SEW EURODRIVE

Manual

MOVIKIT®
Robotics

Edition 03/2020

29180252/EN

Table of contents

1 General information 8
1.1 About this documentation 8
1.2 Content of the documentation 8
1.3 Structure of the safety notes 8
1.3.1 Meaning of signal words 8
1.3.2 Structure of section-related safety notes 8
1.3.3 Structure of embedded safety notes 9
1.4 Decimal separator in numerical values 9
1.5 Rights to claim under limited warranty 9
1.6 Product names and trademarks 9
1.7 Copyright notice 9
1.8 Other applicable documentation 10
1.9 Short designation 10

2 Safety notes 11
2.1 Preliminary information 11
2.2 Target group 11
2.3 Network security and access protection 11
2.4 Designated use 11

3 System description 12
3.1 Module description 12
3.1.1 Advantages 12
3.1.2 Operating principle 13
3.2 Functions 14
3.3 Add-ons 15
3.3.1 MOVIKIT® Robotics add-on MediumModels 15
3.3.2 MOVIKIT® Robotics add-on Touchprobe 15

4 Basics 16
4.1 Kinematic models 16
4.1.1 Basic types 16
4.1.2 Constellation 17
4.1.3 Transformation between axes and joint axes 18
4.2 Motion control 18
4.2.1 Interpolation 18
4.2.2 Path 18
4.2.3 Blending 19
4.2.4 Motion profiles 20
4.2.5 Sets of motion parameters 21
4.2.6 Scaling using override 21
4.2.7 Default units 21
4.2.8 Emergency stop 22
4.2.9 Path events 22
4.2.10 Touchprobe 22
4.3 Coordinate systems 23

4.4 Communication and process data exchange ... 23

5 Project planning information ... 24

5.1 Requirement ... 24

5.2 Hardware ... 24

5.3 Software ... 24

5.4 Licensing ... 25

6 Startup ... 26

6.1 Requirements ... 26

6.2 Startup procedure ... 26

6.3 Configuring a project ... 27

6.3.1 Example project ... 27

6.3.2 Configuring the MOVI-C® CONTROLLER ... 28

6.3.3 Adding MOVIKIT® Robotics ... 29

6.3.4 Configuring subordinate nodes ... 30

6.3.5 Configuring MOVIKIT® Robotics ... 32

6.4 Generating an IEC project ... 42

6.4.1 IEC project structure ... 43

6.5 Importing the MOVIKIT® fieldbus monitor ... 44

6.6 Starting the control program ... 44

6.7 Installing the RobotMonitor ... 45

6.7.1 System requirements ... 45

6.8 Starting the RobotMonitor ... 45

6.9 Establishing the connection to the controller ... 46

6.10 Managing users (optional) ... 46

6.11 Setting sets of motion parameters ... 46

6.12 Referencing axes and performing function test ... 48

6.12.1 Function test for the simulated robot ... 48

6.12.2 Referencing and function test on the real robot ... 48

6.13 Programming the robot ... 49

6.14 Integrating the robot ... 49

7 Functional description ... 50

7.1 Robot states ... 50

7.1.1 Operating mode ... 50

7.1.2 Enable mode ... 51

7.1.3 Enable state ... 53

7.1.4 Type of control ... 54

7.2 Jog mode ... 54

7.3 Program mode ... 55

7.3.1 Types of program sequence ... 56

7.3.2 Establishing the basic state ... 56

7.4 Repositioning ... 57

7.5 Access management ... 58

7.6 Simulation ... 59

7.7 Kinematic models ... 60

7.7.1 Emergency stop ramps of the joints ... 60

7.7.2 CARTESIAN_GANTRY_LL_M10 61
7.7.3 CARTESIAN_GANTRY_LLL_M10 61
7.7.4 CARTESIAN_GANTRY_LLR_M10 62
7.7.5 CARTESIAN_GANTRY_LLR_M20 62
7.7.6 CARTESIAN_GANTRY_LLLR_M10 63
7.7.7 ROLLER_GANTRY_RR_M10 64
7.7.8 ROLLER_GANTRY_RRR_M10 65
7.7.9 ROLLER_GANTRY_RRLR_M10 66
7.7.10 SCARA_RR_M10 67
7.7.11 SCARA_RRR_M10 68
7.7.12 SCARA_LRRR_M10 69
7.7.13 SCARA_RRRR_M10 70
7.7.14 DELTA_RR/RRR_M10 71
7.7.15 TRIPOD_RRR/RRRR_M10 73
7.7.16 MIXED_LR_M10 75
7.7.17 MIXED_LRR_M10 76
7.7.18 MIXED_LRR_M20 77
7.7.19 MIXED_LRRL_M10 78
7.7.20 MIXED_LRLR_M10 79
7.7.21 MIXED_RLLR_M10 80
7.8 Path events 81
7.9 Touchprobe 86
7.9.1 Activation/deactivation 86
7.9.2 Triggering and measuring 88
7.9.3 Sensor-based positioning 89
7.9.4 Touchprobe as path event 92
7.9.5 Using path events and touchprobe simultaneously 92

8 Control via the process controller 93
8.1 Process controller 93
8.2 Starting up the process controller 93
8.3 Important status signals 93
8.4 Prerequisites for movements of the robot 94
8.5 Process start 95
8.6 Process sequence 95
8.7 Process stop 97
8.8 Handling errors 97
8.8.1 Limitation of the work envelope 97
8.9 Repositioning required 98

9 Robot program 99
9.1 Requirements 99
9.2 User interface 99
9.2.1 General robot control 100
9.2.2 General status and diagnostic messages of the robot 100
9.2.3 User management 101
9.2.4 3D simulation 104

9.2.5 Program status and control of program operation 105
9.2.6 Jog mode control 105
9.2.7 Operator panel 106
9.3 Creating the robot program 107
9.3.1 Creating a new program 108
9.3.2 Editing an existing program 108
9.3.3 Copying an existing program 109
9.3.4 Adding commands 109
9.3.5 Loading a standard program 110
9.4 Saving the robot program 110
9.5 Executing the robot program 111
9.6 Teaching poses 112
9.6.1 Teaching a pose variable 112
9.6.2 Explicit teaching including inserting a linear segment 112
9.7 Motion commands 113
9.7.1 Linear segments 113
9.8 Assignments 115
9.8.1 Setting translatory motion properties 115
9.8.2 Setting parameters for blending 116
9.8.3 Adding a motion parameter set assignment 119
9.8.4 Adding a Boolean assignment 119
9.8.5 Adding real assignment/calculation 120
9.9 Control structures 121
9.9.1 Conditional statement 121
9.9.2 Loop 122
9.9.3 Wait command 123
9.9.4 IEC function call 124
9.9.5 End of program 124
9.9.6 Subroutine call 125
9.9.7 Path event 125
9.9.8 Touchprobe 127
9.10 Variables 130
9.10.1 Editing robot program variables 130

10 IEC programming 131
10.1 Opening the IEC project 131
10.2 User interface 131
10.3 Diagnostics 132
10.4 Access management 132
10.5 Basic functions (basic) 133
10.6 Inverter functions (Inverter) 136
10.7 Jog 138
10.8 Program control (Prg) 139
10.9 Program variables (PrgVar) 141
10.10 IEC function call for the robot program 142

11 Process data assignment 143

11.1 Fieldbus interface 143
11.1.1 Fieldbus profiles 143
11.1.2 Consistent data transmission 144
11.1.3 Control words 1-4 145
11.1.4 Status words 1-4 147
11.1.5 Standard profile for positioning 149
11.1.6 Flexible parameterizable profile 152

12 Diagnostics 156
12.1 MOVIKIT® fieldbus monitor 156
12.1.1 User interface 157

13 Application examples 158
13.1 Non-safe monitoring of robot behavior 158
13.2 Switching to single axis control 158
13.2.1 Switching from robot to single axis 158
13.2.2 Switching from single axis to robot 158
13.3 Using external selector switch for operating modes 159
13.4 Using free function keys of the operator panel 159
13.5 Combining the RobotMonitor with visualization 159
13.6 Setting motion parameters 161
13.6.1 Speed 161
13.6.2 Acceleration 161
13.6.3 Jerk 161
13.7 Optimizing the cycle time 162

14 Fault management 163
14.1 Troubleshooting 163
14.1.1 The robot is not carrying out any motion 163
14.1.2 Communication not possible between RobotMonitor and MOVI-C® CONTROLLER 164
14.1.3 Robot program not found 165
14.2 Fault codes 166
14.2.1 Robotics errors 166
14.2.2 Robotics messages 173
14.2.3 Subordinate software module errors 174
14.2.4 Subordinate software module messages 179

Index 180

1 General information

1.1 About this documentation

This documentation is an integral part of the product. The documentation is intended for all employees who perform work on the product.

Make sure this documentation is accessible and legible. Ensure that persons responsible for the systems and their operation as well as persons who work with the product independently have read through the documentation carefully and understood it. If you are unclear about any of the information in this documentation, or if you require further information, contact SEW-EURODRIVE.

1.2 Content of the documentation

The descriptions in this documentation apply to the software and firmware versions applicable at the time of publication. These descriptions might differ if you install later software or firmware versions. In this case, contact SEW-EURODRIVE.

1.3 Structure of the safety notes

1.3.1 Meaning of signal words

The following table shows the grading and meaning of the signal words for safety notes.

Signal word	Meaning	Consequences if disregarded
⚠ **DANGER**	Imminent hazard	Severe or fatal injuries
⚠ **WARNING**	Possible dangerous situation	Severe or fatal injuries
⚠ **CAUTION**	Possible dangerous situation	Minor injuries
NOTICE	Possible damage to property	Damage to the product or its environment
INFORMATION	Useful information or tip: Simplifies handling of the product.	

1.3.2 Structure of section-related safety notes

Section-related safety notes do not apply to a specific action but to several actions pertaining to one subject. The hazard symbols used either indicate a general hazard or a specific hazard.

This is the formal structure of a safety note for a specific section:

⚠

SIGNAL WORD

Type and source of hazard.

Possible consequence(s) if disregarded.

- Measure(s) to prevent the hazard.

29180252/EN – 03/2020

Meaning of the hazard symbols

The hazard symbols in the safety notes have the following meaning:

Hazard symbol	Meaning
⚠	General hazard

1.3.3 Structure of embedded safety notes

Embedded safety notes are directly integrated into the instructions just before the description of the dangerous action.

This is the formal structure of an embedded safety note:

⚠ **SIGNAL WORD!** Type and source of hazard. Possible consequence(s) if disregarded. Measure(s) to prevent the hazard.

1.4 Decimal separator in numerical values

In this document, a period is used to indicate the decimal separator.

Example: 30.5 kg

1.5 Rights to claim under limited warranty

Read the information in this documentation. This is essential for fault-free operation and fulfillment of any rights to claim under limited warranty. Read the documentation before you start working with the product.

1.6 Product names and trademarks

The brands and product names in this documentation are trademarks or registered trademarks of their respective titleholders.

1.7 Copyright notice

© 2020 SEW-EURODRIVE. All rights reserved. Unauthorized reproduction, modification, distribution or any other use of the whole or any part of this documentation is strictly prohibited.

1.8 Other applicable documentation

Observe the corresponding documentation for all further components.

Always use the latest edition of the documentation and the software.

The SEW-EURODRIVE website (www.sew-eurodrive.com) provides a wide selection of documents for download in various languages. If required, you can also order printed and bound copies of the documentation from SEW-EURODRIVE.

1.9 Short designation

The following short designations are used in this documentation:

Type designation	Short designation
MOVIKIT® Robotics	Software module
Complete system comprising hardware and software components for carrying out the motion tasks	Robot
SEW Robot Language	SRL
MOVISUITE® RobotMonitor	RobotMonitor

2 Safety notes

2.1 Preliminary information

The following general safety notes serve the purpose of preventing injury to persons and damage to property. They primarily apply to the use of products described in this documentation. If you use additional components, also observe the relevant warning and safety notes.

2.2 Target group

Software specialist

Any work with the software may only be performed by a specialist with suitable training. A specialist in this context is someone who has the following qualifications:

- Appropriate training
- Knowledge of this documentation and other applicable documentation
- SEW-EURODRIVE recommends additional training for products that are operated using this software.

2.3 Network security and access protection

A bus system makes it possible to adapt electronic drive technology components to the particulars of the machinery within wide limits. There is a risk that a change of parameters that cannot be detected externally may result in unexpected but not uncontrolled system behavior and may have a negative impact on operational safety, system availability, or data security.

Ensure that unauthorized access is prevented, especially with respect to Ethernet-based networked systems and engineering interfaces.

Use IT-specific safety standards to increase access protection to the ports. For a port overview, refer to the respective technical data of the device in use.

2.4 Designated use

MOVIKIT® Robotics is a software module for the MOVI-C® CONTROLLER used for path control of robots and to jog and reference robot axes.

Use the device-independent MOVISUITE® engineering software to start up and configure the axes and to download the complete configuration to a MOVI-C® CONTROLLER.

3 System description

3.1 Module description

In automation, handling a product is an important step in the process chain. Loading and unloading directly contributes to the cycle time of the machine. The motion profile must be fast and at the same time gentle on the product and on the mechanical components.

When a drive positions a workpiece forwards or backwards, it moves in one dimension. It is easy to optimize the positioning time and consequently the process. With only 2 axes, the workpiece already moves in a two-dimensional space. It is difficult to know which motion profile is best to ensure that handling is as fast as possible while still providing process stability. The positions in the space can often vary and be reached via different paths.

Typical applications are pick and place, palletizing, and secondary packaging. Applications such as plotting, applying glue, or finishing products, e.g. decorating foods, have similar requirements. These processes can be static or dynamic.

The MOVIKIT® Robotics module was developed precisely for these fields of application. This module is the ideal solutions platform for implementing a path motion just as easily and optimally as the movement of individual axes.

The MOVIKIT® Robotics software module in detail:

- Can be run with the software platform MOVIRUN® flexible
- Large selection of standard kinematic models (special kinematics are available on request)
- 3D simulation of the robot to reduce startup time and as additional protection against incorrect configuration
- Intuitive configuration of the robot through adaptive 3D models
- Programming and teaching within the SEW-EURODRIVE programming environment (also directly from the keypad)
- Optimal support of handling tasks such as pick and place or conveyor tracking
- Bidirectional coupling with IEC-61131 runtime system for flexibility in a wide range of applications.

3.1.1 Advantages

The advantages of the software module at a glance:

- Tried and tested encapsulated path control for robots
- Fast and easy startup of a robot thanks to the intuitive wizard and user-friendly diagnostics and monitoring functions
- Intuitive robot language for configuring the robot
- Reduction in cycle time by means of the synchronous path control featuring customizable blending and travel around interfering edges true to the outline

3.1.2 Operating principle

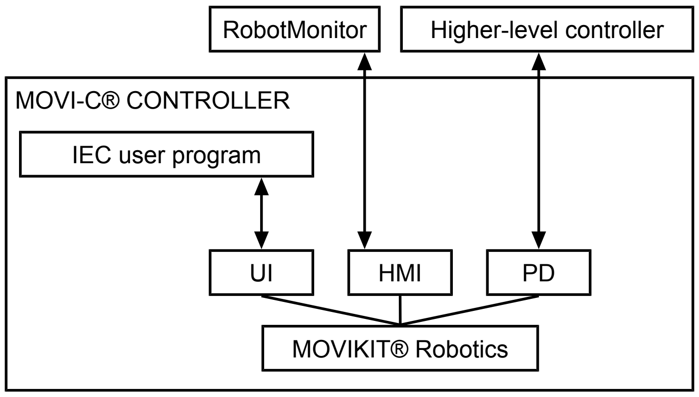
Process controller

The software module is controlled by a process controller. By starting and (optionally) parameterizing robot programs, the process controller tells the software module which paths are followed with which sets of motion parameters.

