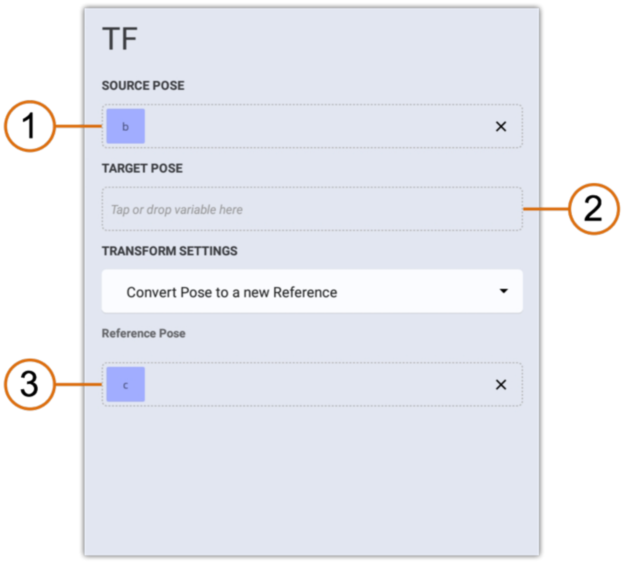
Source Pose	x = 50 mm, y = 50 mm, z = 0 mm
	roll = 0°, pitch = 0°, yaw = 45°
	reference pose = source_ref

Target Pose	x = 100 mm, y = 0 mm, z = 0 mm
	roll = 0°, pitch = 0°, yaw = 90°
	reference pose = source_ref

Convert Pose to a new Reference

This type of transformation changes the reference pose of Source Pose, recalculates the coordinates and stores the result in Target Pose (if available) or overrides the Source Pose (if Target Pose not available).

Transform Pose (Variable) Options Panel



Item	Description
1	**Source Pose** Stores the source coordinates to be recalculated into the Reference Pose coordinate system. If Target Pose is not available, the result is stored into Source Pose. Supports drag drop of variables (pose data type). Clicking the field opens *“Create New Variable”* dialog.
2	**Target Pose** (Optional) Determines the output variable. The result of transformation is stored in the Target Pose (if available). Supports drag drop of variables (pose data type). Clicking the field opens the *“Create New Variable”* dialog.
3	**Reference Pose** Allows the user to set the new reference pose. Supports drag drop of variables (pose data type). Clicking the field opens the *“Create New Variable”* dialog.

Convert Pose to a new Reference Example

The following example shows the typical usage of a Convert Pose to a new Reference function. The target pose coordinates (x, y, z, roll, pitch, yaw) into the reference pose coordinate system.

Input	Value

Source Pose	x = 50 mm, y = 50 mm, z = 0 mm
	roll = 0°, pitch = 0°, yaw = 45°
	reference pose = source_ref

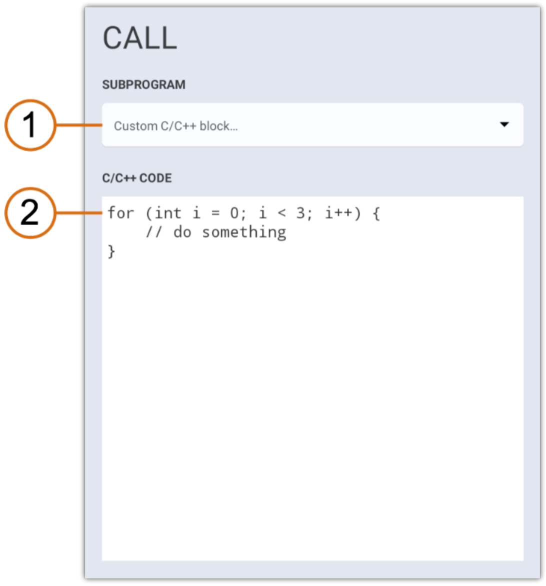
Target Pose	Any pose variable
Reference Pose	ref_pose

5.2.10 Call

The CALL command either executes selected Subprogram or evaluates custom (user defined) C/C++ code (intended for expert users only). C/C++ headers can be included via Sequence command.

Options Panel

The CALL command has the following Options panel.

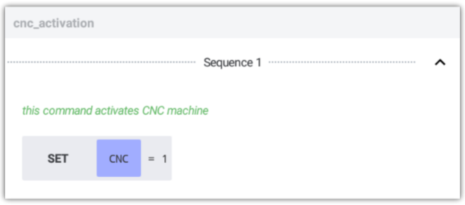


```
for (int i = 0; i < 3; i++) {
    // do something
}
```

Item	Description
1	**Subprogram** Allows the user to select one of the existing subprograms or custom C/C++ code.
2	**C/C++ Code** Custom C/C++ code to be executed at the time of CALL command evaluation. Available only if C/C++ subprogram is selected.

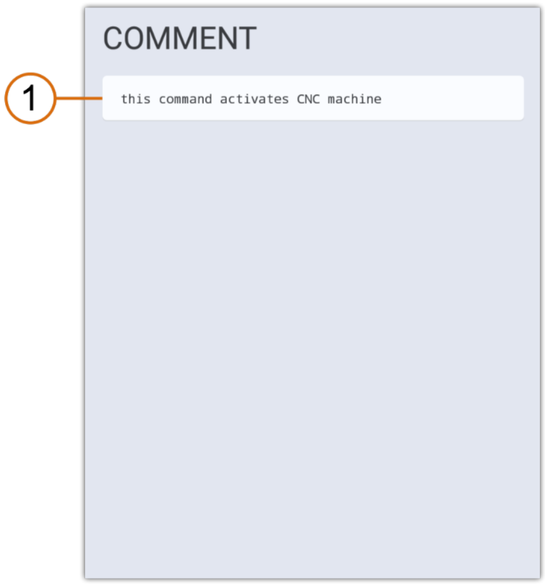
5.2.11 Comment

The Comment command allows the user to add comment for easier orientation in program. Comments do not affect program execution.



Options Panel

The Comment command has the following Options panel.



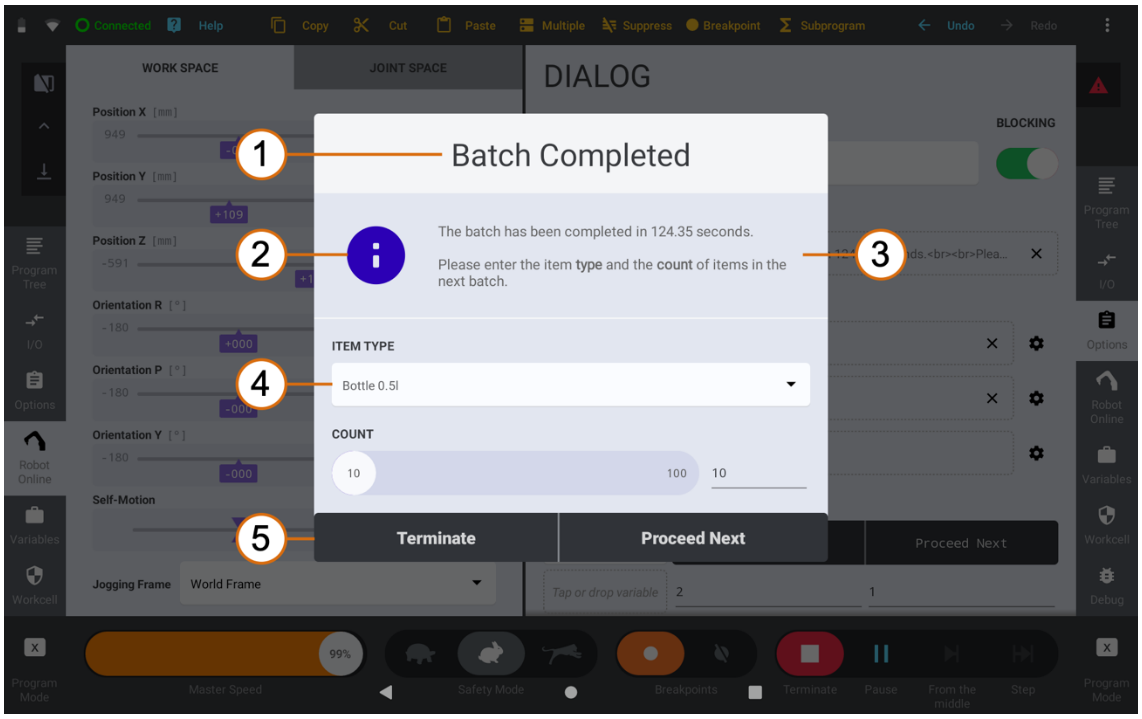
5.2.12 Dialog

The DIALOG command allows to invoke a custom dialog message that will be shown on the teach pendant screen each time the command is called. This custom dialog can serve both as output and input interface of the robot program, ie. it can show (output) message to the user and also collect (input) parameters entered by the user.

The DIALOG content is evaluated at each call of the command; therefore it always shows the output based on the recent data. Note that the dialog content is not evaluated continuously once the dialog is visible.

The custom dialog can be always dismissed by clicking on any of its content buttons. In case the user clicks outside of the dialog, the dialog is just hidden temporarily. Clicking the warning icon at the top right corner of the application will show the dialog again (if it is active).

In addition, the dialog window can be dismissed automatically on signal (with no user interaction). This is suitable for situations when the dialog is shown while some defined conditions are met (such as human presence being detected by a laser scanner). Please note that for the entry variables, values for the used arguments are collected only if the dialog was dismissed by the user.



Batch Completed

The batch has been completed in 124.35 seconds.

Please enter the item type and the count of items in the next batch.

ITEM TYPE: Bottle 0.5l

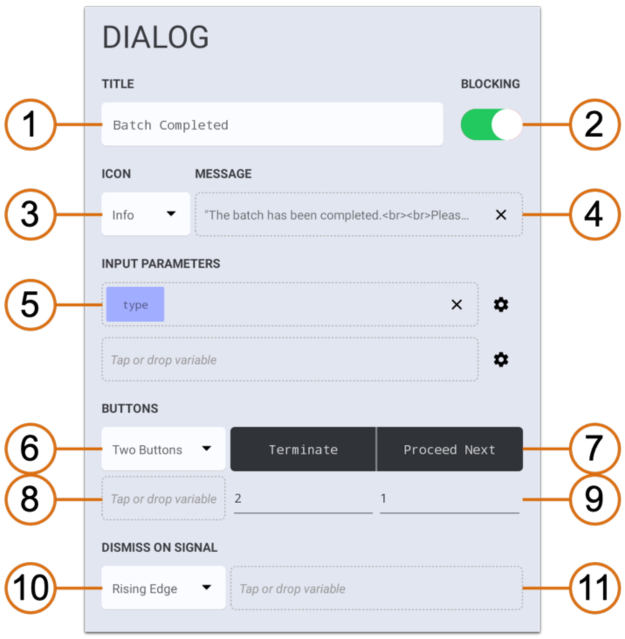
COUNT: 10 — 100 — 10

Figure 1. Custom defined dialog sample

Item	Description
1	**Title** Dialog title should express the dialog purpose.
2	**Icon** (optional) The dialog can contain one of the following icons: Info, Warning or Error.
3	**Message** (optional) Dialog message should provide details of the dialog purpose. Dialog message can also display values of some variables (for example: The batch has been completed in 124.35 seconds).
4	**Input Parameter** (optional) The dialog can contain up to three input parameters. The input parameters are collected once the user clicks any of the control buttons and the collected values are assigned to the appropriate variables.
5	**Control Buttons** The dialog has one to three control buttons. All of the buttons will dismiss (close) the dialog. In addition, a value can be bound to each of the buttons and once the user clicks the button the value will be assigned to the target variable.

Options Panel

Various entry parameters can be available for the custom dialog message composition.

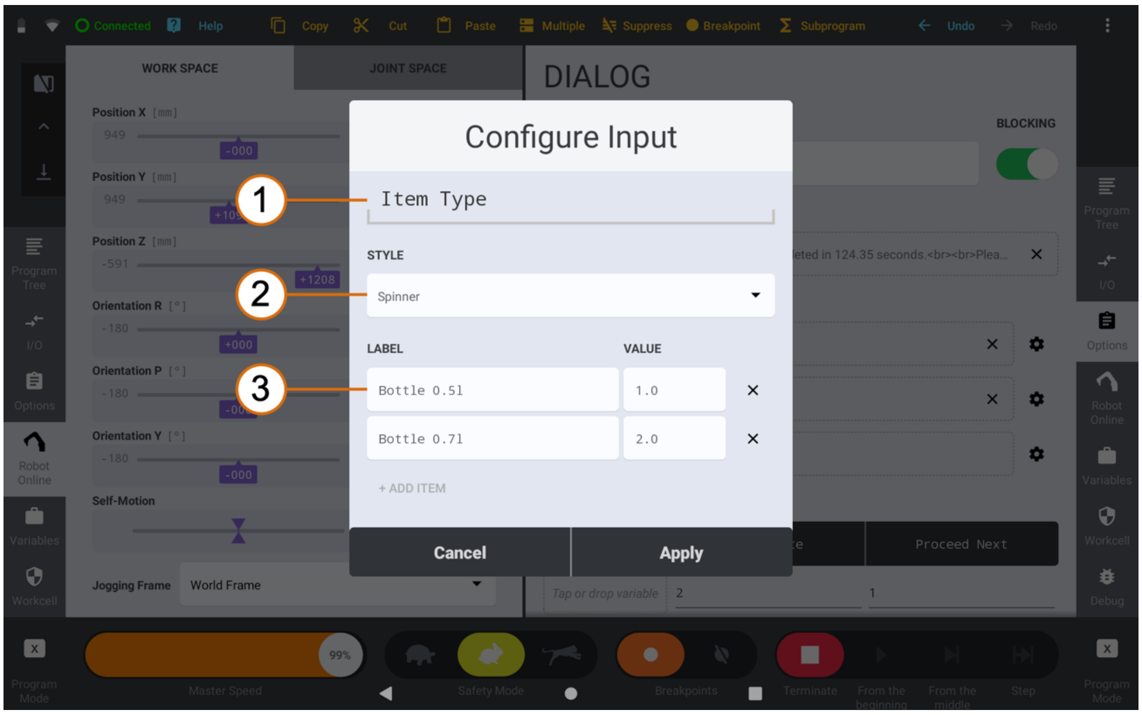


Item	Description
1	**Title** Allows the user to define the dialog title. The title should express the main purpose of the dialog.
2	**Blocking** Specifies whether the command is blocking. If blocking is enabled, the DIALOG command will be blocked unless the dialog is dismissed either by user or signal.
3	**Icon** Selects one of the following icons: None, Info, Warning and Error.
4	**Message** Clicking the message will allow the user to define the message via expression builder. The message allows concatenation of text and variables. In addition, the text also supports markdown language. Please see the example bellow: `“The batch has been completed in ‟ + duration + ‟ seconds.<br><br>Please enter the item **type** and the **count** of items in the next batch.”`
5	**Input Parameter** Defines up to 3 input parameters. The input parameter is defined by dragging target variable into the input parameter box (Number, Pose and Load data types are supported). This variable will be updated with the user defined value once the dialog is dismissed. The target variable also defines

	the default value of the input parameter view. Clicking the settings button opens the Configure Input dialog.
6	**Buttons Layout** Selects one, two or three buttons dialog layout.
7	**Button Labels** Allows the user to define custom button labels.
8	**Button Target** Specifies the target to be edited on button click event, ie. once the user clicks one of the buttons, the button value will be assigned to the target variable.
9	**Button Values** A value to be assigned to the target variable once the button is pressed.
10	**Dismiss Signal Edge** Determines whether the dialog should be dismissed on signal rising edge, falling edge or toggle of the signal.
11	**Dismiss Signal Expression** Specifies the signal to be used for dismissing of the dialog. If the dismiss signal is not defined, the dialog can be closed only by the user.