Process control can be implemented as follows:

- Via the MOVISUITE® app RobotMonitor on any panel
- Via control by means of an IEC user program
- Via control by means of a higher-level controller (PLC)

Control



31654692235

RobotMonitor

The motion path is parameterized in the RobotMonitor. The path points and other process signals can be predefined or changed by the process controller at runtime.

Parameterizable switching signals control when the robot executes which motion. The speed and acceleration at which the robot travels along these path segments is defined by preconfigurable sets of motion parameters. Examples of sets of motion parameters include rapid speed, creep speed, or gripping motions.

Types of controls

In addition to program mode, the software module can move the joint axes and Cartesian axes in jog mode. The lower-level axes also offer the full function of the respective single axis, such as referencing.

Error handling

Error handling with error message transmission to the process controller is integrated into the system. Several diagnostics tools are available for a more detailed error analysis and startup of the system, e.g. the 3D simulation integrated into the RobotMonitor.

3.2 Functions

Overview of functions:

- Control and coordination of the drives of a robot application
- Motion control by means of path interpolation
- Support of mechanical components with up to 2 Cartesian degrees of freedom (translation along the X, Y, Z axis, or rotation around these axes)
- Support of mechanical components with a default of up to 2 joint axes (including gantry robots, delta robots, and SCARA)
- Handling objects at rest
- Operating modes: Manual" and "Automatic" (program)
- Jog mode of the joint axes/Cartesian axes
- Different types of program sequences in program mode: automatic, single step for each set command or motion command
- Operation via RobotMonitor or IEC Editor
- Simulation of processes for diagnosing problems early, without a real machine
- Reproducible path fidelity even after disturbances by repositioning to the path
- Configuration of wait points at any point during the robot program
- Scaling the speed using an override input value
- Tool transformation
- Integrated, automatic 3D simulation of the robot and its paths in the RobotMonitor
- Robot programming using the SEW Robot Language (SRL):
  - Teach-in function
  - 20 memory locations for programs with hundreds of motion commands each
  - Linear interpolation with jerk-limited blending
  - Use of explicit coordinates or variable poses
  - All variables (BOOLEAN, REAL, POSE) can be written and read in IEC
  - Use of control flow elements: IF, WHILE
  - CallFunctions for synchronized and consistent execution of IEC code

It is also possible to extend the range of functions of the software modules by add-ons. For further information, refer to chapter "Add-ons" (→ 🗎 15).

3.3 Add-ons

INFORMATION

The add-ons are activated in the configuration menu "Basic settings" of the software module in the "Functions used" section. After activation, an additional configuration menu is displayed in the configuration. Please note that a "license" (→ 🗎 25) might be required to use the add-on.

3.3.1 MOVIKIT® Robotics add-on MediumModels

Support of kinematic models with 3 or 4 Cartesian degrees of freedom of the types gantry robot, delta robot, tripods, SCARA, and MIXED.

The add-on contains the 3D models of the kinematic models in the RobotMonitor and enables convenient configuration of the model in MOVISUITE®.

3.3.2 MOVIKIT® Robotics add-on Touchprobe

When switching a sensor or changing the state of a BOOLEAN variable in the robot program, the Cartesian actual position of the robot on the robot's path is determined. A defined action can then be performed.

A possible action to be performed is, for example, sensor-based positioning. A specific remaining distance is traveled starting from the measured path point on the programmed path of the robot. The remaining distance is specified in a certain direction, for example when palletizing along the z-coordinate.

For further information, refer to the chapters "Basics" (→ 🗎 22), "Functional description" (→ 🗎 86), and "Robot programming" (→ 🗎 127).

4 Basics

4.1 Kinematic models

The kinematic models differ in the following areas:

- Type and layout of the joints: basic types (e.g. SCARA)
- Type and number of the joint axes: J1 to J6, linear/rotary (e.g. LRRR)
- Differences in detail: layout of the drives (e.g. M10)

The identifiers of the kinematic models are structured accordingly.

4.1.1 Basic types

CARTESIAN GANTRY

CARTESIAN GANTRY is a kinematic model in which 2 or 3 linear axes are at a 90-degree angle to each other, thereby setting up a Cartesian work envelope.

ROLLER GANTRY

ROLLER GANTRY is a kinematic model in which 2 degrees of translational freedom of 2 usually stationary drives are controlled by a circulating toothed belt. Additional degrees of freedom may be upstream or downstream of this assembly.

SCARA

SCARA is the English-language acronym for "Selective Compliance Assembly Robot Arm". SCARA is a kinematic chain in which 2 rotary axes are arranged in parallel to each other. These axes are called shoulder joint and elbow joint, analogous to the human arm.

DELTA

DELTA is a kinematic model in which 2 kinematic subchains are connected in parallel between the kinematic base and the tool flange using a triangular layout.

TRIPOD

TRIPOD is a kinematic model that can be characterized as a tripod. It consists of 3 kinematic subchains between the kinematic base and the tool flange, arranged in parallel to each other.

QUADROPOD

QUADROPOD is a kinematic model that consists of 2 kinematic subchains between the kinematic base and the tool flange, arranged in parallel to each other.

HEXAPOD

HEXAPOD is a kinematic model that consists of 6 kinematic subchains between the kinematic base and the tool flange, arranged in parallel to each other.

ARTICULATED

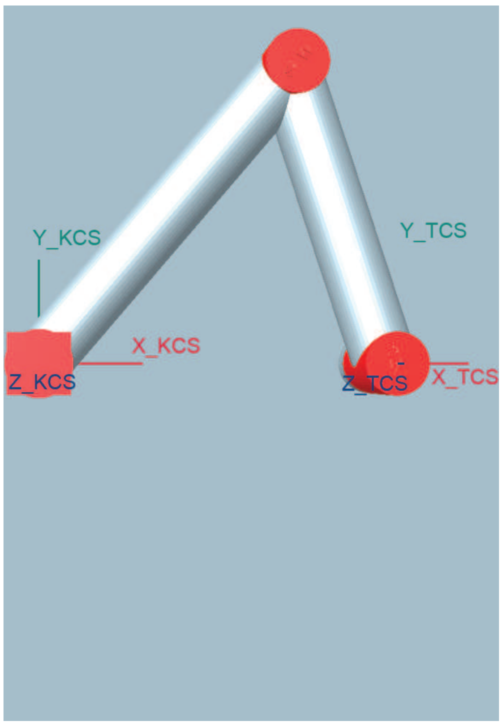
ARTICULATED is the kinematic model of an articulated arm robot. An articulated arm robot has 5 to 6 degrees of freedom and includes multiple rotary joints.

MIXED

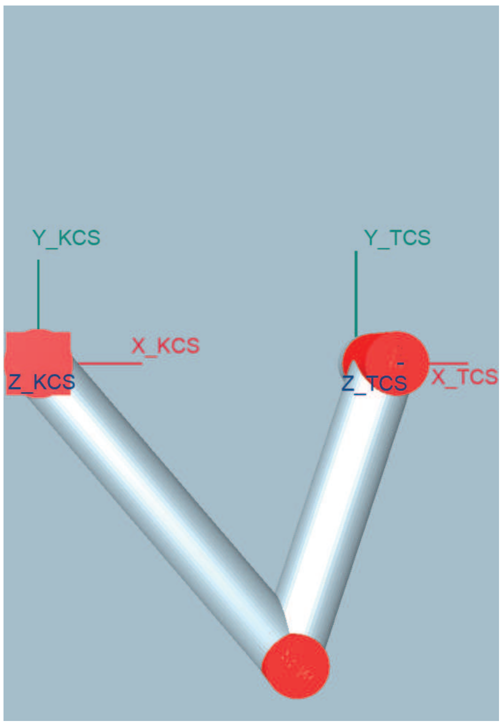
MIXED describes those kinematic models that do not clearly show the characteristics of other kinematic models. In particular, these kinematic models do not correspond to the kinematic models CARTESIAN CANTRY, ROLLER GANTRY, SCARA, DELTA, TRIPOD, QUADROPOD, HEXAPOD, and ARTICULATED.

4.1.2 Constellation

For certain kinematics, the Cartesian pose (ISO 8373: "Combination of position and orientation in space") is not sufficient to clearly describe the axis position. A simple example of this would be the SCARA kinematics, where the same Cartesian pose can be achieved by 2 different axis positions (see figures below). The constellation describes the position of the robot during the pose. A constellation means assigning numbers to the possible axis positions.



13935221643



13935224075

INFORMATION

The constellation is maintained during movement by means of path interpolation, and while the Cartesian axes are being jogged.

Jogging the joint axes allows for the constellation to be changed.

4.1.3 Transformation between axes and joint axes

Each kinematic model offers the option of configuring a transformation between the axes (linear or rotary axes actuated by a drive) and the actuated joint axes of the kinematic model.

This allows for additional mechanical transmissions (converting a linear motion to a rotary motion) and couplings between the axes (deflecting a rotary motion by means of a parallelogram or a belt) to be modeled, for example.

Contact SEW-EURODRIVE if you wish to implement such a transformation.

4.2 Motion control

4.2.1 Interpolation

The robot motion control creates a profile of the motor setpoint positions for all operating states of the kinematic model. The motion path is generated by means of path interpolation. The target is given in Cartesian coordinates.

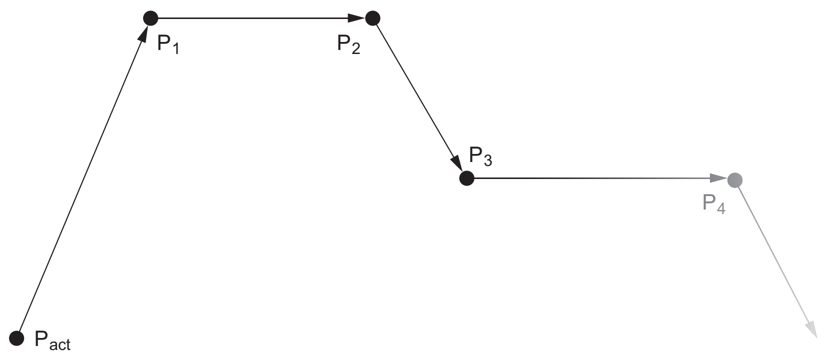
Path interpolation

Path interpolation generates a geometrically defined path of the translational and rotary degrees of freedom.

The robot tool moves along a path, consisting of straight lines and blending curves, defined within the work envelope. For the transitions between 2 straight segments, you can define a rounding of the path (blending).

4.2.2 Path

The robot program specifies the path of the tool to be used by the robot. The path is set up by a series of path points. For example, a path in which the system stops at each path point may look like the one below using 4 path points.



13929520267

P_{act}	Path point, actual
P_{1-4}	Path points 1 to 4

The segment of motion from one path point to the next is called the path segment.

INFORMATION

Whether the system moves exactly along the straight line between the individual path points or not depends on the type of interpolation, the wait points, and the blending.

29180252/EN – 03/2020

4.2.3 Blending

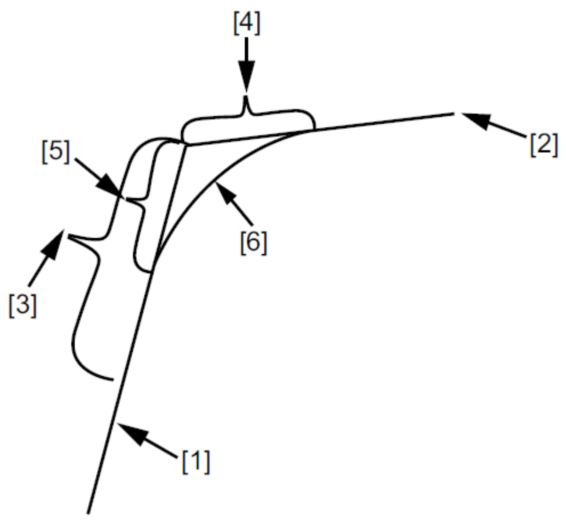
Blending is the rounding of the corners of a motion path. Blending ensures a continuous transition of the path and the path speed toward the next path segment. Blending protects the mechanical components and reduces cycle time. Without blending, the kinematic model would stop at the *target pose* and then start its motion toward the next target point.

Blending starts as soon as the tool is close enough to the current target pose. For each path segment, the robot program defines the distance (blending distance) from one path segment to the next. However, this blending distance is limited to a percentage of the segment length on which blending is performed. The default value for this limiting percentage is 50%. However, the value can be increased to up to 99%.

This means the actual blending distance (*blending distance*$_{\text{effective}}$*)* results from the smaller value (minimum) of the two variables, limited by the remaining distance if the new path segment is taken over late (e.g. because of a waiting point).

Blending distance$_{\text{effective}}$ =

min. (*blending distance*$_{\text{target}}$, *segment length*, *limiting percentage, remaining segment*)

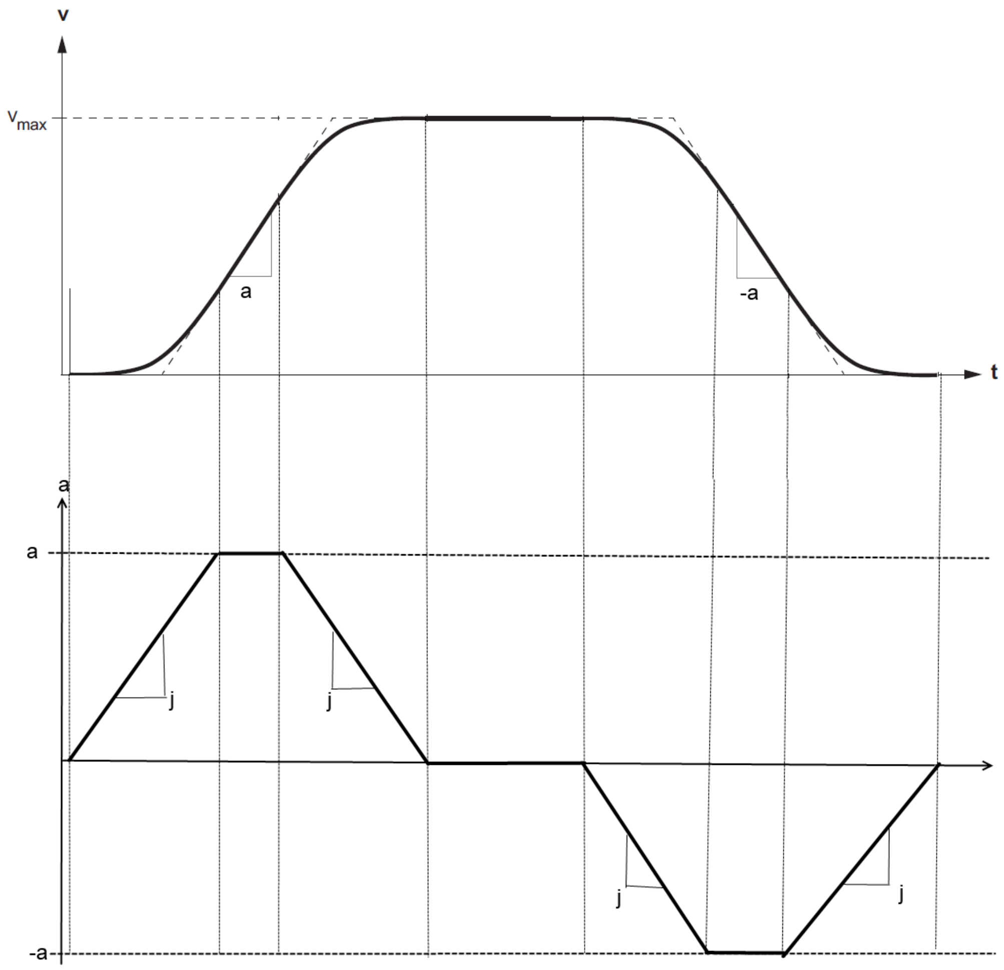


31458163211

[1] Linear segment to the first target position
[2] Target position of the second linear segment
[3] Set blending distance$_{\text{TARGET}}$
[4] Segment length * limitation percentage
[5] Blending distance$_{\text{effective}}$
[6] Resulting symmetrical blending arc

4.2.4 Motion profiles

Any motion along an axis or around an axis (joint axis, Cartesian axis, or path axis) requires a defined motion profile. The following figure shows which sizes you can use to define the motion profile when using this software module.



9007222991719691

v_{max}	Maximum permitted speed for the motion task
a	Maximum permitted acceleration for the motion task
y	Jerk (rate of change of acceleration)

v_{max} is not reached in the following cases (no constant travel):

- a is extremely low compared to v_{max}.
- j is extremely low compared to a.
- The distance to be covered is extremely short.

In these cases, the travel diagram above is simplified to a jerk-limited triangular profile. Accordingly, it is also possible that acceleration or deceleration will not be achieved (no section with constant acceleration).

INFORMATION

– The jerk limitation increases the time required for a given distance.

– In addition to these bases, there are other factors and underlying conditions that may influence the actual travel diagram. The travel diagram above is only used to explain the principle of the relevant motion parameters.

4.2.5 Sets of motion parameters

To move the robot, it is necessary to configure motion parameter sets (e.g. specifications for rapid speed and creep speed). It is therefore necessary to define the Cartesian degrees of freedom for each axis, or the specific motion parameters for the path segments (speed, acceleration, deceleration, jerk).

Example of configuring motion parameters

If a speed of 1 m/s is to be traveled and this is to be reached in 0.2 s, an acceleration of 5 m/s^2 must be set (a=v/t; assumption: without jerk limitation). However, if an acceleration of 5 m/s^2 is to be reached in 0.1 s, a jerk of 50 m/s^2 must be set. In the case with jerk limitation in this example, the desired speed is then reached in less than 0.4 s (0.2 s + 2 * 0.1 s = 0.4 s).

For setting the parameters, see "Setting motion parameters" (→ 🖹 161).

4.2.6 Scaling using override

In addition to the motion parameters defined in the sets of motion parameters, the input value of *Override* is also applied. Using *override*, the programmed speed can be scaled as a percentage between 0% and 100%. The *Override* only affects speed, but not acceleration, deceleration, and jerk. Therefore, the time required for the complete robot program is not scaled proportionally.

4.2.7 Default units

The following units are used by default:

Parameter	Unit
Track measure (translation)	mm
Translational speed	mm/s
Translational acceleration	mm/s^2
Translational jerk	mm/s^3
Angle dimension (rotation)	Degree
Rotational speed	degree/s
Rotational acceleration	$Degrees/s^2$
Rotational jerk	$Degrees/s^3$
Blending distance	mm

29180252/EN – 03/2020

4.2.8 Emergency stop

Emergency stop brakes the kinematic model to standstill using the ramps set for the current operating mode. An emergency stop occurs in the following situations:

- Fault status of a software module
- When canceling the enable signal: In this case, the enable state of the individual axis is automatically canceled after standstill is reached; as a result, emergency stop is activated by means of the brakes or the inverter controllers.

Emergency stop is not used if the inverter reports a fault or if controller inhibit is set. In this case, each inverter uses the emergency stop ramp configured in the inverter to brake and decelerate its drive.

4.2.9 Path events

Path events are events that are triggered at a defined position on the path of the robot or at a defined time before or after reaching this position. For example, the task of applying glue can be started and stopped at certain positions, or the vacuum of a suction gripper can be switched on a certain time before reaching the end of the path. Path events are triggered at the parameterized position or the parameterized time before or after independently of the execution of the robot program, i.e. of the progress of the program pointer.

For further information, refer to the chapters "Functional description" (→ 🗎 81) and "Robot programming" (→ 🗎 125).

4.2.10 Touchprobe

The "touchprobe" function allows for triggering an instruction during a program by switching a sensor or by changing the state of a BOOLEAN variable. When the event is triggered, the Cartesian actual position of the robot on the track is determined. As an instruction, the measured position can be stored in a POSE variable, the state of a BOOLEAN variable can be changed, a function can be called, or sensor-based positioning can be performed. In sensor-based positioning, the programmed path is continued by a certain length in the specified direction starting from the measured path point. If the remaining distance is greater than the programmed path, the last segment will be extended. The sensor can be connected directly to the MOVI-C® CONTROLLER (e.g. via a digital input) and cause the status change of a BOOLEAN variable. In this case, the actual positions of the connected drives are used for the measurement. If the requirements on accuracy are very high, the sensor is connected to all inverters belonging to the robot. In this case the touchprobe positions of the inverters will be used for measuring the path point. The measured positions of the drives are transformed into the Cartesian position.

For further information, refer to the chapters "Functional description" (→ 🗎 86) and "Robot programming" (→ 🗎 127).

4.3 Coordinate systems

The pose of the robot can be specified via different coordinate systems. Depending on the position of the coordinate system, the coordinates of a pose are different for each coordinate system. MOVIKIT® Robotics has the following coordinate systems:

- Base

  A coordinate system that is generally fixed rigidly in the base of the kinematic model. This system is used as a reference coordinate system for direct kinematic transformation (transformation of the joint axis values into Cartesian values of the tool).

- Joint

  A coordinate system in which each coordinate corresponds to one joint axis of the kinematic model.

4.4 Communication and process data exchange

The software module's hardware topology includes at least the following system components that communicate with each other:

- MOVI-C® CONTROLLER with the software module
- Application inverter

The MOVI-C® CONTROLLER with the software module uses the EtherCAT®-based system bus to communicate with the application inverters. To ensure unique addressing of the messages and thus of the motion tasks for each individual drive, each application inverter requires a different address on the system bus.

In addition, there may be an optional Programmable Logic Controller (PLC) that is on a higher topological level than the MOVI-C® CONTROLLER. The MOVI-C® CONTROLLER communicates with the PLC via the fieldbus. Process data is exchanged for this purpose. Both systems must know how to interpret the data.

5 Project planning information

5.1 Requirement

Correct project planning and proper installation of the devices are required for successful startup and operation.

For detailed project planning information, refer to the documentation of the respective devices.

5.2 Hardware

The following hardware is required:

- MOVI-C® CONTROLLER – "Observe" (→ 🗎 28) cycle time configuration
  - UHX25A/UHX45A with a minimum cycle time of 5 ms
  - UHX65A/UHX85A with a minimum cycle time of 1 ms
- MOVIDRIVE® as an interpolating device

  (Inverter must support the "Interpolated speed control" operating mode)

5.3 Software

The following software is required:

- MOVISUITE® engineering software

  (includes MOVIRUN® flexible, MOVIKIT® MultiMotion, and the IEC Editor)
- MOVISUITE® RobotMonitor

For more detailed information on the hardware requirements of the individual software components, see the documentation for the respective software.

5.4 Licensing

The following licenses are available and are required:

- MOVIRUN® flexible

  License for the software platform MOVIRUN® flexible. Required once per MOVI-C® CONTROLLER.
- MOVIKIT® Robotics

  License for the basic module MOVIKIT® Robotics. Required once per robot.
- MOVIKIT® Robotics add-on MediumModels

  License for the "MediumModels" add-on. Required per robot with 3 or 4 joint axes.
- MOVIKIT® Robotics add-on Touchprobe

  License for the "Touchprobe" add-on. Required per robot with touchprobe measurement or touchprobe positioning.
- MOVIKIT® Robotics add-on Circle (in preparation)

  License for the "Circle" add-on. Required per robot with circular interpolation.
- MOVIKIT® Robotics add-on ConveyorTracking (in preparation)

  License for the "ConveyorTracking" add-on. Required per robot with interpolation in a USER coordinate system.

For further information on licensing, refer to the document "MOVI-C® Software Components". You can download the document from the SEW-EURODRIVE website (www.sew-eurodrive.com).

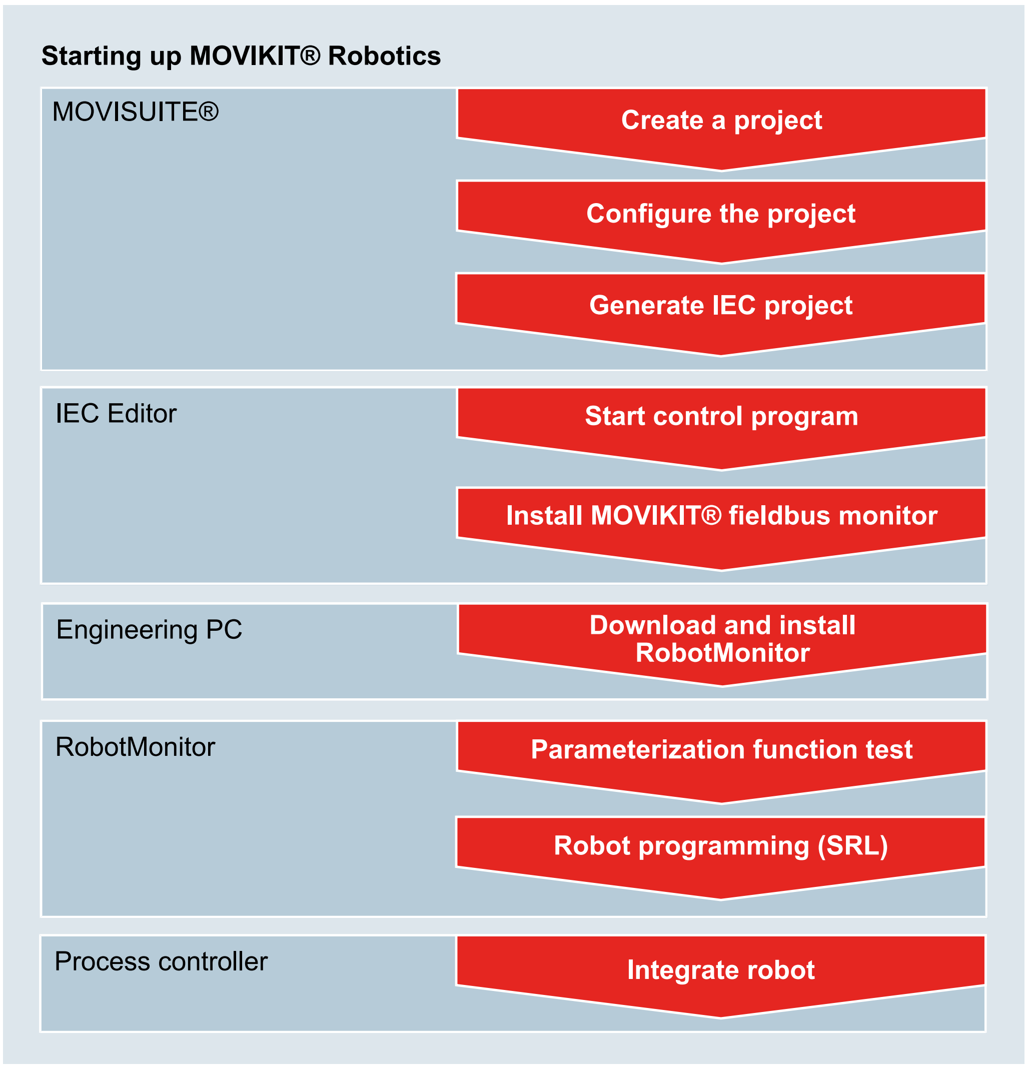
6 Startup

6.1 Requirements

- Check the installation of the inverters and, if installed, also check the encoder connection.
- Observe the installation notes in the documentation of the respective device and software components.
- The devices to be started up are displayed in MOVISUITE®.

6.2 Startup procedure

The schematic diagram below shows the startup procedure:



9007222536109195

The startup steps specific to these software modules are explained in detail in the following chapters of this manual. For startup, also observe the documentation of all the other components in use.

6.3 Configuring a project

INFORMATION

For detailed information on how to operate the MOVISUITE® engineering software, refer to the corresponding documentation.

✓ A MOVISUITE® project has been created and is open.

1. Add required device nodes, software nodes (MOVI-C® SoftwareNode) and software modules to the project.

   ⇨ See "Example project".

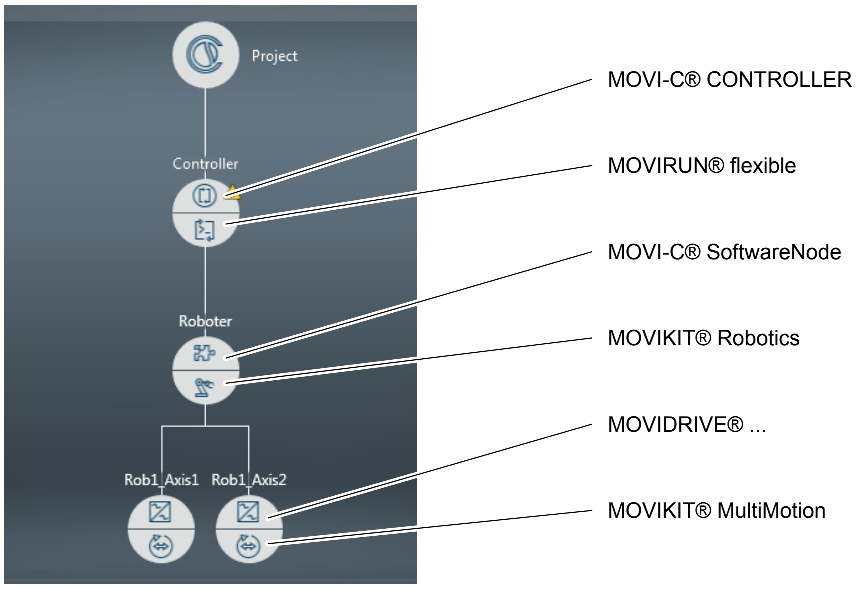
2. Configure the added devices or software modules. If available, observe the specific notes in the following chapters that apply to MOVIKIT® Robotics. For detailed information on the configuration of devices or other software modules, refer to the respective documentation.

6.3.1 Example project

The following figure shows an example project:

INFORMATION

Axes subordinate to MOVIKIT® Robotics require MOVIKIT® MultiMotion (and possibly the MOVIKIT® MultiAxisController)



31222970507

29180252/EN – 03/2020

6.3.2 Configuring the MOVI-C® CONTROLLER

INFORMATION

For detailed information on how to configure the MOVI-C® CONTROLLER, refer to the corresponding documentation.

The default setting for the cycle time of the MOVI-C® CONTROLLER is 1 ms. Depending on the performance class of the MOVI-C® CONTROLLER, the computing resources are not sufficient for calculating several instances of software modules on a MOVI-C® CONTROLLER within 1 ms. In this case, the cycle time must be increased. See chapter "Setting the cycle time" (→ 🗎 28)

INFORMATION

When increasing the cycle time, take into account that the path of the axes becomes less accurate as the cycle time increases.

At least the following cycle time is required for an instance:

- MOVI-C® CONTROLLER standard/advanced: 5 ms
- MOVI-C® CONTROLLER progressive/power: 1 ms

Each additional instance (function in preparation) requires a corresponding increase of the cycle time.

Setting the cycle time

The cycle time is set in the following steps:

Setting the "Controller setpoint cycle" on the axes

In MOVISUITE®, perform the following steps for all lower-level axes:

1. Open the configuration for the axis.
2. In the "Functions" section, open the "Setpoints" configuration menu and its submenu "Basic settings".
3. In the "Basic settings" section, set the value in the "Controller setpoint cycle" edit box to the required value.

Setting the TaskHighPrio cycle time for the MOVI-C® CONTROLLER

In MOVISUITE®, perform the following steps for the MOVI-C® CONTROLLER:

4. Open the configuration of the MOVI-C® CONTROLLER.
5. In the "MOVIRUN® flexible" section, open the "Task system" configuration menu.
6. Under "Task system", enter the required value in the "Cycle time of the HighPrio Task" edit box.
7. Click on the blue arrow in the "Task system" area of the "Sync offset EtherCAT" edit box to accept the suggested value.

Setting up fieldbus connection

Perform the following steps to allow the MOVI-C® CONTROLLER access to the field-bus via IEC function blocks. This setting is required for direct fieldbus connection of software modules.