Configure Input Dialog

The Configure Input Dialog allows to specify the input label and style.



Item Type

STYLE: Spinner

LABEL	VALUE
Bottle 0.51	1.0
Bottle 0.71	2.0

Figure 2. Custom defined dialog window sample with the entry fields

Item	Description
1	**Label** Defines the input parameter label. The variable label is used by default.
2	**Style** Specifies the style of the input parameters. All supported data types provides the plain input method. In case of Number data type, also the switch, spinner and slider styles are supported.
3	**Style Configuration** Allows detailed configuration of the selected style. The style configuration options depend on the selected style. In case of spinner style, the configuration contains list of spinner items. The selected item value will be assigned to the target variable once the dialog was dismissed by the user.

5.3 Motion Control commands

Move command represents the basic building block for handling various KR robot motions. Because of the complexity of its options and integral role of this structure for the robot programming it is described within this dedicated chapter.

This group of commands provides a primary interface for all kinds of elementary motions the user can grasp and utilise for his/her desired application to achieve various robot trajectories. The recent version of the software supports several types of motion commands (***Trajectory Type***), which can be chosen from the main roll down menu of the MOVE options [Figure 3/1].

Type	Label	Description
Joint	**MOVE J**	Move to joint position (configuration), the trajectory is linear within the joint space.
Linear	**MOVE L**	Move to cartesian coordinates (pose), the trajectory is linear within the work space.
Spline	**MOVE S**	Move to cartesian coordinates (pose), the trajectory is rendered by a set of splines.
Arc	**MOVE A**	Move to cartesian coordinates (pose) copying the arc segment with a derived radius.

Based on the selected trajectory type, the MOVE options panel provide layout for the settings and user parameters separated into two common groups: ***Trajectory*** and ***Advanced*** [Figure 3/2].

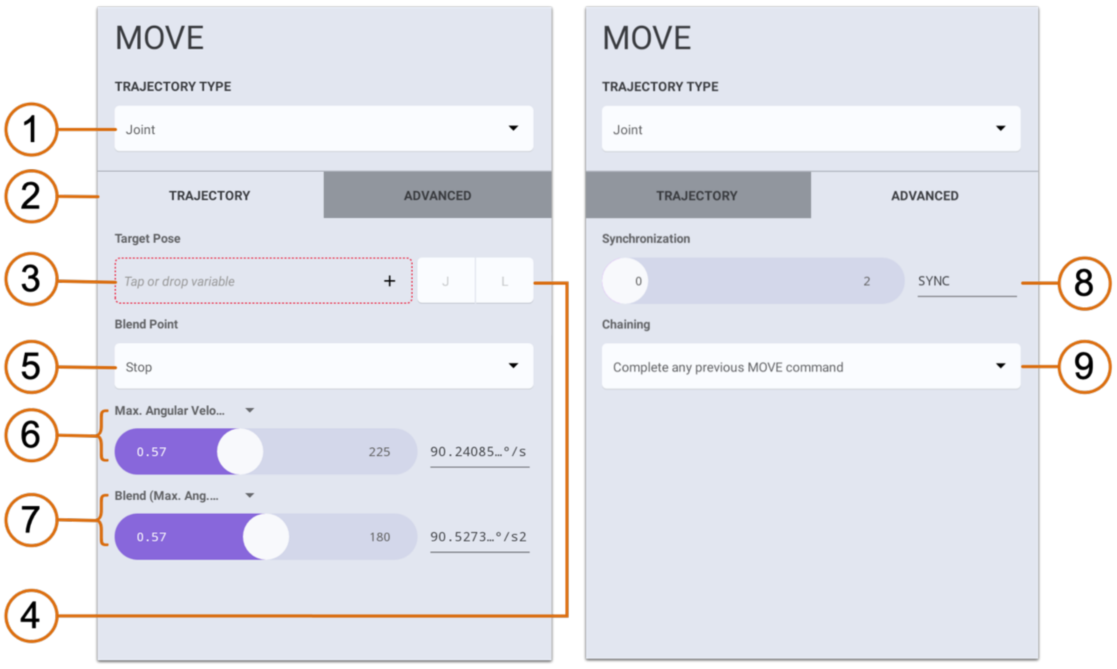


Figure 3. MOVE J Options layout

In general, the *Trajectory* tab encompasses a set of basic parameters necessary for the motion evaluation (target points or speed specification) while in *Advanced* the user can change the default behaviour of the command processing or specific geometrical features.

Target Pose

The common entry for all elementary motion commands is the ***Target Pose*** [Figure 3/3], which is also accompanied by the ***Instant Jogging Tool*** [Figure 3/4]. Using *J* or *L* buttons the user can immediately jog the robot to the pose entered within the *Target Pose* box. The *J* button uses the joint target position while the *L* uses the cartesian space and orientation for the jogging.

For a reminder please note, that each Pose variable contains both, the joint-position as well as the cartesian-space (or work space) coordinates. However, it is not verified by the system that those two sets are kinematically equivalent or even that those are valid when the values were modified/transformed programmatically or modified manually by the user.

5.3.1 MOVE J

The joint space move (MOVE J) is a basic, but very efficient kind of a motion command that simply drives the robot from the actual robot position into the target robot joint-configuration. The RC software provides the joint-space reference trajectory independently for each of the 7 robot joint axes. The shape and properties of the motion speed/timing profile are determined by various user input parameters.

Blend Point

Each elementary motion command can be set with the *Stop* or *Via value for the **Blend Point*** option [Figure 3/5] This option determines how the robot approaches the given target, i.e., if the robot stops exactly at the target point before continuing pursuing the next target, or it is allowed to blend with the consecutive motion requests.

Geometry and Speed profile

The reference trajectory of the joint move is provided independently for each robot joint axis. Its geometry is trivial, based on the linear profile connecting the starting (actual robot joint configuration) with the target joint position, assuming valid joint boundaries.

The speed profile has rectangular composition (acceleration, constant target speed, deceleration), determined by the speed and blending parameters. The timing of the trajectory is shared for all joints, i.e. all joints start and finish their motion at the same instant.

By default, the target speed is specified by the ***Max. Angular Velocity*** option [Figure 3/6] and its value [degrees° per sec]. Alternatively, the timing of the segment can be set explicitly, use the spinner and choose the "*Duration Time*" instead the "*Max. Angular Velocity*".

- *Maximum Angular Velocity* – target joint speed [degrees° per sec] (each joint will
- *Duration Time* – explicit time interval [s] is required to reach the target joint configuration

The way how the MOVE J blends when used in the context of consequent move commands is based on the same principle as in the case of MOVE L, except it is provided only in a single dimension. The resulting speed profile of such a blend is than given as a simple superposition of two consequent joint moves.

- *Blend (Time)* – the blending starts given time interval [s] before reaching the target robot configuration
- *Blend (Max. Angular Acceleration)* – a maximum joint acceleration (speed tilt) [deg/s^2] is allowed for all joints

(To get a better idea about the trajectory and blending composition, please check the next command description. MOVE L and MOVE J are quite similar at the level of their implementation, despite the fact they operate in different domains.)

Synchronization

Moves are processed sequentially and the consequent instruction or move command will be processed just at the moment, when the robot enters the blending phase of the actual move. This way, any decision about the next step or trajectory can be decided at the very last moment in the real-time.

Use the ***Synchronization*** [Figure 3/8] option located within the *Advanced* tab to change the interval to release the move instruction sooner, i.e., before reaching up the blending edge. The default value of this parameter is set to 0 meaning the program will continue to the next instruction right when the robot enters the respective blending area.

Immediate motion replacement

By default, each new executed move instruction is put at the end of the actual tracking plan. In some situations, it may be handy to cancel the present trajectory plan and replace it by the new one immediately. To allow that the ***Chaining*** option [Figure 3/9] must be set with the *"Interrupt any previous MOVE command"* option.

5.3.2 MOVE L

The linear move (MOVE L) represents the basic move command providing a coordinated robot motion to exercise the TCP along the line rendered in the cartesian-space coordinates, laid out by the initial (the actual TCP robot pose) and user defined *Target Pose*. The speed profile of this motion is determined primarily by the target speed, blend parameters and trajectory length.

Once the command is invoked, the RC draws up the linear reference trajectory representation which the robot follows by engaging all 7 joints. The orientation reference coordinates are evolving similarly, but within the context of rotational trajectories.

Blend Point

Analogously as in the case of the joint move, the MOVE L ***Blend Point*** *Stop* option [Figure 4/5] results in a trajectory that brings the robot TCP to the target pose before it continues to any further move operation.

When the MOVE L is set with the *Via* ***Blend Point*** option, the trajectory can be blended with the trajectory segment generated by the instruction which follows next in the program listing.

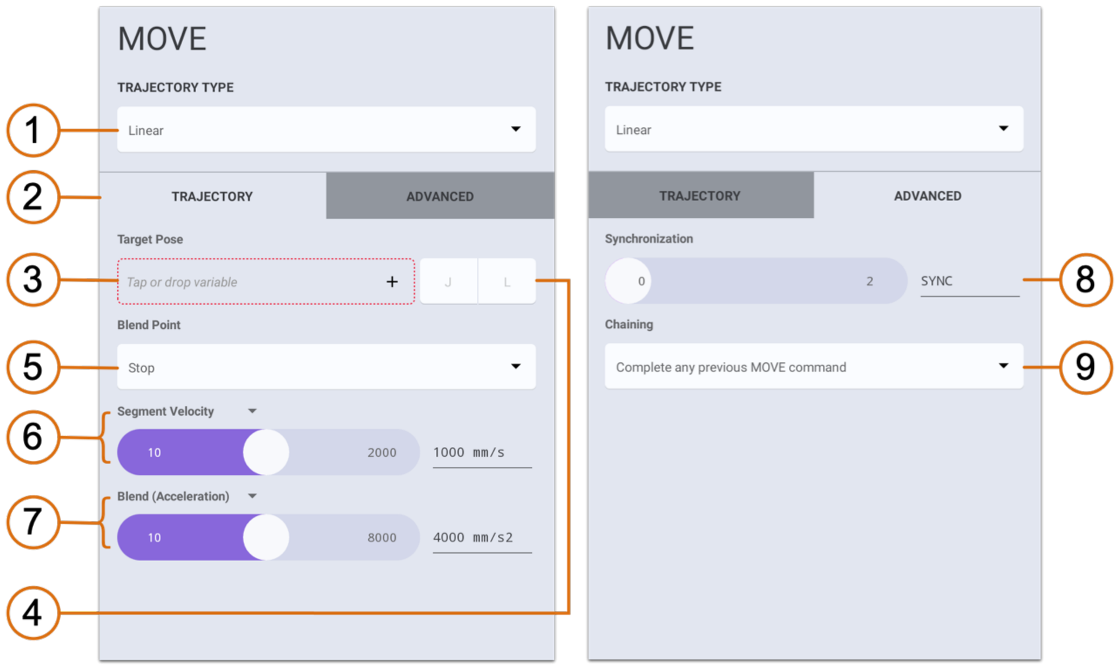


Figure 4. MOVE L Options layout

Geometry and Speed profile

The sequence of blending MOVE L commands is realized as alteration of blending and constant speed segments [Figure 5Figure 5]. The blending segment refers to the part of the trajectory, where the reference trajectory changes direction and speed to progress towards the next target pose. In other words, it is the part of the reference trajectory, where the TCP accelerates or decelerates.

The constant segment speed is specified by the ***Segment Velocity*** option [Figure 4/6] and its value. This is the target translational speed the TCP is requested to reach on its way towards the target pose. Alternatively, the timing of the segment can be set explicitly, use the spinner and choose the “Duration Time” instead the “Segment Velocity”.

- *Segment Velocity* – target speed [mm/s] to be reached on the linear segment
- *Duration Time* – explicit time interval [s] is required to reach the target pose

To determine the resulting geometry and the speed profile of a linear move, proportions of the blend segment are also required. There are 3 ways how to define the blend segment size (use the spinner to change the default “*Blend (Acceleration)*” option):

- *Blend (Distance)* – the blending starts at a specified distance [mm] from the target pose
- *Blend (Time)* – the blending starts given time interval [s] before reaching the target pose
- *Blend (Acceleration)* – only a maximum acceleration [mm per s^2] is allowed during the blending

When a sequence of linear moves is put together (by using the *Via* Blend Point option), the consequent moves provide the geometrical and speed blending around each target point.

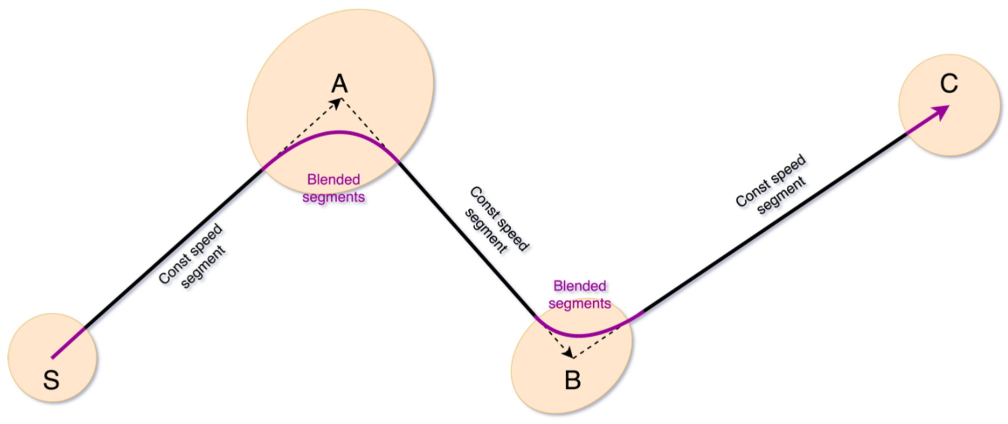


Figure 5. Blend of three consequent MOVE L commands (using via points)

Synchronization

Moves are processed sequentially and the consequent instruction or move command will be processed just at the moment when the robot enters the blending edge (sphere) of the actual move. This way the decision about the next step or trajectory can be decided at the very last moment in the real-time.