✓ A MOVISUITE® project has been created and is open.

✓ The MOVISUITE® project includes a MOVI-C® CONTROLLER.

1. In the function view of MOVISUITE®, click on the node of the MOVI-C® CONTROLLER.
   ⇨ The configuration menu of the MOVI-C® CONTROLLER opens.
2. In the "MOVIRUN® flexible" configuration menu, open the submenu "Fieldbus".
3. In the "Fieldbus card" section, select the fieldbus protocol in use.
4. In the "Fieldbus connection via IEC function blocks" section, set the value of the field "Activate fieldbus connection" to "Yes".

6.3.3 Adding MOVIKIT® Robotics

INFORMATION

For detailed information on how to operate the MOVISUITE® engineering software, refer to the corresponding documentation.

✓ A MOVISUITE® project has been created and is open.

1. Click on the empty software module section of the required node.
   ⇨ The catalog section opens and displays the available software modules.
2. In the catalog section, click on MOVIKIT® Robotics.
   ⇨ A context menu opens.
3. Select the version from the respective drop-down list in the context menu and confirm your selection with [Apply].

⇨ The MOVIKIT® Robotics is assigned to the node, the configuration is created, and the basic settings are performed.

6.3.4 Configuring subordinate nodes

⚠ WARNING

Unforeseen movements of the robot by configuring a jerk limitation on the inverters.

Physical injury and death.

- Make sure that no jerk limitation is configured on the inverters.

INFORMATION

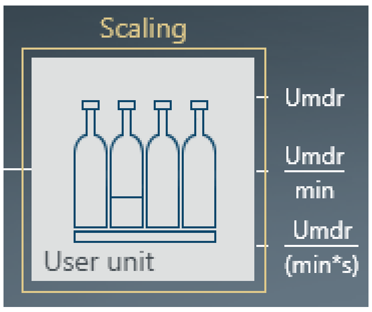
For detailed information on how to configure the individual nodes, refer to the corresponding documentation.

Observe the following notes for nodes subordinate to the software module:

- The following axis group members may be used under the robot:
  - MOVIDRIVE® with MOVIKIT® MultiMotion (recommended) or MOVIKIT® MultiMotion Camming
  - MOVI-C® virtual axis with MOVIKIT® MultiMotion or MOVIKIT® MulitMotion Camming
  - MOVI-C® SoftwareNode with MOVIKIT® MultiAxisController
- The drives are braked individually via the emergency stop in the inverter in the event of an axis-by-axis emergency stop of MOVIKIT® Robotics. No central profile generator is used for braking. This can cause the lower-level MOVIKIT® MultiAxis-Controller to jam the mechanics in the "Torque priority" operating mode and unexpected skew in the "Skew priority" operating mode.
- The software and hardware limit switches must be set before optimizing the drive train.

For the correct operation of the software module, at least the following startup steps are necessary for the lower-level nodes. Refer to the respective documentation for more information.

1. Configure the drive train and define the user unit. See chapter "Default units" (→ 🗎 21). The "user unit" component must be used in the drive train to ensure that the decimal places are set correctly.



31637091083

2. Depending on the mechanical installation of the respective drive, a direction of rotation reversal must be set in the "Controller" parameter group. The positive direction of movement can be seen on the display in the 3D simulation. The procedure is described in chapter "Referencing axes and performing function test" (→ 🗎 48).

29180252/EN – 03/2020

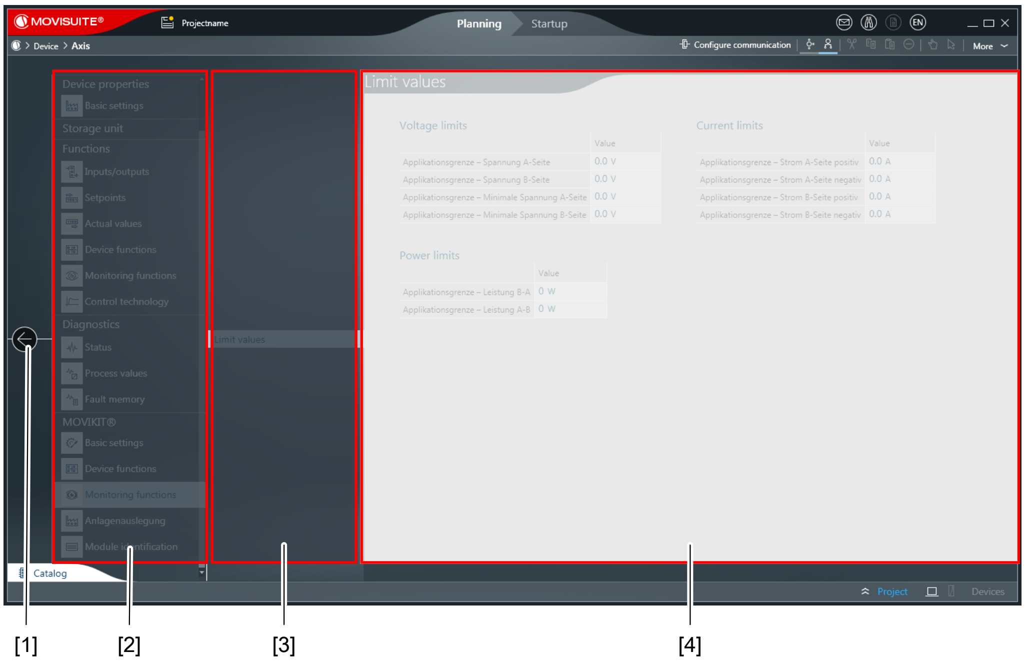
3. Insert the MOVIKIT® MultiMotion software module or MOVIKIT® MultiAxisController and use the basic setting for the decimal places.
4. Activate the existing hardware limit switches (optional).
5. Activate and configure the software limit switches (the software limit switches should be selected in such a way that they are located in the work envelope in front of the hardware limit switches in a distance that corresponds to the braking distance).
6. Configure the reference travel.
7. Configure the emergency stop ramps. Set the jerk to "0".
8. Configure the torque limit.
9. Optimize the "controller".

6.3.5 Configuring MOVIKIT® Robotics

INFORMATION

For detailed information on how to operate the MOVISUITE® engineering software, refer to the corresponding documentation.

1. In MOVISUITE®, click on MOVIKIT® Robotics.
   ⇨ The configuration menus of the software module are displayed. The configuration menus are explained in the following subchapters.



Limit values

Voltage limits

	Value
Applikationsgrenze – Spannung A-Seite	0.0 V
Applikationsgrenze – Spannung B-Seite	0.0 V
Applikationsgrenze – Minimale Spannung A-Seite	0.0 V
Applikationsgrenze – Minimale Spannung B-Seite	0.0 V

Current limits

	Value
Applikationsgrenze – Strom A-Seite positiv	0.0 A
Applikationsgrenze – Strom A-Seite negativ	0.0 A
Applikationsgrenze – Strom B-Seite positiv	0.0 A
Applikationsgrenze – Strom B-Seite negativ	0.0 A

Power limits

	Value
Applikationsgrenze – Leistung B-A	0 W
Applikationsgrenze – Leistung A-B	0 W

9007228165413771

[1] Button to return to the project overview
[2] Main menu of the software module configuration (MOVIKIT® section)
[3] Submenus of the configuration
[4] Setting fields of the respective submenu

2. Configure the software module using the respective setting fields.
3. Click button [1] after having completed the configuration.
   ⇨ The project overview is displayed.

INFORMATION

For the changes made to the configuration to take effect, you have to update the configuration data. To do so, click [Update configuration data] in the respective notification at the node or in the context menu of the MOVI-C® CONTROLLER. The MOVI-C® CONTROLLER is stopped and restarted for updating the configuration data.

Kinematic model

Model selection

Parameter name	Description
Kinematic model	
Model	Selection of the kinematic model

INFORMATION

Depending on the kinematic model selected, additional submenus with specific configuration options are displayed. For detailed information on how to configure the kinematic models and on the parameters they contain, refer to chapter "Kinematic models" (→ 🗎 60).

Path emergency dynamics

⚠ WARNING

Uncontrolled movements of the robot due to jerk setting on the inverter.

Damage to property and death.

- Deactivate the jerk for emergency stop of the inverters (set to 0).

INFORMATION

These parameters are also the upper limits for the deceleration and jerk of the robot movement in jog and program mode.

Make sure the motion parameters are defined for an emergency stop on the path. A distinction is not made between Cartesian degrees of freedom, but between translation and rotation. See also "Sets of motion parameters" (→ 🗎 21) and "Setting motion parameters" (→ 🗎 161).

Parameter name	Description
Path emergency dynamics	
Trans EStop Dec	Rate of deceleration with which the translation brakes on the path during an emergency stop.
Trans EStop Jerk	Jerk with which the translational deceleration is generated during an emergency stop.
Rot EStop Dec	Rate of deceleration with which the rotation brakes on the path during an emergency stop.
Rot EStop Jerk	Jerk with which the rotary deceleration is generated during an emergency stop.

The emergency stop ramps must be set in such a way that they...

- Can be performed at any time by the robot's inverters without them reporting an error (lag error, speed monitoring, etc.).
- Do not damage the mechanical components.
- Do not cause unwanted vibrations and noise during braking.
- Adhere to the maximum permitted braking distance.

Cartesian limits

Parameter name	Description
Cartesian SWLS	
Cartesian software limit switch The Cartesian software limit switches are always in operation and limit the tool coordinates in the base coordinate system for the translational degrees of freedom X, Y, and Z, as well as rotation around these axes. Not the joint axes of the kinematic model are limited here, but the 3D space coordinates of the tool. When the robot leaves this area, the software module stops with an emergency stop when the joint axes are jogged. The robot stops at this limit when the Cartesian axes are jogged as well as in program mode.	

Transformations

Parameter name	Description
Tool transformation	
Coordinate transformation from the flange coordinate system to the tool coordinate system The flange coordinate system concerned is the Cartesian coordinate system that is derived from the base coordinate system by means of the direct kinematic transformation. Using this transformation, you can take into account offsets and twists or rotations caused by the tool of the robot, unless these have already been taken into account in the kinematic model. This transformation cannot be changed during operation.	

Axis-joint transformations

Couplings table

Configure coupling factors between individual axes (axis) and joint axes (joint). The coupling factor indicates how much impact a single axis has on a joint axis that is not directly assigned to it, which means that does not have the same index. If there is no coupling, you can set the number of couplings to 0. If you set the couplings to greater than 0, a table will be displayed showing one row per coupling. For each coupling, you can set the input index, the output index, and the coupling factor.

Example

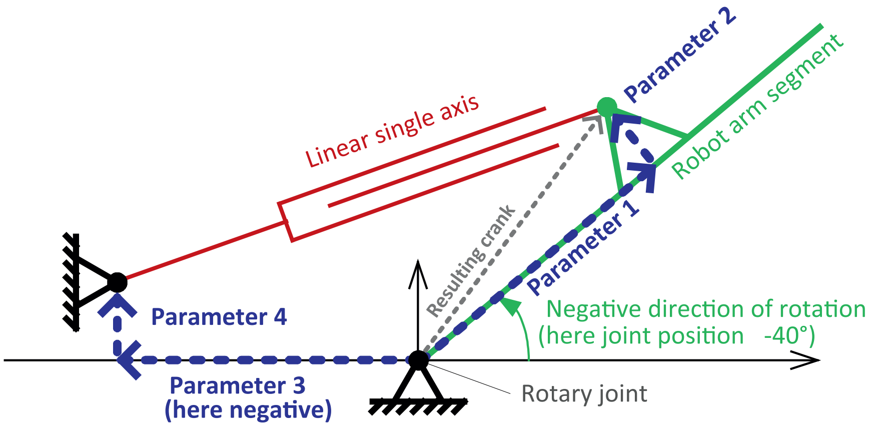
If the single axis number 4 rotates by 20°, only the joint axis number 4 rotates by 20°, if the coupling has an input index of 4, an output index of 5, and a coupling factor of 0. If the coupling factor is 1, however, both joint axis number 4 and joint axis number 5 rotate by 20°. Joint axis 5 is additionally rotated by the value of single axis 4.

Parameter	Description
Couplings	
Number of couplings	Number of couplings If this is set to greater than 0, the coupling factor table with its factor, input index, and output index columns is displayed.
Input index	Single axis from which the coupling is to take the position value.
Output index	Joint axis from which the coupling is to add the position value of the single axis (weighted by this factor).
Factor	Factor with which the position value of the single axis is weighted before the position value is then added to the joint axis.

Cranks table

Cranks are levers that rotate around an axis of rotation. In robots, cranks are used to rotate the lever around the axis of rotation (the axis of the robot joint) with a single linear axis.

If the number of cranks is set to 0, then a linear joint axis with index j is assigned to a linear single axis with index j, and a rotary joint axis with index k is assigned to a rotary single axis with index k. If the number of cranks is set to greater than 0, a table with a row per crank is displayed. The index and 4 parameters can be set for each crank.



9007228304930315

Example: In the example, parameters 1, 2, and 4 are positive, parameter 3 is negative. As parameter 2 is positive, the linear single axis moves above the rotary joint axis. The robot arm has the joint axis value 0. For example, if the robot arm is pointing vertically upwards, the robot arm has joint axis value -90°. This means the linear single axis would have to be shortened according to the figure.

Parameter	Description
Cranks	
Number of cranks	Number of cranks When set to greater than 0, the crank parameter table is displayed with the columns "index" and parameters 1 to 4.
Index	Index of single axis and joint axis between which the crank operates
Parameter 1	Offset from the rotary joint axis along the robot arm that is turned around the rotary joint axis to the point of connection between linear single axis and robot arm (parameter 1 > 0).
Parameter 2	Additional offset vertically to the robot arm and at the level of movement of the robot arm to the point of connection between linear single axis and robot arm. The linear single axis moves on one side of the rotary joint axis or on its other side depending on the sign of parameter 2.
Parameter 3	Offset from the rotary joint axis along the axis of the robot arm with joint axis value 0 to the point of connection between base and linear single axis. (Parameter 3 < 0)
Parameter 4	Additional offset horizontally to the axis of parameter 3 and at the level of movement of the robot arm to the point of connection between base and linear single axis.

Fieldbus interface

INFORMATION

The current version of the software module does not support referencing and unreferenced jogging of the axes via the fieldbus. Robots with absolute encoders function without problems after they have been referenced once using manual mode in MOVISUITE®. For further questions and an application solution for referencing and unreferenced jog mode via fieldbus, contact SEW-EURODRIVE.

General settings

Parameter name	Description
Fieldbus configuration	
Activating the fieldbus connection	Activating the fieldbus connection. For further information, refer to chapter "Fieldbus interface" (→ 🗎 143).
Start address	Defining the start address of the software module in the fieldbus interface of the MOVI-C® CONTROLLER
Process data length	Process data length automatically calculated based on the configuration
Decimal places via fieldbus	
Table for setting the decimal places for position, speed, acceleration, and jerk for translation and rotation via the fieldbus. A higher accuracy is achieved because the pose values and speed values can only be transmitted as integers via the fieldbus. These settings apply to all variables in the fieldbus interface of the selected type. If the decimal places in the software module are set greater than 0, the input values in the MOVI-C® CONTROLLER are divided by a factor of $10^{NumberOfDecPlaces}$ before they are processed as floating-point numbers. On the other hand, the output values in the MOVI-C® CONTROLLER are multiplied by this factor before they are transferred to the higher-level controller. If the decimal places are set to 2, and a MOVI-C® CONTROLLER input word contains the value 1, then the controller is operated with the value 0.01. If the decimal places are set to 2, and a MOVI-C® CONTROLLER output word contains the value 1, this value must be interpreted as 0.01 in the higher-level controller.	
Fieldbus profile template	
Fieldbus profile	Selection of the fieldbus profile: • Standard profile for positioning • Flexible, configurable profile For further information, refer to chapter "Fieldbus profiles" (→ 🗎 143).

Further setting options are available depending on the selected fieldbus profile.

Fieldbus profile: Standard profile for positioning

Parameter name	Description
Fieldbus configuration	

Parameter name	Description
Maximum number of path segments per telegram (target pose as well as wait and end signals)	The maximum number of path segments is transmitted in one telegram and can be run without exact stop between the path segments. By means of the end signals contained in the path segments, the number of path segments can be adjusted during operation from path to path between 1 and the maximum number.

Fieldbus profile: Flexible, configurable profile

Parameter name	Description
Consistency of a telegram	
Size of consistent blocks	Setting of the size of the consistent blocks used by the higher-level controller in the fieldbus interface area of the software module. The size must be kept the same for all consistent blocks in the fieldbus interface area of the software module on the higher-level controller (except for the last = "Number of consistent blocks" parameter)
Number of consistent blocks	Setting of the number of consistent blocks used by the higher-level controller in the fieldbus interface area of the software module
Expansion modules	
Activating the robot diagnostics module	Assignment of the expansion module "Robot Diagnostics Module" on the bus. It is recommended to activate the expansion module.
Size of a pose variable	
Table for setting the size of the individual coordinates of all poses in the fieldbus interface of the software module. You can choose between "No (0 bit)", "WORD (16 bits)" and "DWORD (32 bits)". The sum of the bits over all coordinates results in the pose size on the bus. By default, only X, Y, Z, A are transmitted as WORD. "No" is selected for B and C. This also applies to the "standard profile for positioning". Only if the total pose size changes (e.g. by adding B as a WORD), other signals in the interface will be shifted (in the example, backwards by one WORD per pose).	

Standard path segments

INFORMATION

Only visible when selecting the fieldbus profile "Flexible, parameterizable profile".

Parameter group/description
Setpoints
Number of standard path segments per consistent block Setting for adding standard path segments to each consistent block. The maximum number depends on the size of the consistent block, the size of the pose, and the process data words already used by other signals. If several consistent blocks are to be used, the parameter "Number of consistent blocks" under "General setting" must be adjusted accordingly.

More poses

INFORMATION

Only visible when selecting the fieldbus profile "Flexible, parameterizable profile".

Parameter group/description
Number of variables on the bus
Setting for adding robot program variables of this type as setpoint or actual value to each consistent block. The maximum number depends on the size of the consistent block, the size of the variable, and the process data words already used by other signals. If several consistent blocks are to be used, the parameter "Number of consistent blocks" under "General setting" must be adjusted accordingly.
Offsets in the list of Boolean variables
Parameter for moving the offset of the setpoint or actual values in the robot variable list of this type. You can select a value between 1 and 100 but there must not be an overlap with variables used elsewhere (between the setpoint and actual values, the variables from the path segments - beginning with 1 - and variables that are not transmitted via the bus). It is recommended to leave enough space between the variables used elsewhere so that the robot program does not need to be adapted when adding more variables.

More REAL variables

INFORMATION

Only visible when selecting the fieldbus profile "Flexible, parameterizable profile".

Parameter group/description
Number of variables on the bus
Setting for adding robot program variables of this type as setpoint or actual value to each consistent block. The maximum number depends on the size of the consistent block, the size of the variable, and the process data words already used by other signals. If several consistent blocks are to be used, the parameter "Number of consistent blocks" under "General setting" must be adjusted accordingly.
Offsets in the list of REAL variables
Parameter for moving the offset of the setpoint or actual values in the robot variable list of this type. You can select a value between 1 and 100 but there must not be an overlap with variables used elsewhere (between the setpoint and actual values, the variables from the path segments - beginning with 1 - and variables that are not transmitted via the bus). It is recommended to leave enough space between the variables used elsewhere so that the robot program does not need to be adapted when adding more variables.
Setpoints in the list of REAL variables
Table for setting the size on the bus as well as the type of decimal places of the configured real variables. The number of decimal places is set under "General settings", see the "Decimal places via fieldbus" parameter.
Actual values in the list of REAL variables
Table for setting the size on the bus as well as the type of decimal places of the configured real variables. The number of decimal places is set under "General settings", see the "Decimal places via fieldbus" parameter.

More Boolean variables

INFORMATION

Only visible when selecting the fieldbus profile "Flexible, parameterizable profile".

Parameter group/description
Number of variables on the bus
Setting for adding robot program variables of this type as setpoint or actual value to each consistent block. The maximum number depends on the size of the consistent block, the size of the variable, and the process data words already used by other signals. If several consistent blocks are to be used, the parameter "Number of consistent blocks" under "General setting" must be adjusted accordingly.
Offsets in the list of Boolean variables
Parameter for moving the offset of the setpoint or actual values in the robot variable list of this type. You can select a value between 1 and 100 but there must not be an overlap with variables used elsewhere (between the setpoint and actual values, the variables from the path segments - beginning with 1 - and variables that are not transmitted via the bus). It is recommended to leave enough space between the variables used elsewhere so that the robot program does not need to be adapted when adding more variables.

Additional functions

Parameter name	Value
Touchprobe measurement + positioning	
Use touchprobe functions	Activation or deactivation of the touchprobe function (measurement and/or sensor-based positioning) When the function is activated, it can be selected in the RobotMonitor and the corresponding license is retrieved by the MOVI-C® CONTROLLER. For further information, refer to chapter "Touchprobe" (→ 🗎 86).
	Index: 50005.5
	IEC name: -

Module identification

Parameter group	Description
Module identification	Includes name and version for identifying the software module.

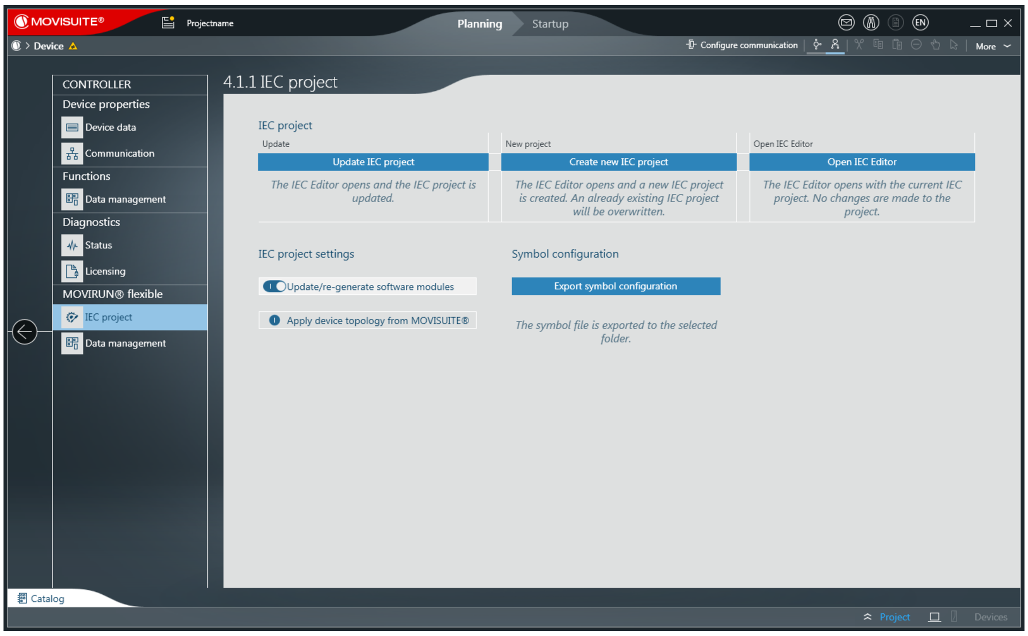
6.4 Generating an IEC project

Carry out the following steps to create an IEC project using automatic code generation and based on the configuration settings in MOVISUITE®.

✓ Configuration of the MOVISUITE® project has been completed.

1. In the function view of MOVISUITE®, click the software module section of the MOVI-C® CONTROLLER.

   ⇨ The "IEC project" menu opens.



4.1.1 IEC project

IEC project

Update	New project	Open IEC Editor
Update IEC project	Create new IEC project	Open IEC Editor
The IEC Editor opens and the IEC project is updated.	The IEC Editor opens and a new IEC project is created. An already existing IEC project will be overwritten.	The IEC Editor opens with the current IEC project. No changes are made to the project.

IEC project settings

Update/re-generate software modules

Apply device topology from MOVISUITE®

Symbol configuration

Export symbol configuration

The symbol file is exported to the selected folder.

27021618448637067

INFORMATION

If you have carried out the configuration in MOVISUITE® using the "Startup" mode and the message "Device cannot be reached" appears, proceed as follows:

- If the MOVI-C® CONTROLLER is not available via the network, switch over to "Planning" mode.
- If the MOVI-C® CONTROLLER is available via the network, carry out a network scan and connect the MOVI-C® CONTROLLER in the network view with the MOVI-C® CONTROLLER in the function view.

2. Click [Create new IEC project].

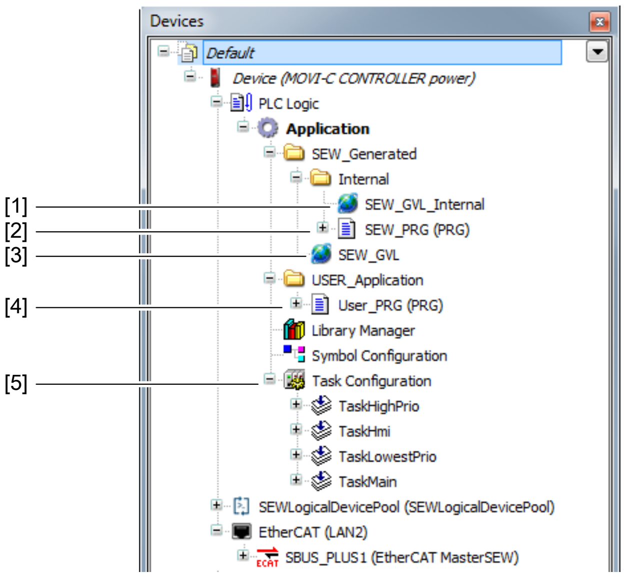
   ⇨ The IEC Editor opens and a new IEC project is created.

INFORMATION

If changes are made to the project structure, to inverter data sets, or to a software module configuration after the IEC project is generated for the first time, a notification symbol is displayed on the MOVI-C® CONTROLLER node. Click on the message icon for more information about the change, and to update the IEC project.

6.4.1 IEC project structure

The IEC project has the following basic structure:



18014423003085323

No.	Name	Description
[1]	SEW_GVL_Internal	The SEW_GVL_Internal global list of variables contains the instances that correspond to the software module used. These variables may not be written to from the user program. In addition, the structure contains an instance as a communication buffer for controlling or monitoring the software module by means of a monitor.
[2]	SEW_PRG	Program that contains all the important instance calls. Automatic code generation recreates this program in accordance with the configuration made in MOVISUITE® each time the IEC project is created, thereby overwriting the previous version. Therefore, you should not make any changes to this program.
[3]	SEW_GVL	The SEW_GVL global list of variables is the interface for accessing the software module features.
[4]	User_PRG	The user program is created once, initially, by automatic code generation. Since the program is not overwritten with each subsequent creation, this is the appropriate place for integrating user programs. The program is divided into five actions. These actions differ in the time at which they are called during the program sequence.
[5]	Task configuration	The list of tasks created in the project. Automatic code generation initially adds tasks that differ in how they are prioritized. The user can add additional programs to existing tasks or create new tasks. It is the responsibility of the user to design the capacity utilization of the tasks to enable the tasks to be processed within the required cycle time. Moving beyond the cyclical tasks, in particular, prevents setpoints for the interpolating axes being generated in time, which means that these axes cannot be operated properly.

6.5 Importing the MOVIKIT® fieldbus monitor

You have to import the MOVIKIT® fieldbus monitor to being able to monitor and control the fieldbus interface.

In the IEC Editor, open the menu [Tools] > [Scripting] > [Scripts] > [F] select the menu entry [Fieldbusmonitor.py]. For further information on how to use the MOVIKIT® fieldbus monitor, refer to the chapter "MOVIKIT® fieldbus monitor" (→ 🗎 156).

6.6 Starting the control program

✓ The IEC Editor and the newly created project are open.

✓ MOVI-C® CONTROLLER and engineering PC are connected.

1. Correct all errors in the library manager (see notification window).
2. Correct all errors occurring during precompilation (see notification window).
3. Click [Compile] in the [Create] menu to compile the project.
4. Correct all compilation errors (see notification window).
5. Click [Login] in the [Online] menu to log in to the controller.
6. Click [Start] in the [Debug] menu to start the controller.
7. In "Startup" mode of MOVISUITE®, click a free position in the function view to open the context menu and choose the [Update device addresses] menu item.

6.7 Installing the RobotMonitor

To operate the MOVIKIT® Robotics software module, installation of the RobotMonitor is required. For this purpose, carry out the following steps:

1. Go to our website (www.sew-eurodrive.de) and open the software download page (Online Support / Data & documents / Software)
2. Enter the term "RobotMonitor" in the search field and click [Find].
3. Select the entry "RobotMonitor" from the list of results and click [Download selection as a ZIP file].
4. Unzip the downloaded ZIP file.
5. Start the *.exe file from the folder with the unzipped files and follow the installation instructions.

6.7.1 System requirements

The following hardware is required:

- Processor: 1 GHz 32-bit (x86) or 64-bit processor
- RAM: 4 GB RAM (recommended)
- Hard disk: 40 MB free memory in total
- Graphic card: 2 GB graphics memory

The RobotMonitor is a 32-bit application.

You need one of the following operating systems:

- Windows 7 at least SP1 32-bit/64-bit

  or

  Windows 8.1 32-bit/64-bit

  or

  Windows 10 32-bit/64-bit

During operation, observe the following recommendations regarding the resolution:

- The best display is achieved with a resolution of 1280 x 800 px or higher
- Using large fonts and low screen resolutions could result in information not appearing on the display.

6.8 Starting the RobotMonitor

✓ The control program with MOVIKIT® Robotics has been started on the MOVI-C® CONTROLLER ("RUN").

✓ There is an Ethernet connection between the MOVI-C® CONTROLLER and the engineering PC.

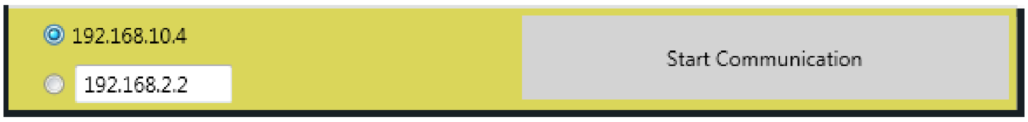
1. Open the application file (RobotMonitor.exe) of the robot monitor. For further information on installing the robot monitor, refer to chapter "Installing the RobotMonitor" (→ 🗎 45).

6.9 Establishing the connection to the controller

INFORMATION

If a connection cannot be established, refer to chapter "Communication not possible between RobotMonitor and MOVI-C® CONTROLLER" (→ 🗎 164) for information on troubleshooting.

1. Select the IP address: Standard IP address of the MOVI-C® CONTROLLER (192.168.10.4) or a user-specific IP address.



9007224167506187

2. Click the [Start Communication] button.
   ⇨ The connection has been established and signals the status "Comm. OK" at three positions in the robot monitor.

⇨ The RobotMonitor communicates successfully with the controller.

6.10 Managing users (optional)

The robot monitor is equipped with user management to manage the access rights of various users. User management allows, for example, to permit or inhibit specific functions, such as changing robot programs, for certain users. See chapter "User management" (→ 🗎 101).

6.11 Setting sets of motion parameters

Motion parameters must be set in advance both for jog mode and for path interpolation (e.g. for straight sections) of joint axes and Cartesian axes of the kinematic model.

For basic explanations on this topic, refer to the chapters "Motion profiles" (→ 🗎 20), "Sets of motion parameters" (→ 🗎 21), and "Setting motion parameters" (→ 🗎 161). The specified "Default units" (→ 🗎 21) apply.