To adjust this synchronisation, you can use the ***Synchronization*** [Figure 4/8] option located within the Advanced tab to change the interval to unlock the present move sooner, before reaching up the blending edge. The default value of this parameter is 0, so it will continue at the next instruction right on entering the blend.

Immediate motion replacement

Same way as in the case of the joint move, each new executed move instruction is put at the end of the actual tracking plan. In some situations, it may be handy to cancel the present trajectory plan and replace it by the new one immediately. To allow that the ***Chaining*** option [Figure 4/9] must be set with the *“Interrupt any previous MOVE command”* option.

5.3.3 MOVE S

A different kind of the TCP trajectory tracking command, the Spline Move (MOVE S), is based on higher order curves representation. Unlike the MOVE L, the spline trajectory always passes through the defined target point (knot point) while keeping the input translational speed. The motion can be configured to use *tangential orientation* option, which provides the orientation exactly within the direction of the TCP motion.

This set of features is handy for various advanced applications like welding, dispensing or cutting. The combination of those features makes the MOVE S a better choice also for the imported curves processing (e.g., by using the Step File CBun).

Knot Pose

It is more common to refer to *knot points* when a geometry is described by a set of polynomial curves, i.e., when the trajectory is being processed as a spline. In our spline moves we do that to emphasize a bit more specialised role of the entry points used for this kind of trajectory parametrisation. By a ***Knot Pose*** we label the respective target knot point also accompanied by the orientation coordinates. This pose is set by the user within the MOVE S interface [Figure 6/3].

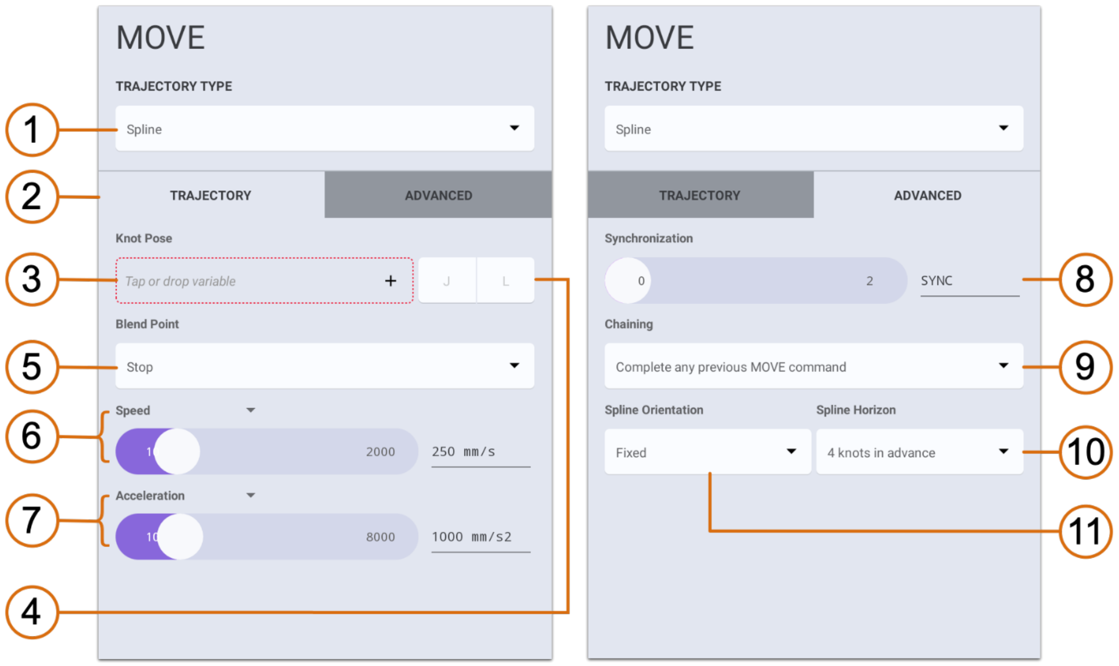


Figure 6. MOVE S Options layout

Geometry

The trajectory composed as a sequence of MOVE S commands is determined by their knot poses with the result curve optimised for low variations and geometrical tensions. In other words, the result shape should be kink, noose or slings free if that was possible for a given combination of arguments.

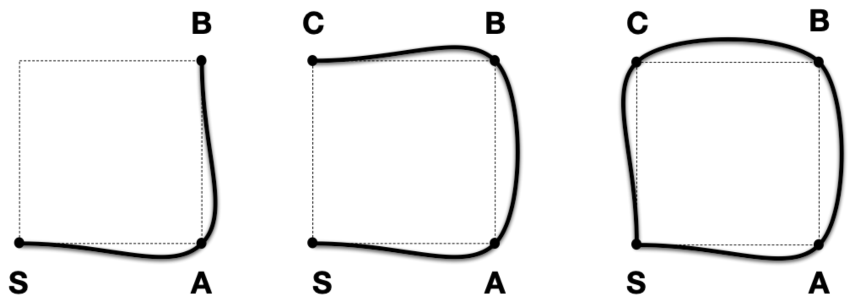


Figure 7. Composition of successive spline segments

The basic concept of the spline composition is illustrated at the picture above [Figure 7]. Consequent spline segments are glued together in a way to guarantee the smoothness and knot points path inclusion while the first and last spline of this composition are treated specifically. Because the direction of the spline can't be determined by a consequent or previous segment at terminal knot points, it is inferred from the first/last couple. Taking the sample illustration from above, the [S, A] and [C, S] segments are determining the direction for their splines.

Please note the spline move is set to ***STOP*** blend point [Figure 6/5] by default. To allow the blending (gluing) between the successive commands the ***VIA*** option needs to be set.

Segment Speed

Aside of geometrical differences, the speed control of the spline move can be beneficial in various applications (as welding, dispensing, gluing) for which the linear motion won't guarantee absolute constant speeds (blended segments).

There are two main parameters determining the speed profile of the sequence of spline moves: the ***Speed*** [Figure 6/6] and the ***Acceleration*** [Figure 6/7]. The first one defines the level of the target absolute translational speed, while the latter specify the max. slope that can be used to achieve that within the corresponding segment.

For example, a succession of MOVE S with the different speeds and accelerations per segment,

```
/* assume the robot is located at the S pose in the beginning) */
MOVE_S  A   [speed: 200mm/s, acceleration: 500mm/s^2]
MOVE_S  B   [speed: 300mm/s, acceleration: 200mm/s^2]
MOVE_S  C   [speed: 100mm/s, acceleration: 600mm/s^2]
MOVE_S  S   [speed: --, acceleration: 150mm/s^2]
```

will result in the following kind of speed profile, where the speed is adjusted for each segment respectively [Figure 8].

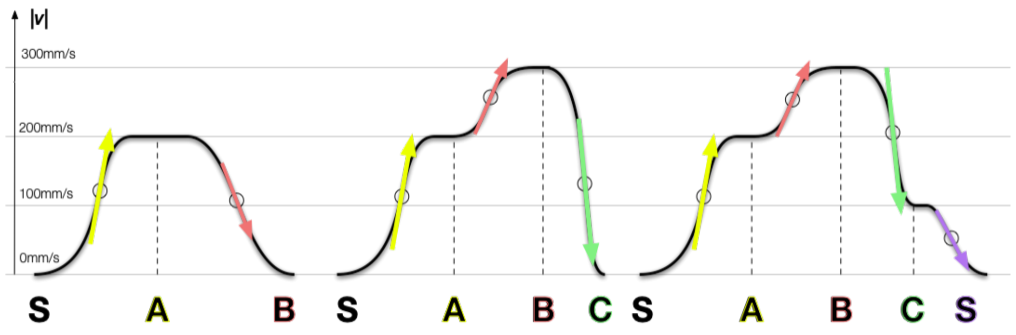


Figure 8. Speed profile of successive spline segments

Tangential orientation

The ***Spline Orientation*** [Figure 6/11] is set as *Fixed* for the MOVE S command by default. However, with the present implementation of the spline moves provide also support for the *tangential orientation*, useful especially for applications where it is required the TCP keeps its direction along the spline trajectory (e.g. welding).

If the spline move command is set as tangential, the main spline is accompanied by a secondary curve, which is used as the orientation handle for TCP orientation. Since the resulting TCP orientation is treated in a way to follow the tangential direction of the main trajectory, the entry pose orientation is used only to rule out the single remaining degree of freedom (free or perpendicular axis), which is located around the tangential axis.

The user can choose from two options, which axis of the entry knot pose to use to determine the perpendicular axis of the tangential orientation, the X – *Tangent X* or Z – *Tangent Z.* This choice is relevant mainly regarding the way the entry poses are being created or teached.

A rule of thumb is that for manually teached entry poses, it is recommended to choose *Tangent Z* option, while for the sampled or generated poses (e.g. from the Step File CBun) it is more adequate to use the other, *Tangent X*.

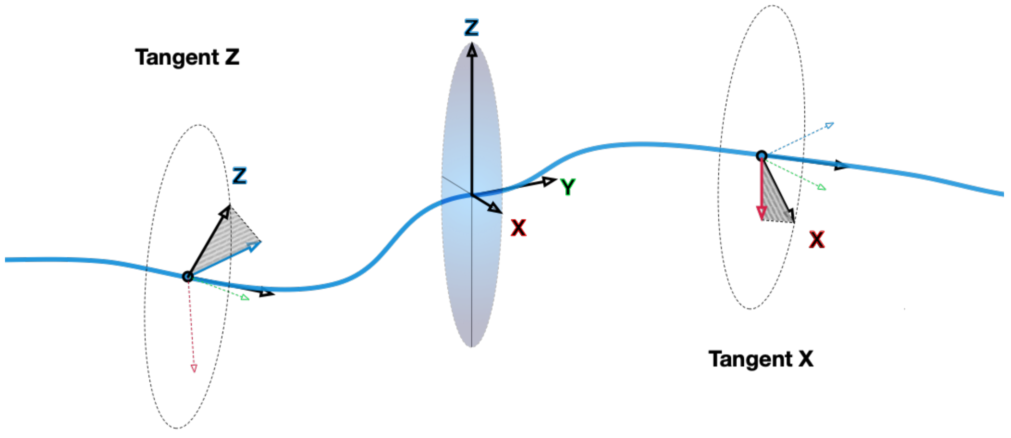


Figure 9. Tangential motion axis handling schema

Real-time processing

As in case of other elementary move commands, the spline move is also implemented in a way which allows to provide the instruction at the very last moment of the execution. This allows to build the trajectory on the fly i.e., while the robot is moving along it, just a segment ahead. Based on this feature the robot program can do decisions about the next target and amend the plan in the real-time.

Moreover, for the optimal curve shape it is crucial to keep several knot-points ahead. The size of this ***Spline Horizon*** can be modified in ***Advanced*** tab of the MOVE S command options [Figure 6/10]. The default value is using 4 knot-points. The size of this buffer affects the smoothness and shape of the resulting geometry. Basically, less points in the horizon results in more ragged curve shape and accelerations, but it requires less calculation power to update the trajectory in one step of the RC.

From the program perspective, the *Spline Horizon* also affects the way how the MOVE S instructions are executed and how the IP (instruction pointer) is updated while the spline move is being processed by the RC. For a particular MOVE S the command doesn't block the execution and continues right to the next instruction if the spline handler doesn't have the required number in its internal buffer yet.

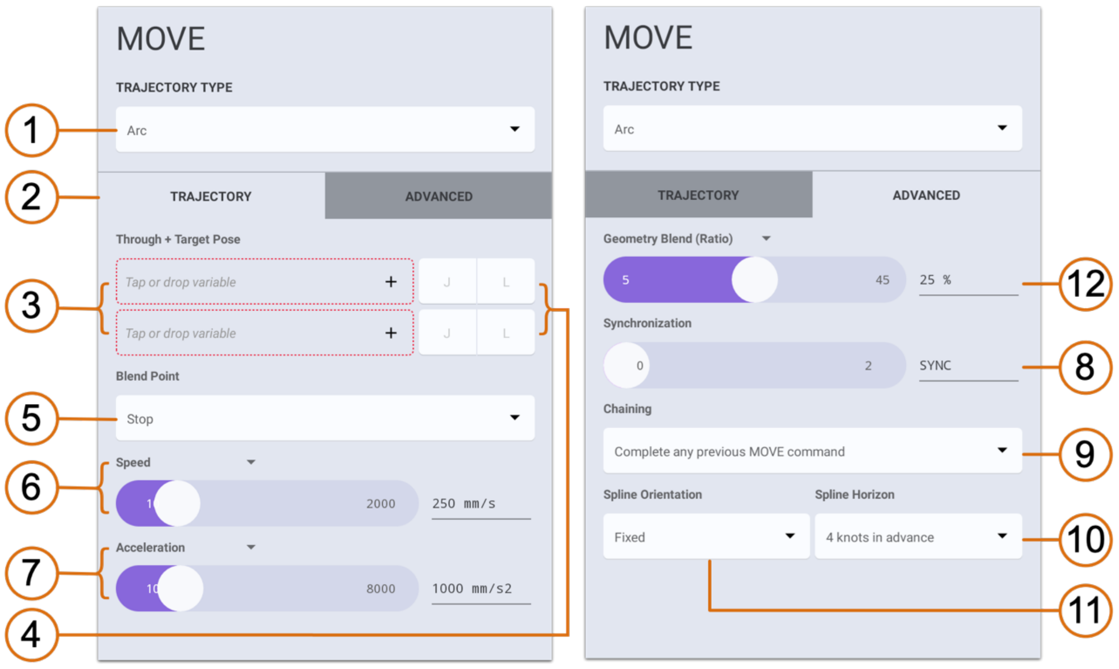
The ***Synchronization*** option [Figure 6/8] is the other way how to modify the MOVE S instruction processing. The default value used for this slider field is set to 0, which means the command will wait up to a sync event, after which the instruction is “released” and the IP will move and execute the next instruction in the program. In majority of cases, the spline move sync event is simply a moment when the TCP passes through the knot-pose defined by the MOVE S command. The ***Synchronization*** value can be than used to adjust the time of the sync event and schedule it before the default timing using the nonzero time interval.