Jog mode

Motion parameters for jogging joint axes can be found under "MotionSet.Kinematic.Joint". Motion parameters for jogging translatory and rotary Cartesian axes are available under "MotionSet.Translation" and "MotionSet.Rotation".

The default for jog mode is motion parameter set "8". The motion parameter set for jog mode can be set via the "user interface in the IEC program" (→ 🗎 131).

In the current software version, the ratio of jerk to acceleration or deceleration must be increased to increase the resulting jerk in jog mode. In future versions, the resulting jerk will correspond to the set value.

Automatic mode

The motion parameters for path interpolation are available under "MotionSet.Translation" and "MotionSet.Rotation".

The current software version also decelerates with the parameterized acceleration in automatic mode. In future versions, the set deceleration will also be used in automatic mode. If, for example, strong acceleration and soft deceleration is required, this is possible in the current software version by using different motion parameter sets for the path segments.

Procedure

✓ The RobotMonitor has started and is connected.

✓ Access permission has been requested.

1. Switch to the "Variables" tab.
2. In the "Variables" tab, switch to the "MotionSet Var" tab.
   ⇨ The 8 motion parameter sets are displayed.
3. Expand the details of the required motion parameter set ("MotionSet") by clicking the arrow.
4. Under "MotionSet", open the "Kinematic" parameter group.
5. Under "Kinematic", open the parameter groups of the joint axes ("Joint") in use.
6. Set the motion parameters for speed, acceleration, deceleration, and jerk for each kinematic joint axis.
7. Under "MotionSet", open the parameter groups "Translation" and "Rotation".
8. Set the motion parameters for speed, acceleration, deceleration, and jerk for translation and rotation.
9. To confirm your settings, click the [Send] button marked in yellow.

INFORMATION

The motion parameter sets are now buffered in volatile memory. To save the motion parameter sets permanently to the memory card, click [Save].

6.12 Referencing axes and performing function test

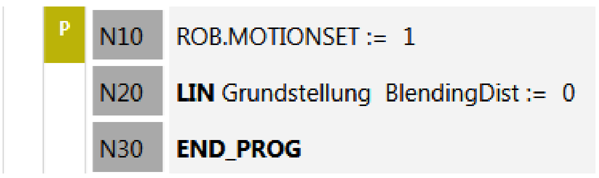
INFORMATION

When using a kinematic model of the type ROLLER GANTRY, note the special referencing feature as described for the respective kinematic model in the chapter "Kinematic model" (→ 🗎 33).

6.12.1 Function test for the simulated robot

✓ No error occurred.

1. Click the [Simulate Inverters] button.
   ⇨ The feedback *Inverters connected* appears at the bottom left of the user interface of the RobotMonitor.
2. Perform start-up with the RobotMonitor by following the steps described in the chapter "Process start" (→ 🗎 95).
3. Configure the sets of motion parameters.
4. Jog each axis of the robot with JOG_JOINT (+ and - buttons) to the zero positions of the joint axes. The values of the joint axes are displayed in the RobotMonitor in the 3D simulation window in the upper left corner. Note the position of the joint axes in the zero position as well as the positive direction of movement of the joint axes as they are displayed in the 3D simulation. You will have to reference the real axes accordingly and also set the direction of rotation of the real axes accordingly.
5. Using *JOG_JOINT* (+ and - buttons), jog the robot to the work envelope limits and verify these limits.
6. Using *JOG_CART* (+ and - buttons), jog the robot to the work envelope limits and verify these limits.
7. Create the following test program:



```
P N10 ROB.MOTIONSET :=  1
  N20 LIN Grundstellung  BlendingDist :=  0
  N30 END_PROG
```

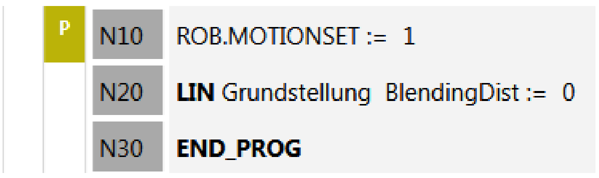
8. Start the test program and wait until the end of program.
9. Change the home position several times and test the program again.

6.12.2 Referencing and function test on the real robot

Perform a function test on the robot by carrying out the following steps:

1. Check the positive direction of movement of all axes in manual mode of MOVISUITE® according to what is displayed in the 3D simulation from step 4 in chapter "Function test for the simulated robot" (→ 🗎 48). If the positive direction of movement of a real axis does not match the positive direction of movement in the 3D simulation, configure a direction of rotation reversal in MOVISUITE®.
2. Reference the axes using the manual mode of MOVISUITE® or MOVIKIT® MultiMotion (request access, activate and start reference travel, return access after successful execution) according to the zero position that you determined in step 4 in chapter "Function test for the simulated robot" (→ 🗎 48).

3. Perform start-up with the RobotMonitor by following the steps described in chapter "Process start" (→ 🗎 95).
4. Jog each axis of the robot with JOG_JOINT (+ and - buttons) and compare the real movement with the movement shown in the 3D simulation. If you detect a deviation of the zero position or the direction of movement, perform step 1 or step 2 again for the respective axis.
5. Jog each axis of the robot with JOG_JOINT (+ and - buttons) by a certain distance for linear axes or by an angle for rotary axes, e.g. 100 mm or 90 °. Check whether the respective axis of the robot covers exactly this distance or angle. If this is not the case, correct the settings in the drive train of the axis in MOVISUITE®.
6. Next, use *JOG_JOINT* (+ and - buttons) to jog the robot to the work envelope limits and verify these limits.
7. Next, use *JOG_CART* (+ and - buttons) to jog the robot to the work envelope limits and verify these limits.
8. Create the following test program:



```
P  N10  ROB.MOTIONSET :=  1
   N20  LIN Grundstellung  BlendingDist :=  0
   N30  END_PROG
```

9. Start the test program and wait until the end of program.
10. Change the home position several times and test the program again. The movements of the real robot must correspond to the movements shown in the 3D simulation.
11. Create an IEC boot application.

6.13 Programming the robot

After the robot has passed the function test, you have to program the motion paths. For this purpose, use the RobotMonitor to create several robot programs.

For further information, refer to chapter "Robot program" (→ 🗎 99).

6.14 Integrating the robot

After having created the robot programs using the RobotMonitor, they must be started by a process controller (RobotMonitor, MOVI-C® CONTROLLER or higher-level controller).

For further information, refer to chapter "Control via the process controller" (→ 🗎 93).

7 Functional description

7.1 Robot states

7.1.1 Operating mode

⚠ DANGER

Non-safe movement of the robot due to non-safety rated functions of the robot controller (MOVI-C® CONTROLLER with MOVIKIT® Robotics)

Death, severe injuries or damage to property

- Before using a robot, carry out a safety assessment for the robot or the machine.

If a robot safety controller exists in the machine, an operating mode selector switch connected to the robot safety controller explicitly specifies the operating mode. If no robot safety controller exists, the process controller may explicitly specify the operating mode.

The following table shows an overview of the operating modes:

Distinguishing feature	**Operating mode**	
	Manually at high speed (**Manual High Speed**)	**Auto active** (**program**)
Jog mode possible?	Yes	No
Behavior during program mode	Jog behavior: Press and hold the program mode start button to continue both program mode and travel along the path (path jog).	Start/stop behavior: Press the program mode start button just once to start automatic program execution.

7.1.2 Enable mode

INFORMATION

You cannot explicitly select the enable mode as it results implicitly from the conditions described in chapter "Robot states".

INFORMATION

The drives are braked individually via the emergency stop in the inverter in the event of an axis-by-axis emergency stop. No central profile generator is used for braking. This can cause the lower-level MOVIKIT® MultiAxisController to jam the mechanics in the "Torque priority" operating mode and unexpected skew in the "Skew priority" operating mode.

The "Enable" state is required for the robot to execute a motion in jog mode and in program mode. The following enable modes are available:

0: Emergency stop (axis-by-axis), 1: Emergency stop (true-to-path), 2: Application stop, 3: Enable

This sequence corresponds to the hierarchy of the enable modes. A higher enable mode is activated if no cause for a lower enable mode applies.

The following table shows the differences in the various enable modes:

	Application stop	Emergency stop (true-to-path)	Emergency stop (axis-by-axis)
Calculation of the braking operation	MOVIKIT® Robotics	MOVIKIT® Robotics	MOVIKIT® of the axis group member
Path fidelity during braking	Yes	Yes	No
Set of motion parameters used for braking	Currently selected set of motion parameters	Parameterized set of motion parameters for emergency stop	Deceleration ramps of the MOVIKIT® axis group member
Behavior of the inverters in idle state/standstill	Interpolated positioning control (with a specified, constant setpoint position)	Emergency stop by means of the brakes or the position hold control of the inverter	Emergency stop by means of the brakes or the position hold control of the inverter
Path can continue after idle state/standstill	Yes	Yes, after "Repositioning" (→ 🗎 57) has been carried out.	Yes, after "Repositioning" (→ 🗎 57) has been carried out.
Possible causes for activation	Jog mode or program mode is not activated.	No enable	Access to the axis group member is activated and, therefore, not set to "Upper".
Possible causes for the software module	-	Fault status	-

	Application stop	Emergency stop (true-to-path)	Emergency stop (axis-by-axis)
Possible causes for axis group members	-	-	• Not connected • In safety stop ("STO activated") • Not ready • Not referenced • Fault status
Possible causes for the safety controller	Requires application stop	Requires emergency stop	Requires emergency stop (axis-by-axis)

7.1.3 Enable state

For software module diagnostics, the enable state of the robot is returned as a signal when it is enabled.

Enable states without access to axis group members

- No access to the axis group member requested: The robot does not request access to the axis group members because enabling the robot is not set and the setpoint of the software module is not activated.

Enable states with access to axis group members

- Enable states in which the setpoint of the robot motion control is not activated.
  - Axes in emergency stop: Emergency stop is requested from all axis group members. The axis group members report idle state/standstill.
  - Emergency stop (axis-by-axis): Emergency stop is requested from all axis group members. The axis group members apply the brakes.
  - Wait for "Setpoint active": Interpolated position control is requested for all axis group members. The system waits for the setpoint to be activated.
- Enable states in which interpolated position control for all axis group members is activated and in which the setpoint of the robot motion control is activated.
  - Emergency stop on the path: The robot motion control is currently braking on the path with the specified emergency stop ramps.
  - Application stop on the path: The robot motion control is currently braking on the path with the application ramps.
  - Position stop control: Setpoint of the robot motion control is constant (idle state). Application stop of the robot is activated.
  - Waiting on motion command: The robot motion control is waiting for the next motion command from the robot program.
  - Path motion: The robot motion control is currently carrying out a motion command.

7.1.4 Type of control

The software module offers the following types of controls for the robot:

INFORMATION

You cannot explicitly select the type of control, as this occurs implicitly when controlling the jog control signals and the program control signals.

Distinguishing feature	Inactive	Jog mode	Program mode
Description	Jog mode and program mode are not activated. Braking operation is in progress.	Robot jog	Carrying out robot programs
Target specified by	(Only for braking operations: see enable mode.)	Jog signals	Robot program
Causes	• Jog mode inactive • Program mode inactive	At least one jog signal is activated.	Program start is activated.
Purpose	Temporary mode	• Availability check • Motion in case of an error	Automated movement of the robot (target state)
Maximum resulting enable mode	Application stop	Enable	Enable

7.2 Jog mode

⚠ CAUTION

Unexpected movement of the robot when acknowledging a fault in jog mode while the jog input signal is set.

Death, severe injuries or damage to property

- Before acknowledging a fault in jog mode, make sure that no jog signal is set.

Jog mode allows for easy manual operation, for example, to check if a certain pose can be achieved.

In jog mode, the robot can be moved as follows:

- Jog mode along each joint axis ("Jog Joints")
  - Jogging the joint axes within the joint axis coordinates.
- Jog mode along the Cartesian spatial axes ("Jog Cartesian")
  - Jogging the robot tool within the Cartesian translation and orientation coordinates.

The motion profiles for jog mode are defined by the set of motion parameters chosen; (by default, this is set of motion parameters number 8).

7.3 Program mode

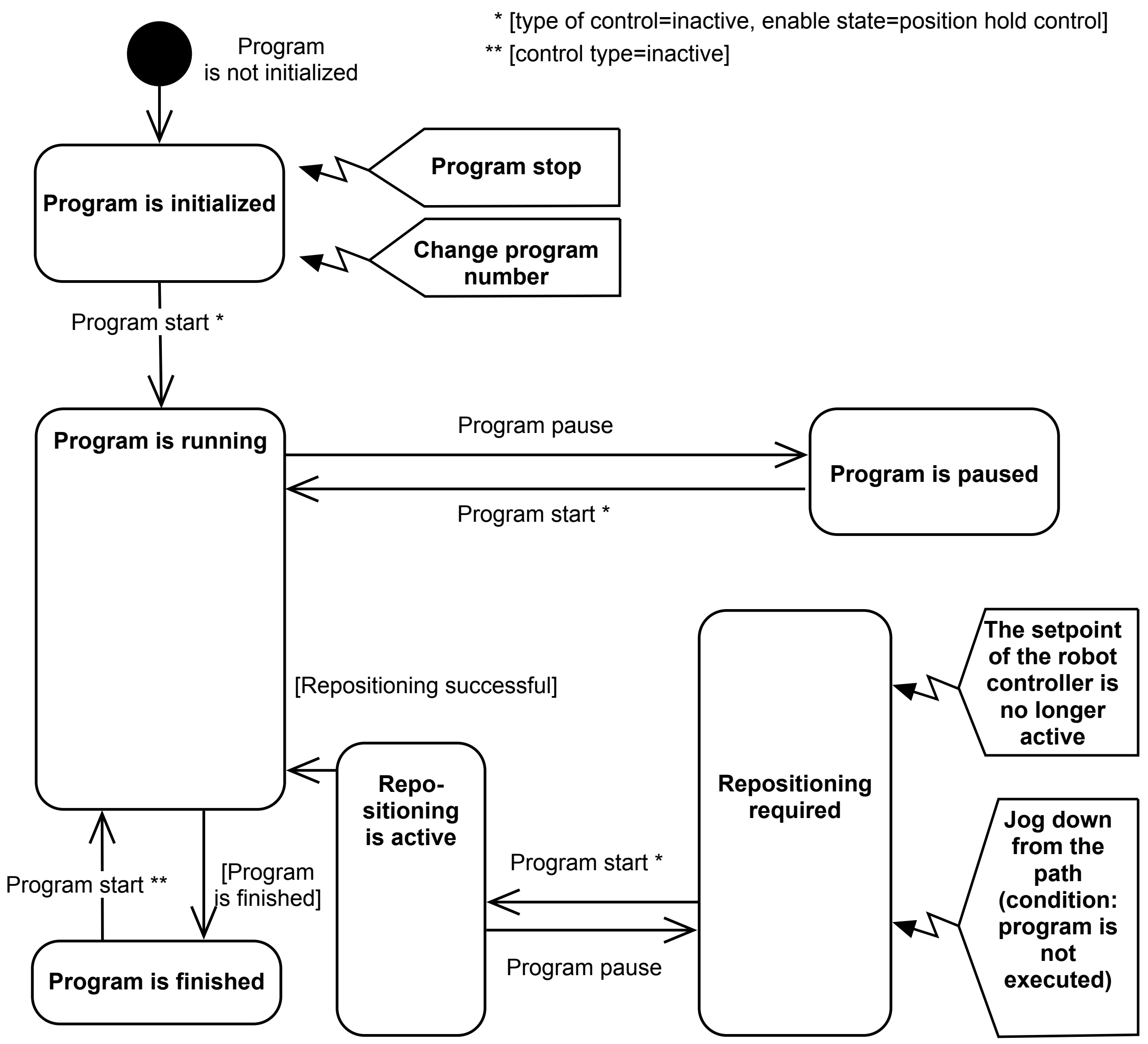
Program mode is used by robot programs to travel along the path points. The following edge-controlled signals are used for program control:

Start, pause, stop

Program execution may be in one of the following states:

Program not initialized, program initialized, program is executed, program paused, program terminated, repositioning required, repositioning active

The following diagram illustrates the transitions between the different states:



For further information on the robot program, refer to chapter "Robot program" (→ 🗎 99).