To eliminate any MOVE S command synchronisation, it is also possible to set the ***Synchronization*** *as ASYNC* (simply by dragging the slider to the rightmost position) and let the RC feed the spline move requests into its internal buffer. Although this option is not recommended because of possible negative effects on the RC performance, it can be handy in some scenarios where the set volume of entry points is well known or under control by the user.

Similarly, as in other kinds of move commands, also spline move allows completely replace/reset the actual trajectory plan by a fresh new MOVE S command. By using the ***Chaining*** option “*Interrupt any previous MOVE command*” [Figure 6/9] the RC will be commanded to cancel any previous move commands and blend the actual dynamical state to move towards a new target point immediately.

5.3.4 MOVE A

The arc move (MOVE A) is a specialised TCP tracking command providing the circular motion which can be helpful in a set of applications requiring regular and precise shapes and rounded edges.



MOVE

TRAJECTORY TYPE: Arc

Through + Target Pose

Blend Point: Stop

Speed: 1 – 2000, 250 mm/s

Acceleration: 10 – 8000, 1000 mm/s2

Geometry Blend (Ratio): 5 – 45, 25 %

Synchronization: 0 – 2, SYNC

Chaining: Complete any previous MOVE command

Spline Orientation: Fixed

Spline Horizon: 4 knots in advance

Figure 10. MOVE A Options layout

The recent MOVE A command interface allows to define the arc (portion of circle) by two entry poses ***Through + Target*** [Figure 10/3]. The first one is used to determine the arc radius while the latter provides the information about the end of arc (it is supposed to be the target point as in case of any other elementary moves).

When the MOVE A is executed in the context of the program, the shape and the placement of the arc trajectory is determined from the actual robot pose (TCP) and two MOVE A entry poses. The rest of the command parameters help to adjust the arc trajectory blending, speed or instruction synchronisation to fit users' needs.

The s*peed profile* of the arc move is composed in analogous way as it is provided for the spline moves [Page 74], assuming the ***Speed*** and ***Acceleration*** [Figure 10/6,7] is specified by the user.

It also applies for the *real-time* and *synchronisation* features, which are based on the same principles.

Blending

With the ***Blend Point*** [Figure 10/4] it is possible to set the arc move segment as *Via.* Similarly, as for other move commands, this option basically allows to blend with any segment processed following the actual MOVE A.

The arc segment is blended into a continuing trajectory by the use of spline curves at both ends (if needed). The length of this blending sub-segment can be changed by the user, and it is expressed as a portion of the arc (in %), ***Geometry Blend (Ratio)*** [Figure 10/12]. The following picture briefly illustrates the basic concepts and arc move composition on 3 consequent arcs interconnected by line moves.

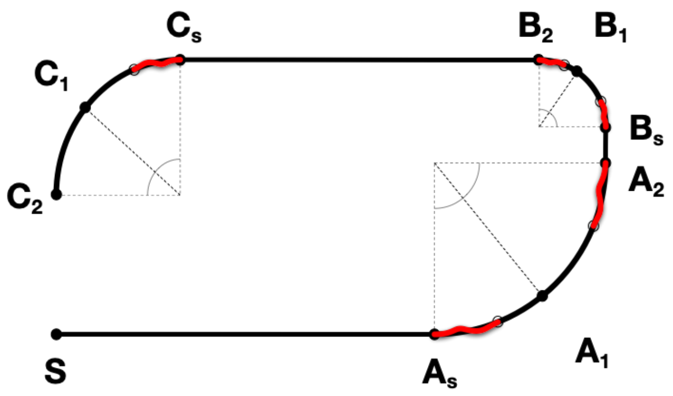


Figure 11. Arc move composition and their blend sub-segments (red)

Please note the default *Stop* value of the ***Blend Point*** option doesn't provide blending with a subsequent motion command and the robot stops tracking the TCP at the target pose before continuing with any further motion.

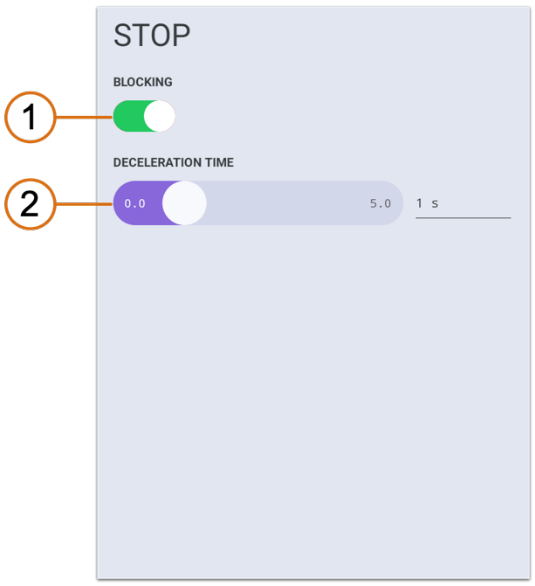
Tangential orientation

The arc move orientation tracking is providing tools based on same principles as in case of the spline move that was described few pages above [Page 74]. Also here, the entry target pose orientation is used to determine the perpendicular axis while the tangential direction is based on the trajectory shape.

5.3.5 STOP

The STOP command suspends the robot motion until the RESUME command is used to cancel this state. The instruction can be called asynchronously from any running sequence.

Command Options



By default, the command is set as ***Blocking*** [1]. In this case the instruction processing is blocked until the robot motion is stopped. To use the STOP asynchronously, slide it to the left/off/red position. The rate of deceleration can be adjusted by the ***Deceleration Time*** [2] value.

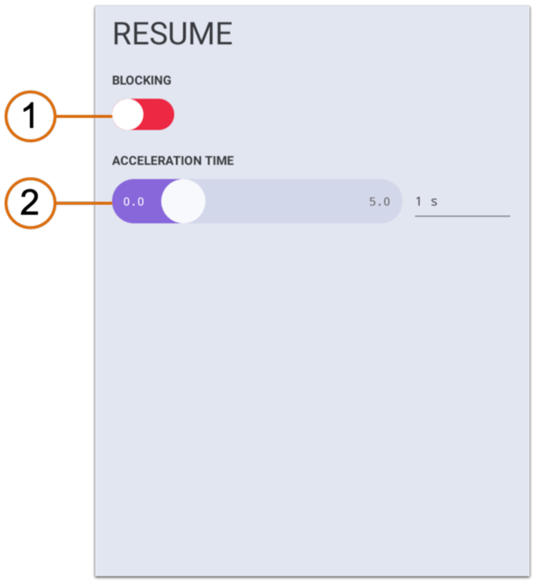
5.3.6 RESUME

The RESUME command cancels the motion suspended state, which was previously invoked by the STOP command. The instruction can be called asynchronously from any running sequence.

The command is set as ***non-Blocking*** [1] by default. To make the command “wait” until the robot cancels the suspended state and reaches the full speed completely set it to on/right/green position.

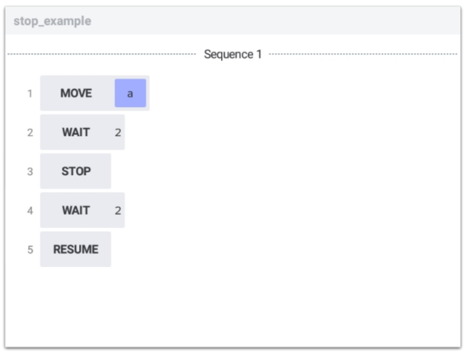
The rate of resuming acceleration can be adjusted by the ***Acceleration Time*** [2] slider/value. This will be a time used to catch-up with a previously suspended robot motion.

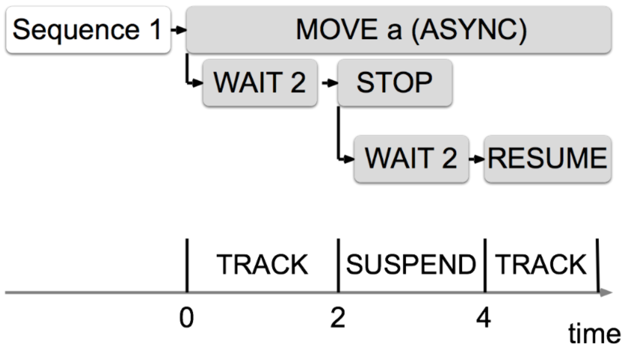
Command Options



Example

The following example illustrates the usage of STOP and RESUME command after asynchronous MOVE command. For the first 2 seconds the robot is tracking. Then the STOP command is called, and robot is suspended for next 2 seconds until RESUME command is called.





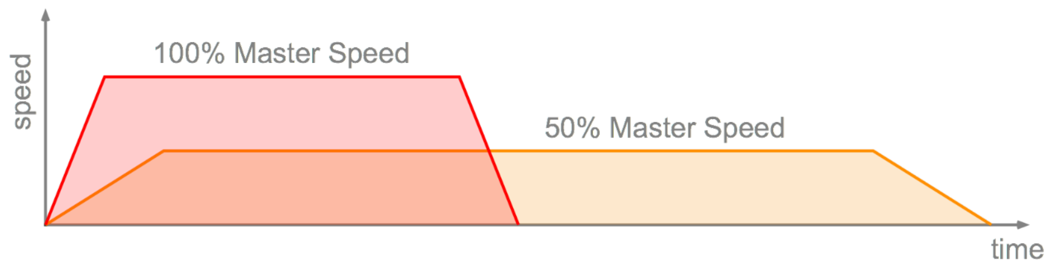
6 Program Control

The Program Control chapter describes how users can run and debug program. At the beginning the execution params are introduced such as master speed and safety mode. Later the program launch is explained as well as program termination and program debugging. All Program Control items can be found in the Bottom Bar (Program Mode) (see 3.1.5).

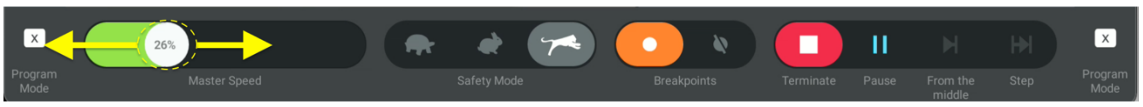
6.1 Execution Params

6.1.1 Master Speed

The *Master Speed slider* allows the user to scale down the motion speed and acceleration during the program execution. The Master speed can be set between 1% and 100%, where 100% means standard motion speed and 1% is the lowest, but the safest speed. Note that the decreasing of Master Speed leads to longer program execution time.



With the recent versions of the RC software, it is possible to change the master speed “on the fly” i.e., also while to robot is in automatic mode providing the user program or in the middle of the motion. For the safety reasons the user needs to **double-tap** the *Master Speed slider* to unlock it to be able to move it.



6.1.2 Safety Mode

The Safety Mode switch determines whether the robot should move fast or provide higher safety. There are 3 states available: Safe, Reduced, Normal. While the Normal state does not limit the speed, Safe and Reduced state limits the maximum speed of any part of the robot. That means, that any part of the robot will move faster than the applied speed limit.

Default speed limits are:

- Safe (Turtle) – 0.25 m/s
- Reduced (Rabbit) – 1 m/s
- Normal (Cheetah) – Unlimited

6.2 Program Launch

Kassow Robots control software allows users to launch their programs in just few clicks.

- Please stand clear (all persons incl. the programmer/operator) of the robot before launching a program.
- Run a new program at low speed to avoid hard collisions with surrounding equipment due to programming errors.

To launch your program:

1. Check program validity (all components have to be valid)
2. Open Program Mode ▶
3. Adjust execution params (see 6.1)
4. Click **From the begging** ▶

6.2.1 Move to Start Pose

In order to avoid any unexpected movement at the beginning of program execution, the first MOVE command pauses program and user interaction is requested. First MOVE command is always provided in Joint Space and with limited parameters.

To move robot to start pose:

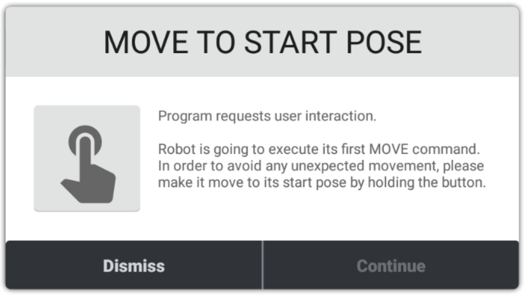
1. Press and hold
2. Wait until the start pose is reached (teach pendant vibrates)
3. Click **Continue** to resume the program execution

6.3 Program Termination

Users can invoke immediate program termination in any moment. Program termination interrupts also any ongoing robot movement.

To terminate program execution:

1. Open Program Mode
2. Click **Terminate**



MOVE TO START POSE

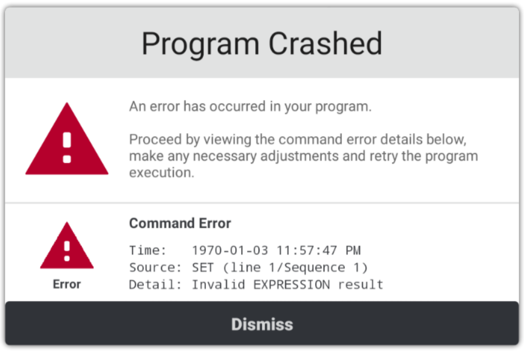
Program requests user interaction.

Robot is going to execute its first MOVE command. In order to avoid any unexpected movement, please make it move to its start pose by holding the button.

6.4 Program Failure

Program failure occurs if the program stops functioning properly and terminates itself. Most program failures are result of command or CBun error. Typical causes include invalid program data, unexpected command operation, robot failure or HW issues.

Each program failure is reported to user with all details relating to it.



Program Crashed

An error has occurred in your program.

Proceed by viewing the command error details below, make any necessary adjustments and retry the program execution.

Command Error

```
Time:   1970-01-03 11:57:47 PM
Source: SET (line 1/Sequence 1)
Detail: Invalid EXPRESSION result
```

6.5 Alarm Handling

Alarm is response of the system to the occurrence of exceptions (anomalous conditions requiring special handling) during the program execution. Alarm is typically triggered on command exception, CBun exception, P-STOP, E-STOP or violation of safety functions.

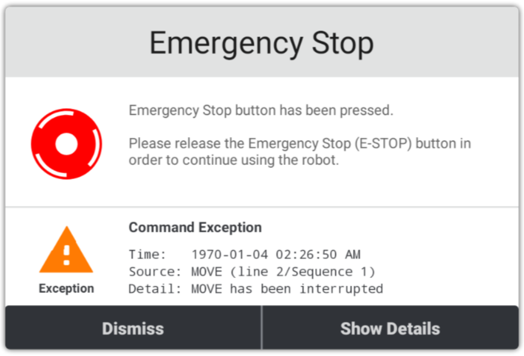
Alarm always causes the program to halt until the exceptional state is properly resolved either by user (default handling) or by one or more dedicated program sequences (custom handling).