INFORMATION

– Even when the program has ended, repositioning can be successful, e.g. if you jog off the path or if the setpoint is no longer active. If you do not want to carry out repositioning, you can suppress it by stopping the program.

– A program can only be changed if the current program has been initialized or is finished. If the program number is changed while the current program has already started, but is not yet finished, then an error message is issued.

– If you wish to abort a program and start another program, you have to stop and initialize the current program by carrying out a program stop.

7.3.1 Types of program sequence

The following types of sequence exist for program execution:

- AUTO – The program runs to the end.
- STEP_BLOCK – A program line is executed.

7.3.2 Establishing the basic state

When a program is called, the software automatically creates a defined basic state, which means that certain motion control parameters are set. This basic state is not established when calling subprograms.

The following parameters are set each time the program is started:

- The target position is set to the current position. In this way only individual coordinates can be controlled irrespective of previous programs without changing the other coordinates.
- Blending is enabled.
- The blending distance is set to 0 mm.
- The base coordinate system is selected.
- The target is set to absolute position control (not to relative position control).
- The type of movement is set to linear interpolation.
- The X-Y plane is set as the plane for circular interpolation.
- The specification for additional positions in circular interpolation is set to relative position control.
- The parameters for path events are set to their default values.

  Reference point = start of segment, distance = 0 mm, time offset = 0 s
- The parameters for touchprobe events and sensor-based positioning are set to default values:

  Trigger source = inverter, index of trigger Boolean variable = 1, level = rising edge, mode = single, measuring direction = Z, length of remaining distance = 0 mm, touchprobe counter is set to "0".

7.4 Repositioning

If the system leaves the specified path, you will need to reposition the system on the path in order to continue the motion. The system leaves the path when the setpoint of the robot controller is deactivated in the "Program is running" or "Program is paused" state. This happens with enable modes "Emergency stop (true-to-path)", "Emergency stop (axis by axis)", and during idle state. The robot then reports a program status of "Repositioning is required". If the setpoint of the motion control is reactivated (enable state "position hold control"), the system is repositioned back on the path at a rising edge at program start.

You can stop repositioning using program pause or, with program start deactivated, using the "Manual high speed" operating mode. You can start repositioning again using program start (program pause and program stop must not be set).

When the program is in the "Program is running" state, repositioning is complete. After repositioning, the program continues automatically. A rising edge at the program start is not required.

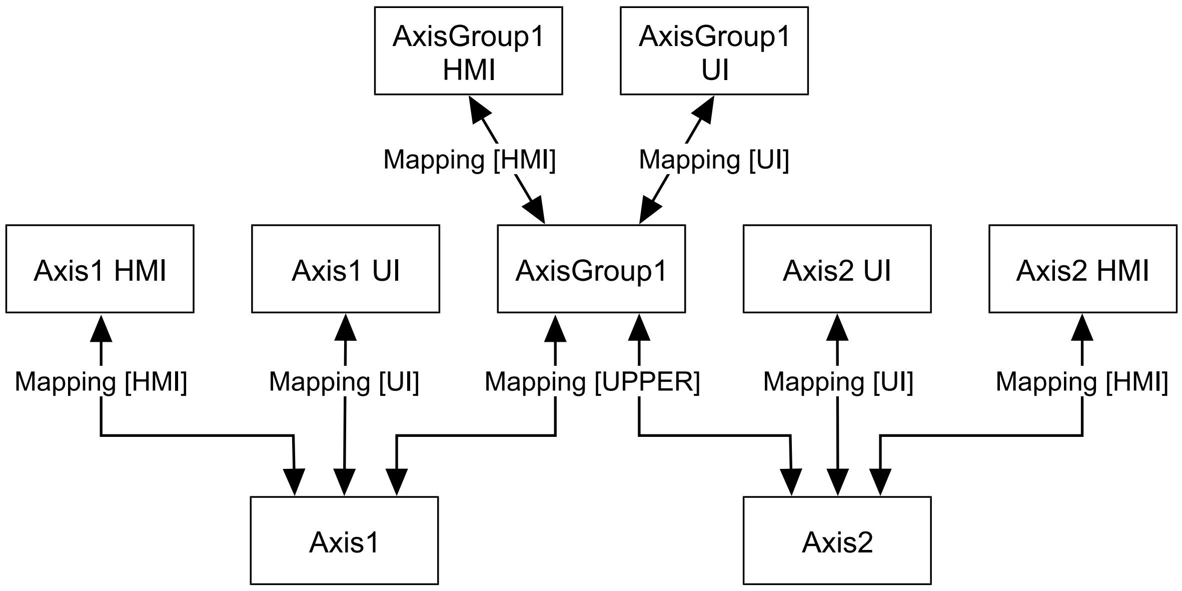
During repositioning, all joint axes are repositioned back on the path in a synchronized manner ("point-to-point"). The set of motion parameters specified in the interface for repositioning is used for repositioning.

7.5 Access management

The access management system controls which control signal source may access the user interface of the software module.

Each control signal source may request access and receives a corresponding feed-back message as to whether access is possible.

For the software module, the control signal sources of HMI, UI, and UPPER are avail-able. This order corresponds to the priority of the control signal sources regarding ac-cess.



24519746443

A separate access management system exists for each single axis. If access is gran-ted via UI or HMI for a single axis, the software module cannot access it.

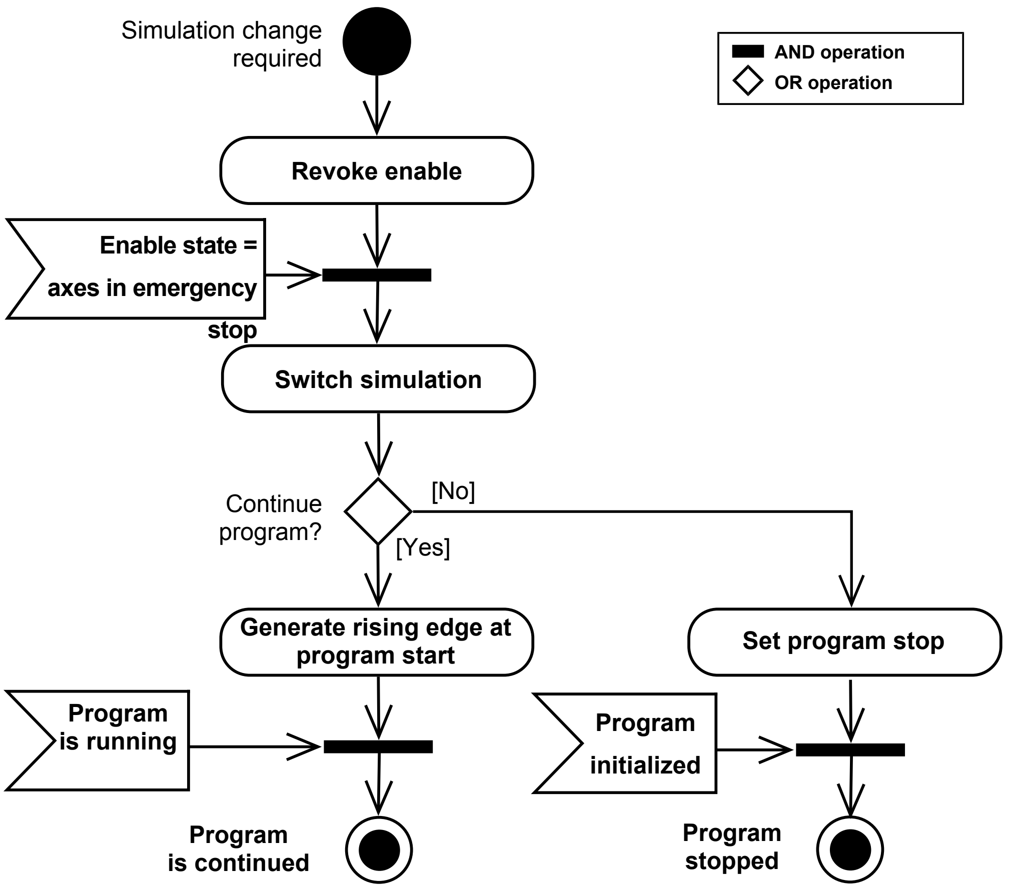
7.6 Simulation

The robot can be simulated by setting the *Simulation* input signal. Doing so will set all lower-level axis groups or axis group members to simulation. This means the state of the inverters is ignored and a simulated drive is run instead.

This can be useful in the following cases:

- No inverters are connected to the robot but the control and movement of the robot is to be tested.
- Inverters are connected but a new movement has been programmed, which is to be tested in a simulated way prior to the real application.

For the second application case, it is necessary to switch from real to simulated axes and vice versa. For this case, the axes (whether simulated or real) must be in a safe state. This is the case when the enable is removed from the robot and all inverters are in emergency stop and report standstill. The following diagram illustrates the switchover:



31362370315

You can also simulate only some of the robot's axes. In this case the simulation must be activated in MOVISUITE® for the respective axes. The robot will then ignore the specified value for these axes. The axes remain simulated as long as the setting is active in MOVISUITE®.

7.7 Kinematic models

The software module provides numerous kinematic models for configuring applications. These kinematic models are selected and configured using the "Kinematic model" (→ 🗎 33) configuration menu. In this chapter you find more detailed information about the kinematic models and the parameters they contain.

INFORMATION

The selected kinematic model (displayed in the "3D simulation" (→ 🗎 104)) must match the real robot. Otherwise, the kinematic control cannot be operated correctly. In addition, the real axes of the robot must be referenced according to the display in the 3D simulation and adjusted in the direction of rotation. You find an instruction on how to do this further down in the chapter "Referencing axes and performing function test" (→ 🗎 48).

7.7.1 Emergency stop ramps of the joints

The emergency stop ramps are configured for all available kinematics models using the following setting fields:

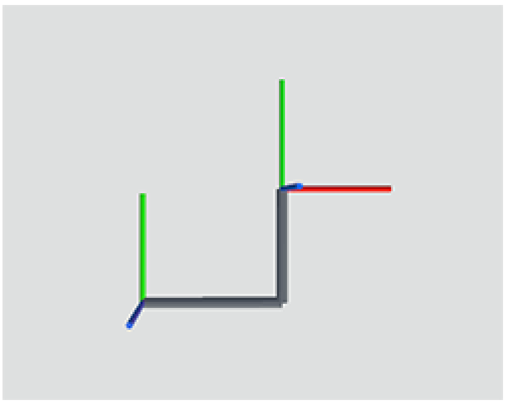
Parameter name	Description
Emergency stop deceleration	Rate of deceleration with which the joint axis brakes during an emergency stop.
Emergency stop jerk	Jerk with which the deceleration of the joint axis is generated during an emergency stop. This setting is used for jerk limitation; (that is, to prevent impacts and damage to the mechanical components).
Software limit switch	Permitted range of the joint axis. If this range is exceeded, a fault status occurs. Select the software limit switches in such a way that they are located in front of the software limit switches of the inverters within the work envelope and away from the software limit switches of the inverters by the braking distance.

See also "Sets of motion parameters" (→ 🗎 21) and "Setting motion parameters" (→ 🗎 161).

29180252/EN – 03/2020

7.7.2 CARTESIAN_GANTRY_LL_M10

Included in the MOVIKIT® Robotics license.

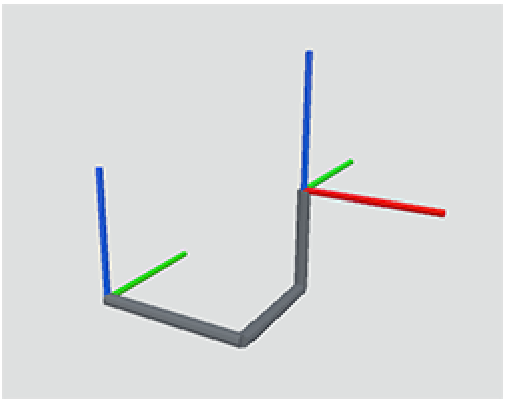
CARTESIAN_GANTRY_LL_M10	
	CARTESIAN GANTRY (portal) with 2 linear axes: • Joint axis 1: X direction • Joint axis 2: Y direction

Joints

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

7.7.3 CARTESIAN_GANTRY_LLL_M10

Included in the MOVIKIT® Robotics add-on MediumModels license.

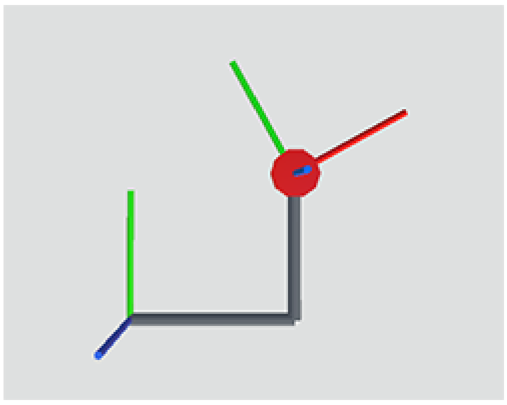
CARTESIAN_GANTRY_LLL_M10	
	CARTESIAN GANTRY (portal) with 3 linear axes: • Joint axis 1: X direction • Joint axis 2: Y direction • Joint axis 3: Z direction

Joints

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

7.7.4 CARTESIAN_GANTRY_LLR_M10

Included in the MOVIKIT® Robotics add-on MediumModels license.

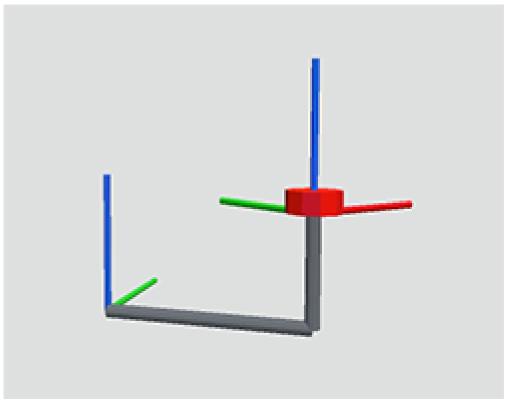
CARTESIAN_GANTRY_LLR_M10	
	CARTESIAN GANTRY (portal) with 2 linear axes and 1 rotary axis (rotation around Z): • Joint axis 1: X direction • Joint axis 2: Y direction • Joint axis 3: rotation around Z

Joints

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

7.7.5 CARTESIAN_GANTRY_LLR_M20

Included in the MOVIKIT® Robotics add-on MediumModels license.

CARTESIAN_GANTRY_LLR_M20	
	CARTESIAN GANTRY (portal) with 2 linear axes and 1 rotary axis (rotation around Z): • Joint axis 1: X direction • Joint axis 2: Z direction • Joint axis 3: rotation around Z

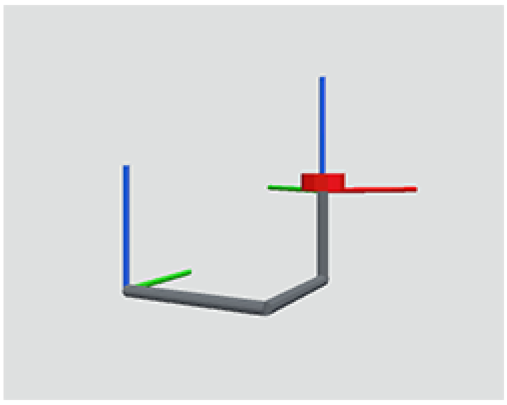
Joints

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

29180252/EN – 03/2020

7.7.6 CARTESIAN_GANTRY_LLLR_M10

Included in the MOVIKIT® Robotics add-on MediumModels license.