6.5.1 Default Handling

Default alarm handling lets the user to resolve the exceptional state manually and decide whether the program execution can continue or should be terminated. Default alarm handling is available only if there is not any custom handling used.

Exception Resolution

Most of the exceptional states must be resolved by user, otherwise the program cannot be recovered. For example, if E-STOP button was pressed during the program execution, the button must be released and the robot has to be re-initialized in order to resolve the exceptional state.



Emergency Stop

Emergency Stop button has been pressed.

Please release the Emergency Stop (E-STOP) button in order to continue using the robot.

Command Exception

```
Time:    1970-01-04 02:26:50 AM
Source:  MOVE (line 2/Sequence 1)
Detail:  MOVE has been interrupted
```

Execution Control

Once the exception is resolved, the execution control can be provided. The user can proceed by:

1. **Program Recovery and Continue**

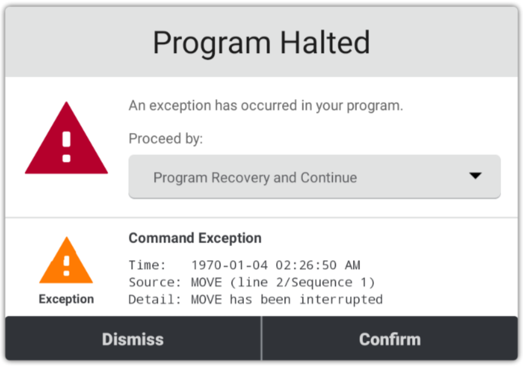
Repeats command that caused the alarm (i.e.. thrown an exception) and continues program execution.

2. **Ignore and Continue**

Ignores command that caused the alarm (ie. thrown an exception) and continues program execution.

3. **Program Termination**

Terminates program execution immediately.


Program Halted

An exception has occurred in your program.

Proceed by:

Program Recovery and Continue

Command Exception

```
Time:   1970-01-04 02:26:50 AM
Source: MOVE (line 2/Sequence 1)
Detail: MOVE has been interrupted
```

6.5.2 Custom Handling

Kassow Robots control software allows users to define one or more sequences that are executed on alarm event (see 5.2.1). If program contains at least one such sequence, the default alarm handling is not used, and we are talking about custom alarm handling.

All alarm handling sequences start simultaneously when the alarm event occurs. At the same time, all other sequences are halted until the program halt state is cleared. Beware that if any exception is thrown in the alarm handling sequence, this sequence becomes halted too and the only way to resolve this situation is the program termination.

If another alarm event occurs during the ongoing custom handling, each alarm handling sequence runs again as soon as it completes the previous execution.

6.6 Debugging

Debugging is the process of finding and resolving unexpected behaviour, exceptions and errors within the program. Kassow Robots control software provides several tools for proper program debugging.

6.6.1 Pause

To pause program execution either:

1. Open Program Mode

2. Click **Pause**

or:

1. Click the Toggle button

Note: Robot motion is also suspended on program pause.

6.6.2 Continue

To continue program execution either:

1. Open Program Mode ▶
2. Click **Continue** ▶

or:

1. Click the Toggle button

Note: Robot motion is also unsuspended on program continue.

6.6.3 Step

Once the program is paused (either via pause button, toggle button or breakpoint), the user can control step-by-step the program execution. During this process, all MOVE commands become interactive, ie. the robot moves as long as the user holds the Step button.

In case of multiple sequences, single command in one of the sequences is called on each step button click. The execution order is not guaranteed.

To execute single command:

1. Open Program Mode ▶
2. Press and hold **Step** ⏭
3. Wait until the command is completed (teach pendant vibrates)

6.6.4 Breakpoints

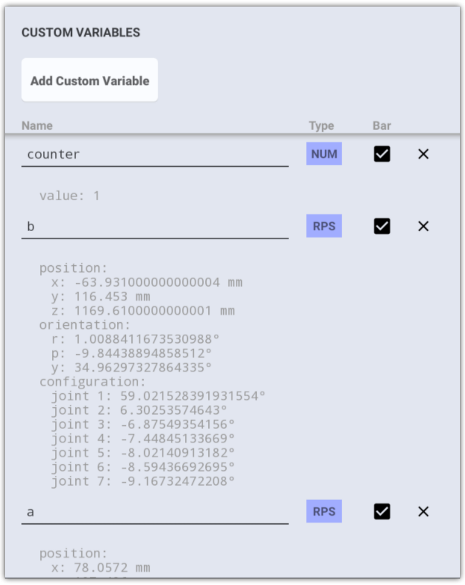
Breakpoint is an execution stop point in a program. When breakpoint is reached, it pauses the program execution and lets the user to debug the program. Note that a breakpoint can be set on any program command and even on multiple commands.

To set a breakpoint:

4. Select a program command line (with no breakpoint)
5. Click **Breakpoint** ●

To remove a breakpoint:

1. Select a program command line (with breakpoint)



Name	Type	Bar
counter	NUM	✓
b	RPS	✓
a	RPS	✓

```
value: 1
```

```
position:
  x: -63.931000000000004 mm
  y: 116.453 mm
  z: 1169.6100000000001 mm
orientation:
  r: 1.0088411673530988°
  p: -9.84438894858512°
  y: 34.96297327864335°
configuration:
  joint 1: 59.021528391931554°
  joint 2: 6.30253574643°
  joint 3: -6.87549354156°
  joint 4: -7.44845133669°
  joint 5: -8.02140913182°
  joint 6: -8.59436692695°
  joint 7: -9.16732472208°
```

```
position:
  x: 78.0572 mm
```

2. Click **Breakpoint** ●

To enable breakpoints:

1. Open Program Mode ▶
2. Enable **Breakpoints**

To disable breakpoints:

1. Open Program Mode ▶
2. Disable **Breakpoints**

6.6.5 Runtime Values

Once the program is paused (either via pause button, toggle button or breakpoint), the user can inspect runtime value of custom variables (Local and Global Scope). These runtime values can be accessed via Variables Tool (see 3.6).

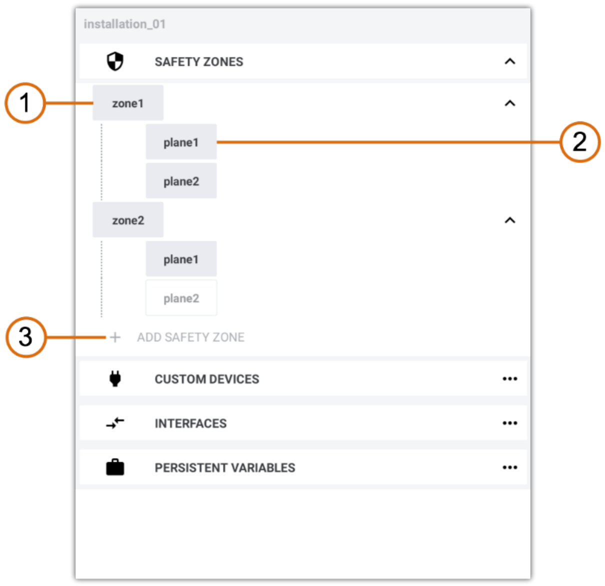
In case of debugging multiple sequences, runtime values are available only for those variables that are involved in sequence that is going to be executed next, ie. contains highlighted command.

7 Workcell

Anytime the robot is used for some specific application, the robot installation setup should be adjusted. This may involve configuration of safety zones, activation and mounting of 3rd party accessories (grippers, cameras, etc.), interfaces setup or persistent variables declaration. All these settings are grouped in so-called Workcell.

7.1 Safety Zones

Safety Zones represent virtual boundaries in the robot workspace. The robot will reduce its speed or stop completely once collides with any safety zone. This can be used for protection of sensitive equipment or areas with human presence.

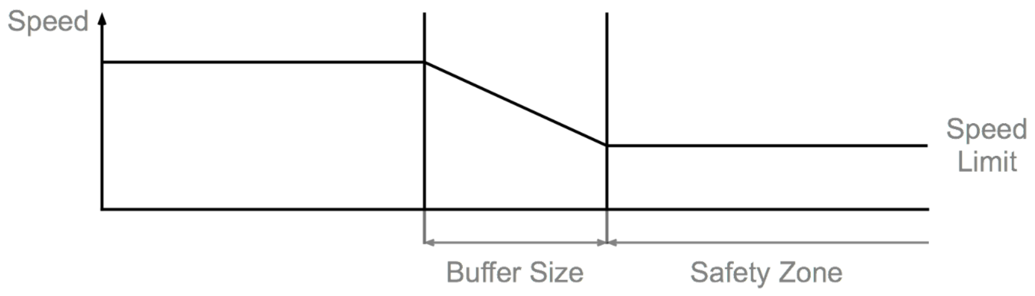


Item	Description
1	**Safety Zone** Represents single virtual boundary as an intersection of its geometries.
2	**Geometry** Acts as a basic building element of a safety zone.
3	**Add Safety Zone** Adds a new safety zone at the end of the list.

7.1.1 Safety Zone

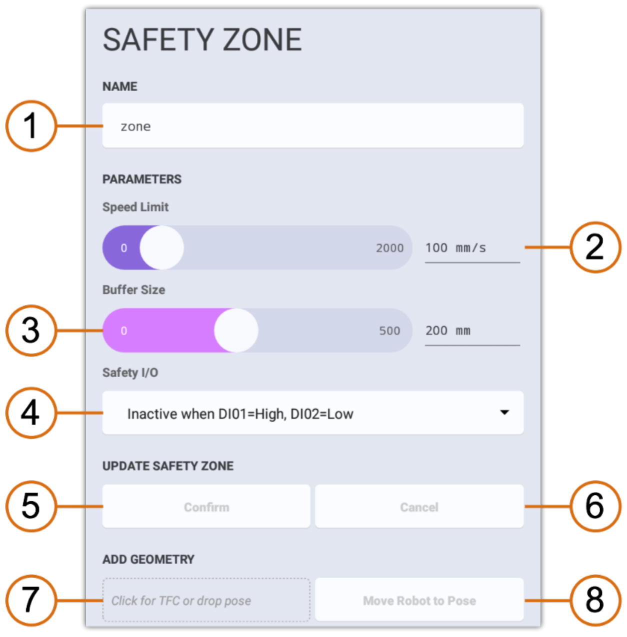
Safety Zone may be defined without geometry (surrounds whole workspace of the robot), with single geometry or as an intersection of multiple geometries. By default, the safety zone is always active, but it can also be activated/deactivated dynamically using Safety I/O.

The robot will reduce its speed while it gets closer to the zone, and it will keep the movement below the speed limit or stop completely inside the safety zone. The deceleration rate depends on the buffer size parameter.



Options Panel

The safety zone has the following Options panel.



SAFETY ZONE

NAME

zone

PARAMETERS

Speed Limit: 0 – 2000, 100 mm/s

Buffer Size: 0 – 500, 200 mm

Safety I/O: Inactive when DI01=High, DI02=Low

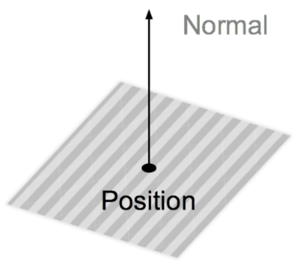
UPDATE SAFETY ZONE

ADD GEOMETRY

Item	Description
1	**Safety Zone Name** Displays the user defined safety zone name.
2	**Speed Limit** Speed limit represents the maximum speed that any part of the robot can move within the safety zone. The speed limit can be set between 0 mm/s (ie. the robot stops when it reaches the safety zone border) to 2000 mm/s.
3	**Buffer Size** The robot starts slowing down linearly when the distance from safety zone is lees then the buffer size.
4	**Safety I/O** Allows the user to select whether the safety zone should be always active (static safety zone) or bound to some I/O.
5	**Confirm Update** Provides update of the safety zone, i.e. applies the local changes.
6	**Cancel Update** Discards the local changes, i.e.. displays the active safety zone configuration.
7	**Geometry Pose** Allows the user to add a new geometry into the safety zone. New geometry pose can be injected by drag drop (Pose) or by clicking into the field (current TFC).
8	**Confirm Geometry** Once the geometry pose is injected, pressing the button makes the robot move toward the pose (in joint space). When the robot reaches the pose, new geometry (Plane) can be added by clicking the button again.

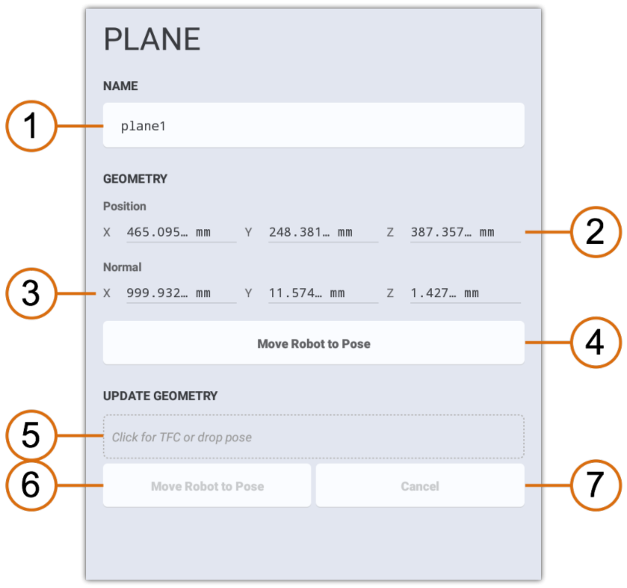
7.1.2 Geometry (Plane)

Geometry is a flat, two-dimensional surface that divides robot workspace into two subspaces. The plane geometry is defined in a point-normal form, i.e. using a point in the plane and a vector orthogonal to it (the normal vector).



Options Panel

The geometry has the following Options panel.

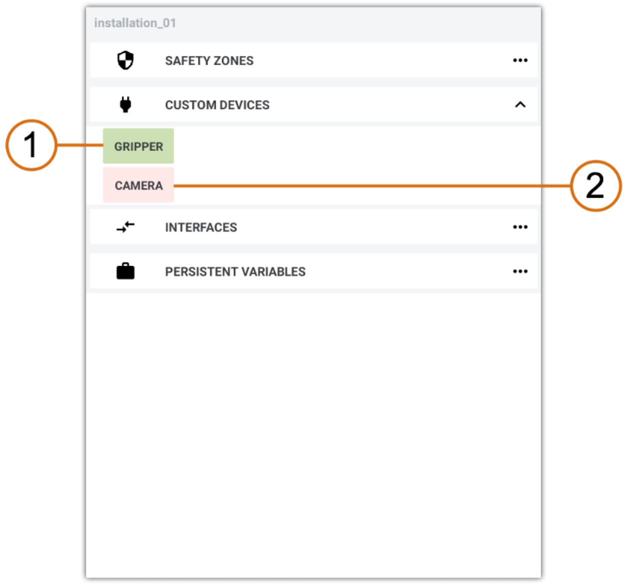


Item	Description
1	**Geometry Name** Displays the user defined geometry name.
2	**Position** Position (X, Y, Z) represents a point in plane.
3	**Normal** Normal (X, Y, Z) represents a direction of plane normal.
4	**Move Robot to Pose** Pressing the button makes the robot move toward the specified pose (in Joint Space).
5	**Geometry Pose** Allows the user to update the plane geometry. New pose can be injected by drag drop (Pose) or by clicking the field (current TFC).
6	**Confirm Update** Once the geometry is updated, pressing the button makes the robot move toward the pose (in Joint Space). When the robot reaches the pose, geometry can be updated by clicking the button again.
7	**Cancel Update** Discards local changes, i.e. displays the active geometry configuration.

7.2 Custom Devices

The concept of custom devices provides plug & play interface for 3rd party HW accessory (such as gripper or camera). Custom device can be added via previously installed CBun. All added devices are listed in Custom Devices section.

All custom devices can be drag dropped into a Program Tree. But the program tree execution can be provided only if all involved devices are successfully activated.



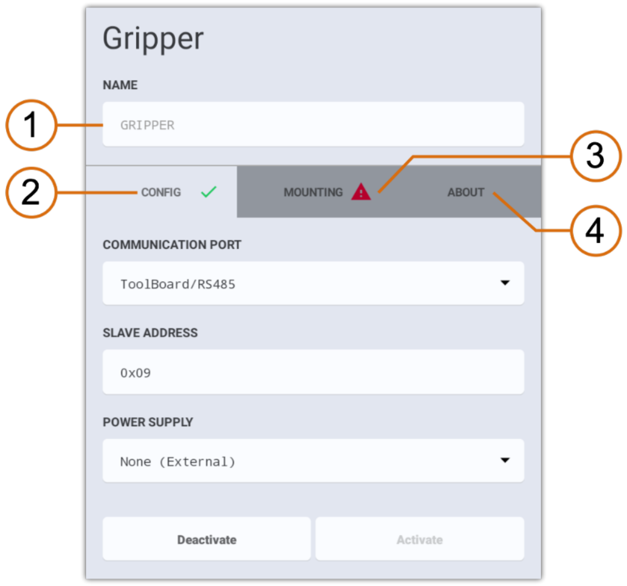
Item	Description
1	**Device (Active)** Represents installed and active custom device.
2	**Device (Inactive)** Represents installed but inactive custom device. Beware that the program can be executed only if all involved devices are active.

7.2.1 Custom Device

Custom device allows the user to activate/deactivate and mount/unmount installed 3rd party devices.

Options Panel

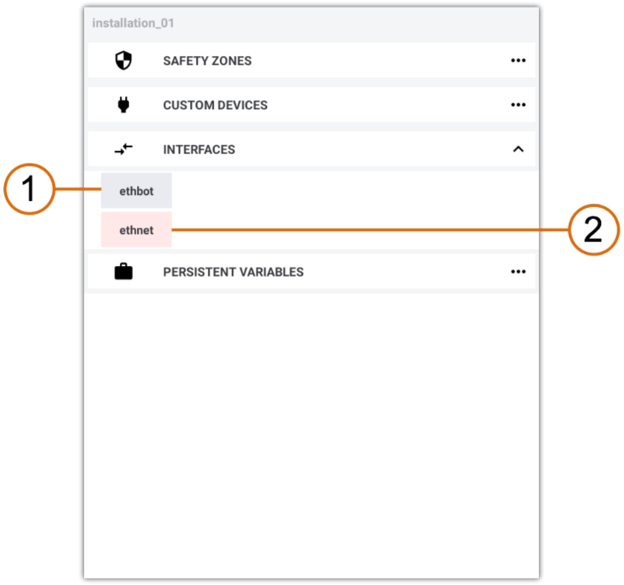
The custom device has the following Options panel.



Item	Description
1	**Device Name** Displays the device name. This name is also used for device identification in Program Tree.
2	**Config Tab** Provides tools for activation/deactivation of the device.
3	**Mounting Tab** (Optional) Allows the user to mount the device to the robot, i.e. adjust the kinematics model.
4	**About Tab** Displays the CBun name, author, version and custom device description.

7.3 Interfaces

Interfaces allows the user to configure built-in network interfaces.



Item	Description
1	**Interface (Valid)** Represents interface with valid configuration.
2	**Interface (Invalid)** Represents invalid interface, i.e. without local changes being applied.

ETHNET

“Ethnet” interface is designed to connect the cabinet to a local network. This can be used for internet access (necessary for remote support) or for communication with other devices/equipment.

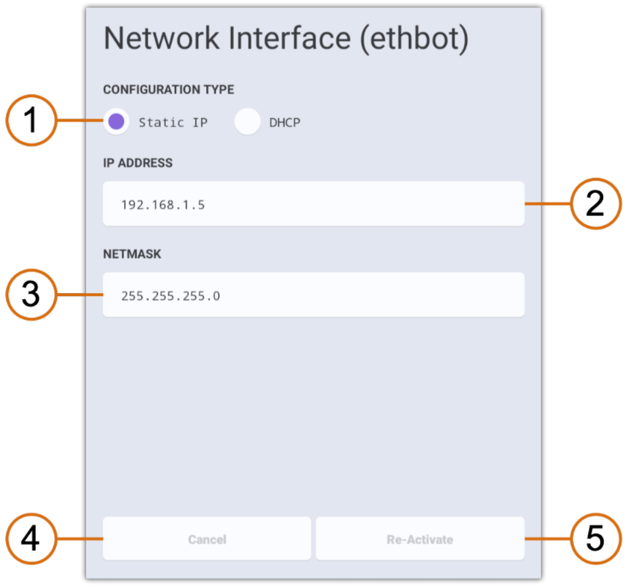
ETHBOT

“Ethbot” interface is reserved for the communication between cabinet and robot arm.

Ethbot configuration should be changed only by expert users that are aware of the possible consequences.

Options Panel

The network interface has the following Options panel.


Network Interface (ethbot)

CONFIGURATION TYPE

[✓ Static IP] DHCP

IP ADDRESS

192.168.1.5

NETMASK

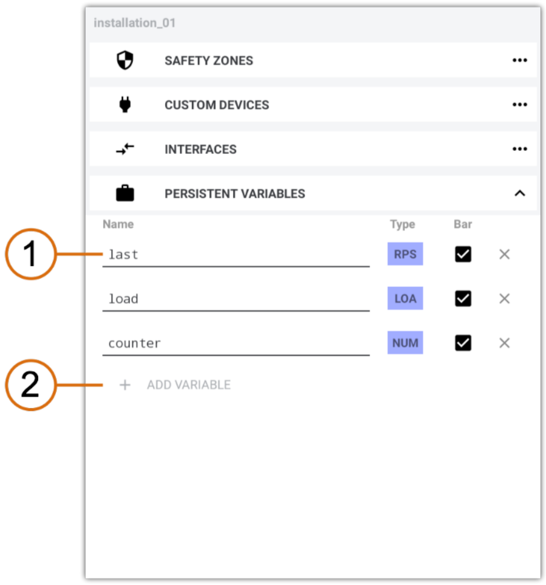
255.255.255.0

Item	Description
1	**Configuration Type** Specifies how the IP address assignment should be provided. The IP address can be either assigned automatically by a DHCP server (if available) or manually by the user (static IP address).
2	**IP Address** Allows the user to assign IP address manually (applies only for Static IP configuration type).
3	**Netmask** Allows the user to specify the network mask (applies only for Static IP configuration type).
4	**Cancel** Discards any local changes, ie. displays the active interface configuration.
5	**Activate** Applies all local changes.

7.4 Persistent Variables

The *Workcell* also includes a register of persistent variables. This register holds all custom variables that are configured with persistent scope. Unlike custom variables with local or global scope, persistent variables exist and keeps their value outside the program execution.

If the persistent variable scope is changed to local or global, the variable is removed from the *workcell* variables register and is placed into program variables register.



Name	Type	Bar
last	RPS	✓
load	LOA	✓
counter	NUM	✓

Item	Description
1	**Persistent Variable** Represents single persistent variable.
2	**Add Variable** Allows the user to create new persistent variable.

8 Software Extensions (CBuns)

The Kassow Software provide its own extensions technology, allowing the high degree of variability based on C++ interfaces and dynamic libraries. Check the list of RC pre-installed modules in Appendix D [Appendix – List of extensions (CBuns)].

The CBun (Capability Bundles) module represents the KR framework item, encapsulating the application, system or peripheral extensions. The set of basic extensions are now maintained by KR, but integrators and device manufacturers are invited to implement and provide support for their own devices and applications.

The basic CBuns control interface is located at the *CBun Manager* [Settings/CBuns], where the user review and adjust the set of loaded CBun modules. The CBun manager allows the basic package functionalities:

- Load CBun module [1]
- Unload CBun module [2]
- Add the CBun instance into *Workcell* [3]

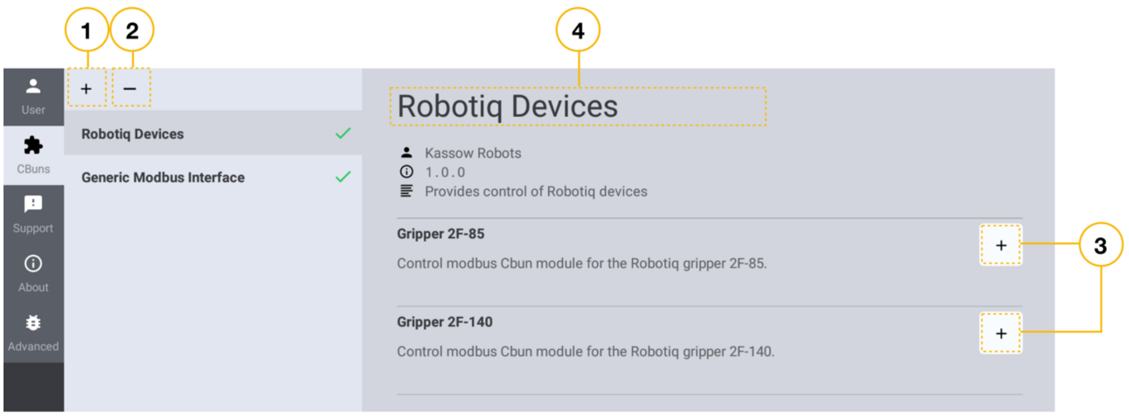

1
2
4
3
+
−
User
CBuns
Support
About
Advanced
Robotiq Devices
Generic Modbus Interface
Robotiq Devices
Kassow Robots
1.0.0
Provides control of Robotiq devices
Gripper 2F-85
Control modbus Cbun module for the Robotiq gripper 2F-85.
Gripper 2F-140
Control modbus Cbun module for the Robotiq gripper 2F-140.

Figure 12. CBuns Manager view

8.1 CBun Installation

The present KSW supports the installation of the CBun from three kinds of sources.

- CBuns coming with the core software package (CBuns maintained by Kassow Robots).
- Installation of the package located at the USB

- Compilation and installation from the custom CBun source code.

In this manual we describe only the first 2 options. The custom CBun development and compilation is beyond the scope of this handbook.

8.1.1 Installation from the KSW packages

When the installation dialog is opened, the default source is pointing to **Robot** folder [5]. That's the part of the KSW core software package containing the set of CBuns maintained by Kassow Robots.

Choose desired CBun, verify the version and click "Install" to proceed with the installation.

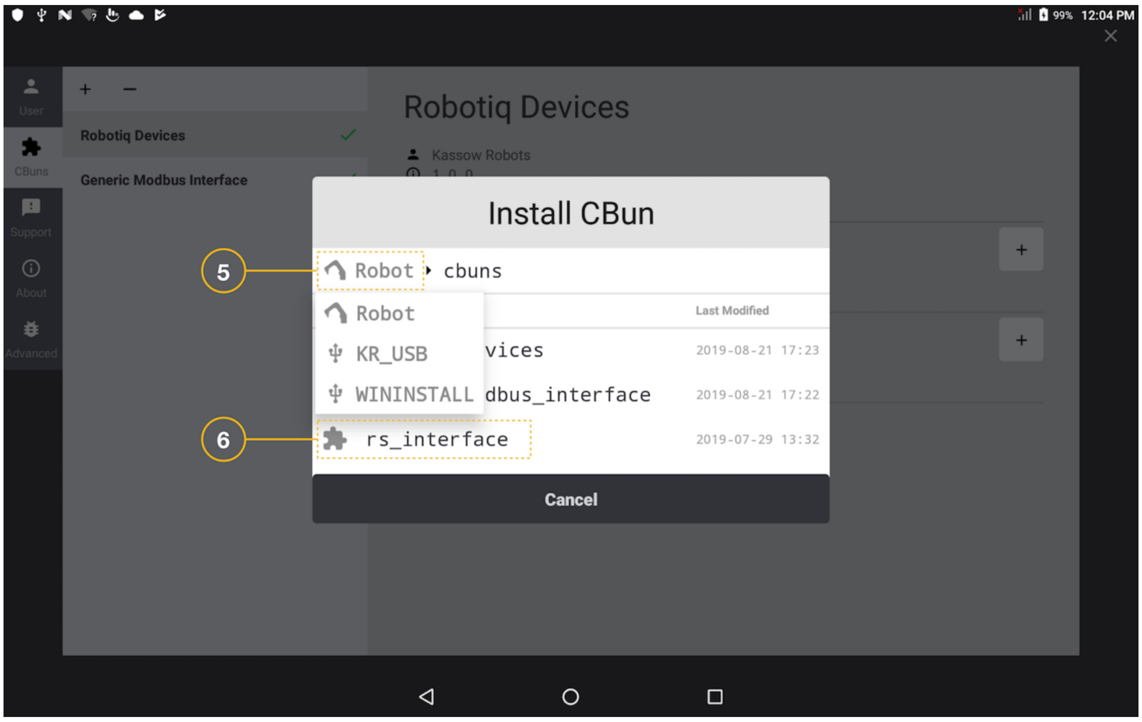


Figure 13. CBun installation dialog

8.1.2 Installation from USB

1) Put the CBun file (*.cbun) on to the USB flash disk.

2) Plug the USB flash with the CBun into RC.

3) Open CBuns Manager and click the "+" button in the top left corner [1].

4) At the install dialog chose the USB as install source.

5) Select the CBun for installation [5].

6) Confirm the installation by clicking the "Install" button.

8.2 Instance

The **CBun instance** is simply an abstraction of the used hardware (in case of the device type) or software class within your programs. When you instantiate the Robotiq or OnRobot gripper, it represents the device with the unique identification and communication settings in the context of your program.

The instance needs to be created and added into workcell before it can be used with the program sequence. It is then present within the workcell list of devices (in the future also other types of instances will be supported) and can be dragged into any part of the robot sequence to execute selected method/action.

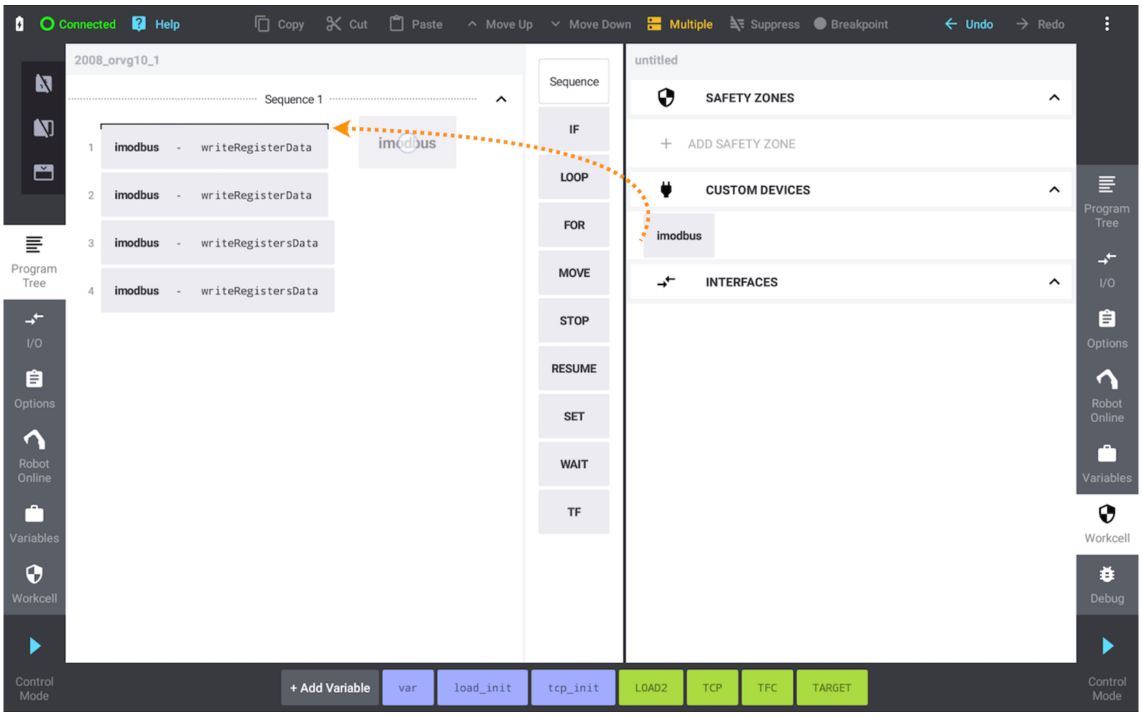


Figure 14. Using instance within the program sequence

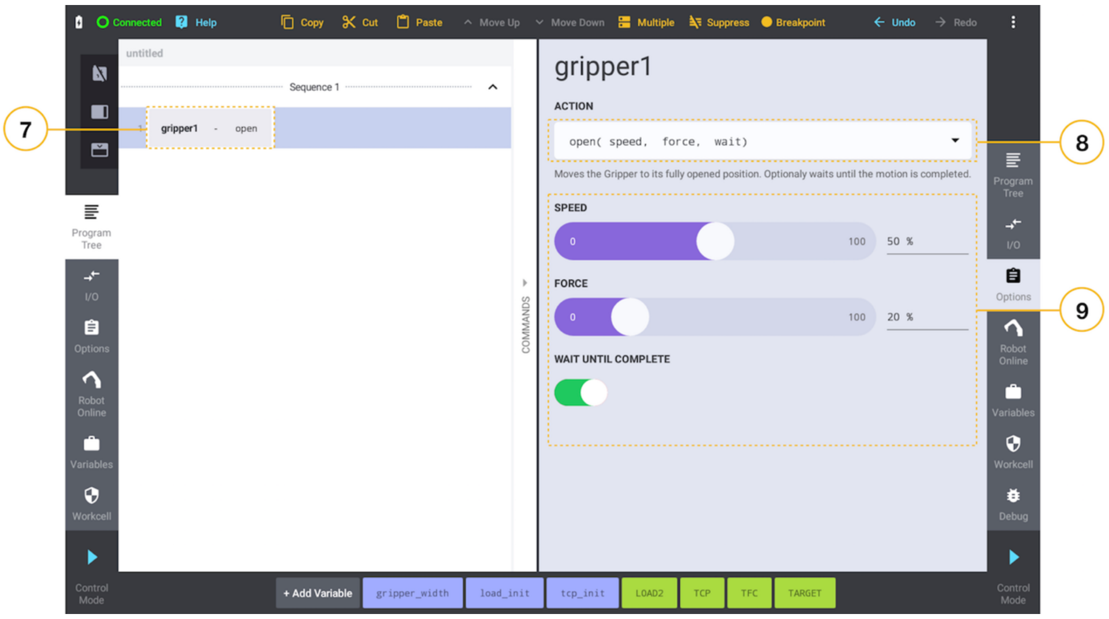
In general, any CBun contains multiple implementations and interfaces related together by the maintainer, type or other logical kinship (e.g., different types of grippers from one manufacturer).

The instance configuration and activation are accessible from the *Toolbar* or the *Workcell/Custom Devices*, where the user can fine-tune the instance settings.

8.2.1 Methods

Methods (denoted as "Actions" for the custom device CBuns) are the basic entry points to the CBun functionalities. Each instance has a list of available public methods which are accessible from the method selector within the instance Options.

When the instance of the CBun is placed into the program area [7], you need to choose one from the list of methods [8] and supply the necessary properties or arguments (method properties [9]).



gripper1

ACTION

`open( speed, force, wait)`

Moves the Gripper to its fully opened position. Optionaly waits until the motion is completed.

SPEED 0 100 50 %

FORCE 0 100 20 %

WAIT UNTIL COMPLETE

Figure 15. CBun instance method Options

9 Settings and Maintenance

The user interface allows various adjustments and modifications of the software configuration and the robot system setup. This involves also access to the user profiles, system information, software updates or even the robot maintenance tools.

In this chapter we share detailed information on following subjects:

- System updates
- Support and datalogs
- User profiles
- Maintenance toolbox (Firmware update and hardware diagnostics)
- User interface configuration

9.1 System Info and Updates

The summary of the system information (software parts and hardware) is available from the brief overview located in the [Settings/About]. This might be handful when contacting the KR support team for a service claim.

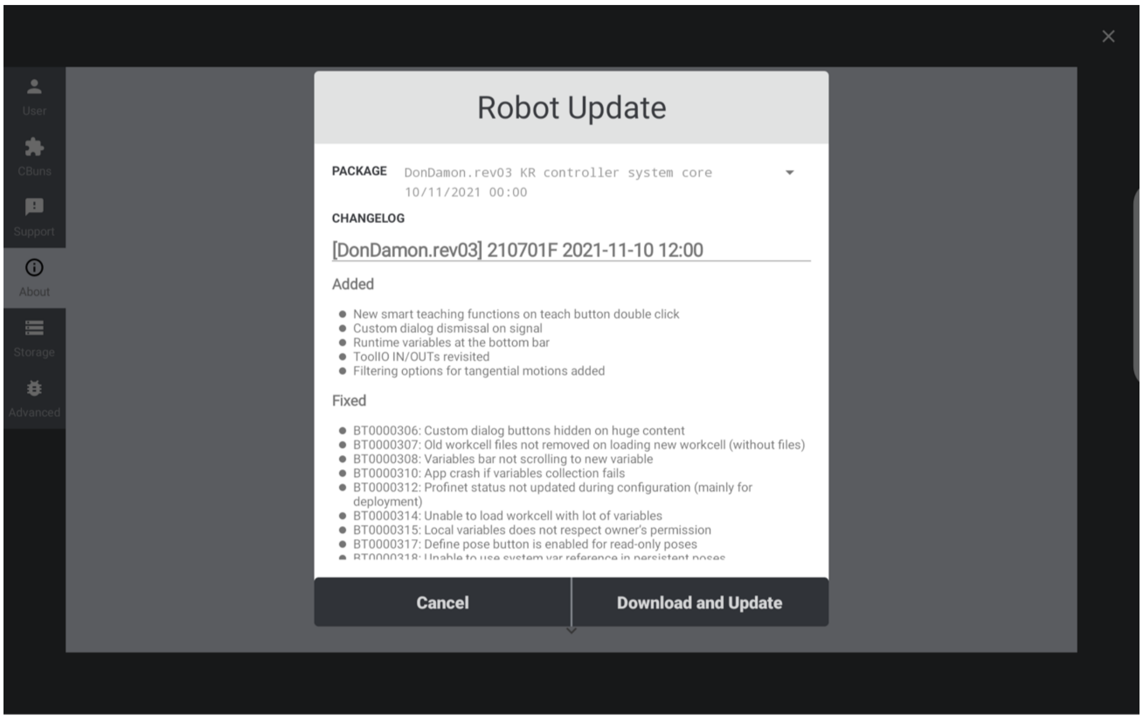
The simple access to the recent software update is one of the basic features of the Kassow. With the standard access to the Wi-Fi you can update your robot using 3 simple button clicks.

To update the robot RC software, proceed as follows:

1. Turn on the main RC switch and wait until the TP is connected.
2. Navigate to the Settings/About and use the *“Check for Updates button”*.
3. When connecting to the Wi-Fi for the first time, choose the network and fill in the connection details.

If the connection is properly set up, proceed by the *Continue* button:

1. After a short while the update dialog pops up and proposes the recent available update.
2. Choose *“Download and Update”*, the installation will start immediately. This could take approx. 5-10 minutes to complete.
3. After the installation was successfully completed, do restart the RC.



Robot Update

PACKAGE DonDamon.rev03 KR controller system core 10/11/2021 00:00

CHANGELOG

[DonDamon.rev03] 210701F 2021-11-10 12:00

Added

- New smart teaching functions on teach button double click
- Custom dialog dismissal on signal
- Runtime variables at the bottom bar
- ToolIO IN/OUTs revisited
- Filtering options for tangential motions added

Fixed

- BT0000306: Custom dialog buttons hidden on huge content
- BT0000307: Old workcell files not removed on loading new workcell (without files)
- BT0000308: Variables bar not scrolling to new variable
- BT0000310: App crash if variables collection fails
- BT0000312: Profinet status not updated during configuration (mainly for deployment)
- BT0000314: Unable to load workcell with lot of variables
- BT0000315: Local variables does not respect owner's permission
- BT0000317: Define pose button is enabled for read-only poses
- BT0000318: Unable to use system var reference in persistent poses

Figure 16. RC software update screen

The RC software update doesn't involve the firmware update automatically. If the robotic arm hardware or the cabinet IO requires the firmware upgrade, please follow the instructions from the maintenance section later in this chapter.

9.2 Support and Datalogs

For the sake of efficient robot trouble shooting and problem identification, the RC keeps a massive record of various kind of data (not longer than recent 2 minutes of the recent robot operations). Those data (datalogs) can be easily delivered to the safe and dedicated cloud storage.

Even though the data extraction procedure is not complicated, you need to follow few steps properly to share the meaningful data with the Kassow Robots support team:

0. Check the robot is not moving (pause or terminate the running program).
1. Press the ESTOP button.
2. Wait until the progress bar ceases (the data are being extracted in this phase).

3. Go to *[Settings/Support]* and use the *"Send Logs"* to upload the data into cloud.

4. Proceed with the Wi-Fi connection dialog and wait until the data are successfully uploaded.

❗ Please note that our team is not automatically notified when the new datalogs are uploaded. Always contact the Kassow Robots support to get assured your issue was reported and is scheduled for examination.

9.3 User Profiles

The Kassow software system supports 5 user profiles, which provide different level of access to various functions of the robot, system configurations or program contents.

The profiles are inclusive. Which means that the higher privileged profile has the access also to the features of any lower privileged profile.

In the following table you can find the overview of supported profiles and list of features accessible to each of them.

User Profile	Allowed features
OPERATOR Can only start and stop programs.	• robot start • program start/stop
PROGRAMMER The low access profile. Allowed to load or modify the program. No safety or the workcell can be provided by this user.	• copy/cut/paste on commands/variables • command suppress option • undo/redo • modify program (add, remove, move commands, change params) • modify variables (add, remove variables, change params) • set breakpoint • convert to subprogram • program management (new/open/save/delete) • pause program • enable/disable breakpoints • update system • send logs • control I/O
SAFETY ADMIN Safety functions and the workcell settings can be modified by this user.	• copy/cut/paste on safety zones, (custom devices, persistency variables) • suppress on safety zones, (custom devices)?!? • import workcell • workcell management (new/open/save/delete)

	• set safety mode, master speed • modify safety zones (add, remove, change params) • modify custom devices, persistent variables (add, remove, change params) • undo/redo
INTEGRATOR All documented functions of the system are available for this profile.	• modify interfaces
ADMIN Can access all the system features with no limits.	• advanced settings access • maintenance mode • firmware update

9.4 Maintenance toolbox

Aside from the rest of regular functionalities, the maintenance toolbox provides a set of specialised tools to access the hardware firmware updates (Joints, IO-Board, Tool-IO) and some diagnostic routines.

The main window of this tool can be entered from the [Settings/Advanced/Controller...].

9.4.1 Diagnostics

To help identify the RC or hardware related issues, the maintenance toolbox contains several basic views. The *General Information* pan provides some basic information about the hardware.

HW Diagnostics can be populated with the actual data extracted from the RC. Use the refresh button to update the data. In some case this tool can be useful on efficient inspection of the hardware issues, related to IO, Joint Boards or Tool-IO.

9.4.2 Factory Poses

Factory poses provide the instant access to some default robot joint configurations, which can be useful or recommended for the firmware update or robot box packing (when preparing the robot for shipping).

Button	Description
Move to Zero Pose	Unfold the robot into all zero joint angles pose
Move to Transport Pose	Fold the robot into transportation pose

9.4.3 Firmware Update

The firmware update allows to improve and extend functionalities of the embedded/electronic parts of the robot (namely the IO-Board, Joint Boards and Tool-IO).

Here we us the firmware term to refer to any software running by all the robot hardware parts (meaning IO-Board, Joints and Tool-IO). This bit of software keeps all electronic parts alive, providing the proper control and communication on each active part of the robot.

As mentioned earlier, the firmware update is not a part of the standard RC software update routine. Typically, it might be recommended by the Kassow team after some significant improvements on hardware control or the communication level and in some accordance with the RC software version.

The firmware packages are stored using the **.fwu* extension and can deliver the simple joint or even the whole set of updates for each part of the robot.

Procedure

The regular procedure for the firmware update consists of following steps

1. **Choose the media source**
   You can install the *fwu* package from the Google drive or the USB storage. For the first option make sure your Google account is set up properly [31]. To go with the USB put the package onto the USB stick and plug it into the RC. But don't initialize the robot (don't use the blue blinking toggle button).

2. **Restart the RC**
   But don't initialize the robot (don't use the blue blinking toggle button).

3. **Enter the Maintenance Mode**
   Open the [Settings/Advanced/Controller...] tab. Locate the *"Maintenance Mode"* section, then press *"Enable"* button and enter the admin password when prompted.

4. **Proceed with the Maintenance Mode**
   Wait until the toggle starts blinking (cyan), then press the toggle. The robot should switch into Maintenance mode after a while (purple).

5. **Start the update**
   Press the *"Update"* button located within the *Firmware Update* section and enter the admin password.

6. **Locate and choose the package**
   Choose the media source [31]. Then navigate to the intended fwu package, click and confirm with "*Proceed"*.

7. **The firmware update progress**
   Wait until the update process was successfully finished. If there was a failure, please contact the Kassow team and share with us any necessary information (RC software version, fwu package). Eventually extract and upload the datalogs.

8. **Disable the Maintenance Mode**
   Use the *"Maintenance Mode/Disable"* button to return back to regular mode and provide the regular robot initialisation routine.

9.5 Advanced Settings

Some special settings of the Teach pendant can be provided from the [Settings/Advanced] tab, accessible only for the “Admin” user. The overview of accessible features is stated in the following table.

Section	Description
ADVANCED CONFIGURATION	Allows to export and import the RC and robot configuration set (cfg file). This can be useful when some unstandard typical changes on the robot configuration are necessary. The USB stick is only medium support ed for the cfg file export and import. It has to be plugged into the RC USB outlet (if availa-ble) or directly to the RC Motherboard.
SINGLE APP	The single app mode is the default setting which doesn’t allow the TP app exit back into the android system or provide the switch into any other application.
DEBUG MODE	This option allows to access several tools which are handful for some system and software analysis.
DEVICE OWNER	The system default setting. Don’t disable the option until in-structed by the Kassow Robots support team.
TABLET SCREEN LOCK	The system default setting. Don’t disable the option until in-structed by the Kassow Robots support team.
ANDROID DEBUG BRIDGE	The system default setting. Don’t disable the option until in-structed by the Kassow Robots support team.
TABLET LOGS	Explicit request to send the application crash data into the cloud (useful in cases of failure or application crashes).
ROBOT CONFIGURATION	The robot model configuration. Please consult with the Kassow Robots support team before changing the predefined settings.
ROBOT CONTROLLER	The maintenance and diagnostic tools.

A Appendix - Tool IO

New robots equipped by the Tool-IO provide an efficient interface for the grippers, sensors, cameras or other peripherals, excluding external wiring and using various control and communication options when accessed from the application programs.

Our first version of the Tool IO supports several hardware communications interfaces, particularly Digital/IO, RS485, Ethernet. It also provides the necessary power supply and the back-drive button access for convenient teach functions.

Various features of the Tool-IO can be accessed directly from the UI. Using a simple variable mapping, the Digital/IO can be used easily within the robot programs.

Description of the hardware and how it routes to the UI is stated in the following charts.

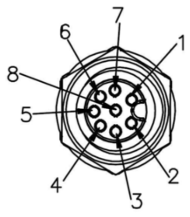


Figure 17. Connector 1, Tool-IO M8 female, 8 poles

Digital IO		Pin no.	Configurations	Specification
-	-	8	-	GND
TPSU01	OUT	5	12V or 24V	Power output 1, 12V or 24V +5% / -15% Max 700mA *
TDO01	OUT	7	12V or 24V	Digital output 1, 12V or 24V +5% / -15% Max 100mA *
TDO02	OUT	6	12V or 24V	Digital output 2, 12V or 24V +5% / -15% Max 100mA *
TAO01	OUT	4	4-20mA or 0-10V	Analog output 1, configurable current or voltage
TAO02	OUT	4	4-20mA or 0-10V	Analog output 2, configurable current or voltage
TAI05	IN	1	-	Analog input 1, 4-20mA in +, Max 12V relative to GND
	IN	2	-	Analog input 2, 4-20mA in -, Max 12V relative to GND

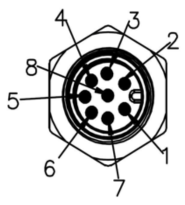


Figure 18. Connector 2,
Tool-IO male M8, 8 poles

Digital IO		Pin no.	Configurations	Specification
-	-	8	-	GND
TPSU02	OUT	5	12V or 24V	Power output 2, 12V or 24V +5% / -15% Max 700mA *
TDO03	OUT	7	12V or 24V	Digital output 3, 12V or 24V +5% / -15% Max 100mA *
TDO04	OUT	6	12V or 24V	Digital output 4, 12V or 24V +5% / -15% Max 100mA *
TAI03	IN	4	-	Input 3, Digital or Analog 1-10V Max 30V
TAI04	IN	3	-	Input 4, Digital or Analog 1-10V Max 30V
IO/API	IN	1	-	RS 485 + baud rate programmable up to 1Mbps
	IN	2	-	RS 485 - baud rate programmable up to 1Mbps

**Note. The total combined current that can be drawn from the Tool IO is below 800mA @24V or 1.6A @12V. Exceeding the specification may damage the electronics or cause robot to stop.*

B Appendix - Expressions

The *expression* is a combination of one or more operands (literals, variables, functions) and operators. This compound is evaluated during the program execution and returns a calculated result. It is an important part of various program command inputs.

B.1 Operators

The following tablet lists all available operators and their precedence. Operators are listed top to bottom, in descending precedence.

Precedence	Operator	Description
1	a[i, j]	Array indexing
2	a.b	Member access
3	-a !a (not a)	Unary minus Logical not
4	a * b a / b	Multiplication Division
5	a + b a - b	Addition Subtraction
6	a < b a <= b a > b a >= b	Less Less or equal Greater Greater or equal
7	a = b a != b	Equality Non-equality
8	a & b (a and b) a | b (a or b)	Logical AND Logical OR

B.2 Members

The following table lists available members of different data types. Properties are accessible via dot notation.

Data Type	Operator	Description	Member Type
Number	-	-	-
Position	a.x	X	Number
	a.y	Y	Number
	a.z	Z	Number
Orientation	a.rpy	Roll, Pitch, Yaw	Number[3]
	a.quat	Quaternion	Number[4]
Configuration	a.j1	Joint 1	Number
	a.j2	Joint 2	Number
	a.j3	Joint 3	Number
	a.j4	Joint 4	Number
	a.j5	Joint 5	Number
	a.j6	Joint 6	Number
	a.j7	Joint 7	Number
Pose	a.pos	Position	Position
	a.rot	Orientation	Orientation
	a.conf	Configuration	Configuration
	a.ref_pose	Reference Pose	Pose
Inertia Matrix	a.xx	Moment of inertia	Number
	a.yy	Moment of inertia	Number
	a.zz	Moment of inertia	Number
	a.xy	Moment of inertia	Number
	a.xz	Moment of inertia	Number
	a.yz	Moment of inertia	Number
Load	a.pose	Pose	Pose
	a.mass	Mass	Number
	a.cog	Center of gravity	Position
	a.imx	Inertia matrix	Inertia Matrix
Grid Axis	a.pose	Final pose	Pose
	a.steps	Number of steps	Number
GridPattern	a.pose	Initial pose	Pose
	a.axis1	Axis 1	Grid Axis
	a.axis2	Axis 2	Grid Axis
	a.axis3	Axis 3	Grid Axis
	a.next	Next pose	Pose
	a.count	total count	Number
Array	a.size	Number of rows	Number

B.3 Functions

Current version of Kassow Robots control software provides built-in functions divided into modules.

B.3.1 Math Module

Math module provides mathematical functions for manipulation with Numbers.

Constants

Function	Description
math.pi()	Returns the mathematical constant π = 3.141592...
math.e()	Returns the mathematical constant e = 2.718281...

Theoretic and Representation

Function	Description
math.abs(x)	Returns the absolute value of x.
math.ceil(x)	Returns the ceiling of x, i.e., the closest integer value greater than or equal to x.
math.floor(x)	Returns the floor of x, i.e., the closest integer value less than or equal to x.
math.round(x)	Returns the closest integer value to x.
math.trunc(x)	Returns the integer part of x.

Power and Logarithmic

Function	Description
math.exp(x)	Returns e^x.
math.log(x)	Returns the natural logarithm of x (to base e).
math.log10(x)	Returns the base-10 logarithm of x.
math.pow(x, y)	Returns x raised to the power x, ie. x^y.
math.sqrt(x)	Returns the square root of x.

Trigonometric

Function	Description
math.acos(x)	Returns the arc cosine of x (in radians).
math.asin(x)	Returns the arc sine of x (in radians).
math.atan(x)	Returns the arc tangent of x (in radians).
math.atan2(x, y)	Returns the arc tangent of y/x. Takes in account signs of both inputs, so it can compute the correct quadrant and return the result between - π and π.
math.cos(x)	Returns the cosine of x in radians.

math.hypot(x)	Returns the Euclidian norm, ie. sqrt(x*x + y*y). This is the length of the vector form from the origin to point (x, y).
math.sin(x)	Returns the sine of x in radians.
math.tan(x)	Returns the tangent of x in radians.

Angular Conversion

Function	Description
math.degrees(x)	Converts angle x from radians to degrees.
math.radians(x)	Converts angle x from degrees to radians.

Min & Max

Function	Description
math.min(x, y,...)	Returns the smallest number of two or more arguments.
math.max(x, y,...)	Returns the biggest number of two or more arguments.

B.3.2 Program Module

Program module provides function related to program execution.

Function	Description
program.time()	Returns time in seconds since the program execution start as a floating-point number. The precision is 1 microsecond.
program.debug()	Returns 1 if the program is in debug mode, otherwise returns 0.

B.3.3 Geometry Module

Geometry module provides functions for manipulation with geometry objects (such as poses).

Function	Description
geometry.dist(A, B)	Returns the distance between pose A and pose B in millimetres.

C Appendix - System Variables

C.1 I/O

Name	Type	Description
Digital Input (1- 16)	Input	16x analog digital inputs (IO Board, 30 V)
Current Input (1, 2)	Input	2x analog inputs (IO Board, 20 mA)
Voltage Input (1, 2)	Input	2x voltage input (IO Board, 10 V)
Relay Output (1- 4)	Output	4x relay output (IO Board)
Digital Output (1- 8)	Input	8x digital output (IO Board, 24 V)
Current Output (1- 2)	Output	2x current output (IO Board, 20 mA)
Voltage Output (1- 2)	Output	2x voltage output (IO Board, 10 V)
Tool-IO Digital Output (1- 4)	Output	4x digital output (Tool Board, 12 or 24 V)
Tool-IO Power Supply Output (1- 2)	Output	2x power supply output (Tool Board, 12 or 24 V)
Tool-IO Analog Output (1- 2)	Output	2x analog output (Tool Board, 20 mA or 10 V)
Tool-IO Analog Input (1- 4)	Input	4x analog input (Tool Board, 20 mA, 10 V)

C.2 Frames

Name	Type	Description
Robot Base Frame	Output	Attached to the base link of the robot.
Robot Link 1 Frame	Input	Attached to the link 1 of the robot.
Robot Link 2 Frame	Input	Attached to the link 2 of the robot.
Robot Link 3 Frame	Input	Attached to the link 3 of the robot.
Robot Link 4 Frame	Input	Attached to the link 4 of the robot.
Robot Link 5 Frame	Input	Attached to the link 5 of the robot.
Robot Link 6 Frame	Input	Attached to the link 6 of the robot.
Tool Flange Centre Frame (TFC)	Input	Attached to the tool flange of the robot.
Load 1 Frame	Output	Attached to the load 1 (tool load).
Load 2 Frame	Output	Attached to the load 2 (payload).
Tool Centre Frame (TCP)	Output	Exact working point of the robot tool.
Last Target Pose	Input	Last target in robot trajectory.

C.3 Loads

Name	Type	Description
Load 1	Output	Describes the rigid body of the attached tool.
Load 2	Output	Describes the rigid body of the attached payload.

C.4 Arrays

Name	Size	Type	Description
Robot Joint Angles	7	Input	Angles [rad] of individual robot joints.
Robot Joint Velocities	7	Input	Velocities [rad/s] of individual robot joints.
Robot Joint Accelerations	7	Input	Accelerations [rad/s^2] of individual robot joints.
Robot Joint Torques	7	Input	Torques [Nm] of individual robot joints.
Safety Joint Torque Deviation [raw]	7	Input	Internal torque deviation values (No filtering applied)
Safety Joint Torque Deviation [filtered]	7	Input	Internal torque deviation values (Smoothed by a filter)

D Appendix – List of extensions (CBuns)

Here we share the list of the official extension modules maintained by the KR. These modules allow to integrate several external devices (e.g., grippers) or to access more specialised system functions.

CBun Module	Description
Generic Modbus Interface	This extension module provides numerous published methods to access Modbus RTU slave device connected to the RC through the RS485, RS232 or the USB (RS adapter) interface.
System Utils	Provides access to various system extensions and functionalities that are not included in the standard user interface (yet). Contains the safety events mapping to I/O (Event Watcher) or interfaces the control program execution and handle alarms (Program Control).
STEP file processor	CAD based curves can be imported and processed by the software into KR robots programming environment. By using this CBun it is possible to let the robot follow the defined trajectory which makes the process of teaching robot complex movements much easier.
Robotiq grippers	Robtiq grippers (2F-85, 2F-140, Hand-E and the vacuum EPick) are easy to integrate by using this CBun. The Robotiq force-torque sensor support (FT300/FTS300) are also included in this package.
OnRobot grippers	KR software installation also includes the OnRobot extension module, but it provides support only for the limited set of OnRobot devices. Check the OnRobot software support website for the full range support of their products integration with the KR: https://onrobot.com/en/robot-compatibility/onrobot-solutions-for-kassow

The documentation and more detailed information about the recent versions of those modules are available at https://docs.kassowrobots.com/. To get the access to this resource please contact the Kassow Robots technical support team.