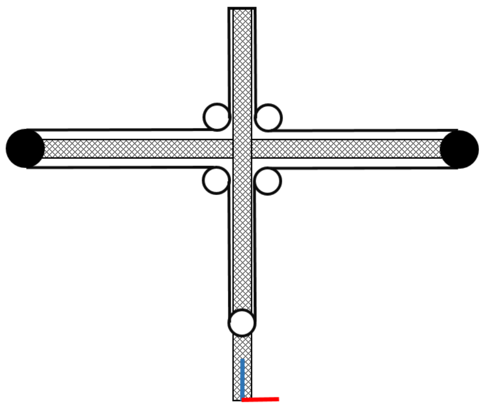
CARTESIAN_GANTRY_LLLR_M10	
	CARTESIAN GANTRY (portal) with 3 linear axes and 1 rotary axis (rotation around Z): • Joint axis 1: X direction • Joint axis 2: Y direction • Joint axis 3: Z direction • Joint axis 4: rotation around Z

Joints

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

7.7.7 ROLLER_GANTRY_RR_M10

Included in the MOVIKIT® Robotics license.

ROLLER_GANTRY_RR_M10	
	ROLLER GANTRY with 2 rotary axes: • Joint axes 1, 2: Drive of the rotating belt in the zx plane • Joint axis 2: Mounted in direction of the x-axis opposite joint axis 1

KinPar

Parameter name	Description
Roll 1 (left)	
Pitch	Pitch of belt pulley 1
Number of teeth	Number of teeth of belt pulley 1
Roll 2 (right)	
Pitch	Pitch of belt pulley 2
Number of teeth	Number of teeth of belt pulley 2

Joints

The software limit switches of joint axes 1 and 2 should remain unused, i.e. set to very large values.

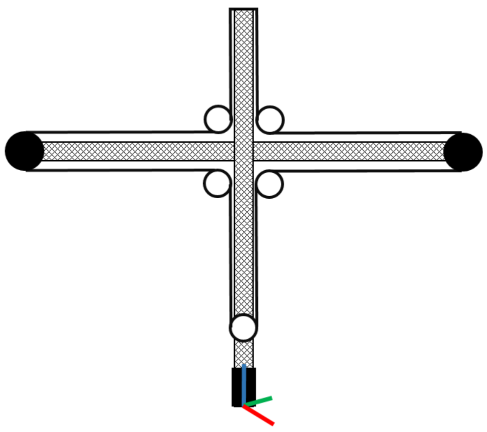
Joint axes 1 and 2 are initially referenced by referencing the individual axes at the position where the robot is located. The robot can then be moved in a Cartesian manner up to the reference switches, e.g. first in z and then in x direction. As soon as the reference switches are actuated, for example by a tool touching them, the robot program is stopped or the Cartesian jog mode is interrupted by removing the jog signals. Afterwards, the robot is referenced again at the position reached by referencing the individual axes, so that X = Z = 0.

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

29180252/EN – 03/2020

7.7.8 ROLLER_GANTRY_RRR_M10

Included in the license for MOVIKIT® Robotics add-on MediumModel.

ROLLER_GANTRY_RRR_M10	
	ROLLER GANTRY with 3 rotary axes: • Joint axes 1, 2: Drive of the rotating belt in the zx plane • Joint axis 2: Mounted in direction of the x-axis opposite joint axis 1 • Joint axis 3: Rotation around an axis parallel to the z-axis

KinPar

Parameter name	Description
Roll 1 (left)	
Pitch	Pitch of belt pulley 1
Number of teeth	Number of teeth of belt pulley 1
Roll 2 (right)	
Pitch	Pitch of belt pulley 2
Number of teeth	Number of teeth of belt pulley 2

Joints

The software limit switches of joint axes 1 and 2 should remain unused, i.e. set to very large values.

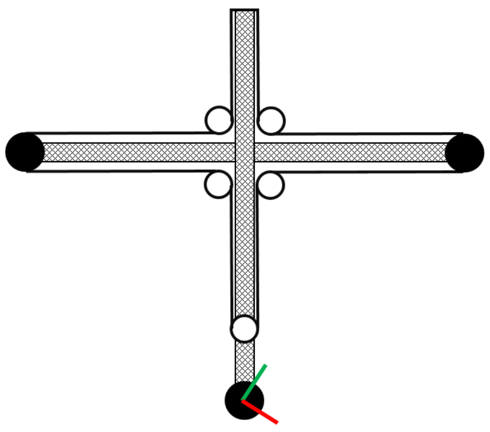
Joint axes 1 and 2 are initially referenced by referencing the individual axes at the position where the robot is located. The robot can then be moved in a Cartesian manner up to the reference switches, e.g. first in z and then in x direction. As soon as the reference switches are actuated, for example by a tool touching them, the robot program is stopped or the Cartesian jog mode is interrupted by removing the jog signals. Afterwards, the robot is referenced again at the position reached by referencing the individual axes, so that X = Z = 0.

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

7.7.9 ROLLER_GANTRY_RRLR_M10

Included in the license for MOVIKIT® Robotics add-on MediumModel.

ROLLER_GANTRY_RRLR_M10



ROLLER GANTRY with 3 rotary axes and 1 linear axis:

- Joint axes 1, 2: Drive of the rotating belt in the zx plane
- Joint axis 2: Mounted in direction of the x-axis opposite joint axis 1
- Joint axis 3: Z direction (pointing vertically out of drawing plane)
- Joint axis 4: Rotation around an axis parallel to the z-axis

KinPar

Parameter name	Description
Roll 1 (left)	
Pitch	Pitch of belt pulley 1
Number of teeth	Number of teeth of belt pulley 1
Roll 2 (right)	
Pitch	Pitch of belt pulley 2
Number of teeth	Number of teeth of belt pulley 2

Joints

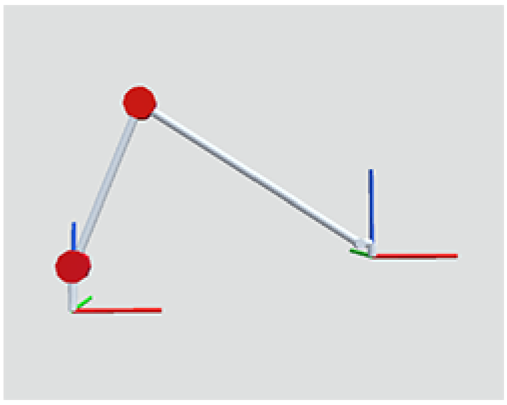
The software limit switches of joint axes 1 and 2 should remain unused, i.e. set to very large values.

Joint axes 1 and 2 are initially referenced by referencing the individual axes at the position where the robot is located. The robot can then be moved in a Cartesian manner up to the reference switches, e.g. first in z and then in x direction. As soon as the reference switches are actuated, for example by a tool touching them, the robot program is stopped or the Cartesian jog mode is interrupted by removing the jog signals. Afterwards, the robot is referenced again at the position reached by referencing the individual axes, so that X = Z = 0.

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

7.7.10 SCARA_RR_M10

Included in the MOVIKIT® Robotics license.

SCARA_RR_M10	
	SCARA with 2 rotary axes: • Joint axis 1: Rotates the upper arm around the shoulder axis (parallel to the Y axis). • Joint axis 2: Rotates the lower arm around the elbow axis (parallel to the Y axis). The flange orientation is constantly maintained by means of a parallelogram or a belt, or the tool only comprises a pen, for example; its orientation is not significant.

KinPar

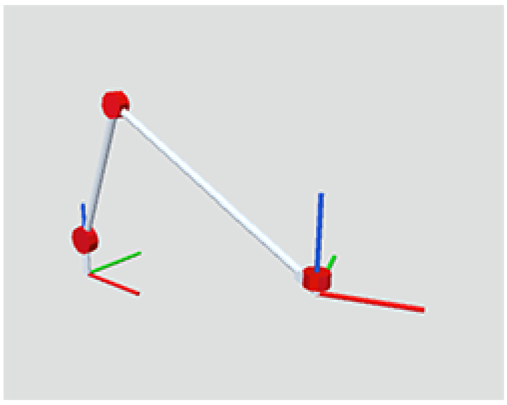
Parameter name	Description
Kinematic parameters	
Base offset (Z)	Base offset in the Z direction
Length upper arm	Length of the upper arm
Length forearm	Length of the lower arm
Flange offset (X)	Flange offset in the X direction
Flange offset (Z)	Flange offset in the Z direction

Joints

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

7.7.11 SCARA_RRR_M10

Included in the MOVIKIT® Robotics add-on MediumModels license.

SCARA_RRR_M10	
	SCARA with 3 rotary axes: • Joint axis 1: Rotates the upper arm around the shoulder axis (parallel to the Y axis). • Joint axis 2: Rotates the lower arm around the elbow axis (parallel to the Y axis). The drive is stationary. • Joint axis 3: Rotates the tool around the wrist axis (parallel to the Z axis).

KinPar

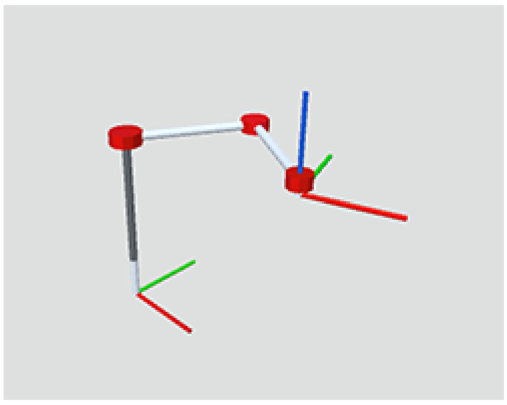
Parameter name	Description
Kinematic parameters	
Base offset (Z)	Base offset in the Z direction
Length upper arm	Length of the upper arm
Length forearm	Length of the lower arm
Flange offset (X)	Flange offset in the X direction
Flange offset (Z)	Flange offset in the Z direction

Joints

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

7.7.12 SCARA_LRRR_M10

Included in the MOVIKIT® Robotics add-on MediumModels license.

SCARA_LRRR_M10	
	SCARA with 1 linear axis and 3 rotary axes: • Joint axis 1: Z direction • Joint axis 2: Rotates the upper arm around the shoulder axis (parallel to the Z axis). • Joint axis 3: Rotates the lower arm around the elbow axis (parallel to the Z axis). • Joint axis 4: Rotates the tool around the wrist axis (parallel to the Z axis).

KinPar

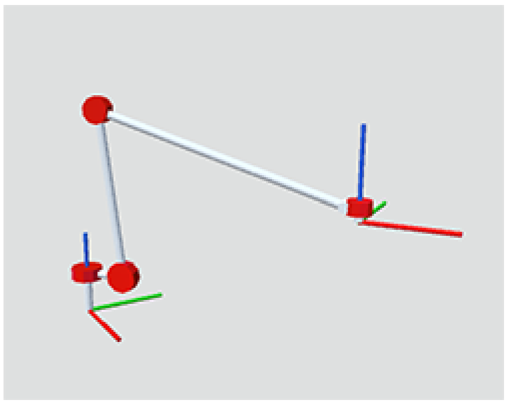
Parameter name	Description
Kinematic parameters	
Base offset (Z)	Base offset in the Z direction
Length upper arm	Length of the upper arm
Offset upper arm -> forearm	Offset between upper and lower arm in the Z direction
Length forearm	Length of the lower arm
Flange offset (Z)	Flange offset in the Z direction

Joints

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

7.7.13 SCARA_RRRR_M10

Included in the MOVIKIT® Robotics add-on MediumModels license.

SCARA_RRRR_M10	
	SCARA with 4 rotary axes: • Joint axis 1: Rotates the turret around the turret axis (parallel to the Z axis). • Joint axis 2: Rotates the upper arm around the shoulder axis. • Joint axis 3: Rotates the lower arm around the elbow axis. • Joint axis 4: Rotates the tool around the Z axis. The inclination of the tool is constantly maintained by means of a parallelogram or a belt.

KinPar

Parameter name	Description
Kinematic parameters	
Base offset (Z)	Base offset in the Z direction
Base offset (X)	Base offset in the X direction
Base offset (Y)	Base offset in the Y direction
Length upper arm	Length of the upper arm
Length forearm	Length of the lower arm
Flange offset (X)	Flange offset in the X direction
Flange offset (Z)	Flange offset in the Z direction

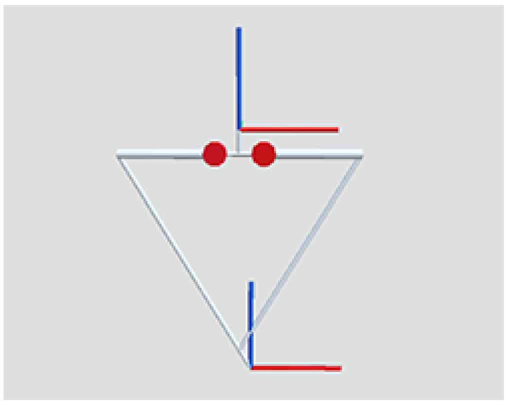
Joints

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

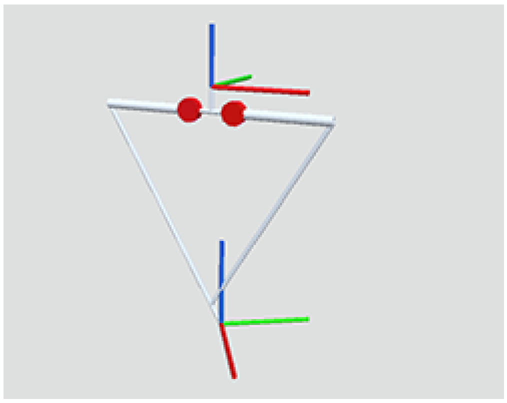
29180252/EN – 03/2020

7.7.14 DELTA_RR/RRR_M10

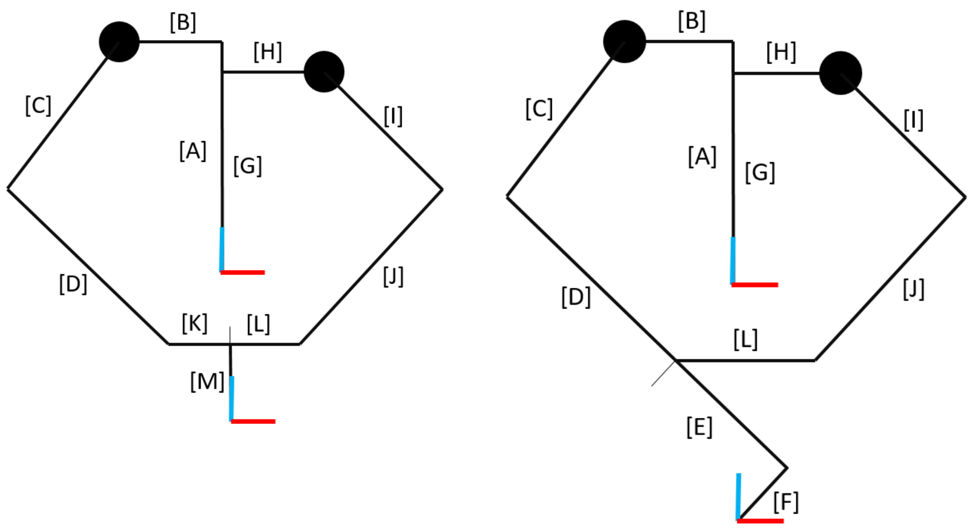
Included in the MOVIKIT® Robotics license.

DELTA_RR_M10	
	DELTA with 2 rotary axes: • Joint axes 1 and 2: In combination, they allow translational motion of the tool on the ZX plane. The motion of a joint axis cannot be clearly assigned to the motion of a Cartesian axis.

Included in the MOVIKIT® Robotics add-on MediumModels license.

DELTA_RRR_M10	
	DELTA with 3 rotary axes: • Joint axes 1 and 2: In combination, they allow translational motion of the tool on the ZX plane. The motion of a joint axis cannot be clearly assigned to the motion of a Cartesian axis. • Joint axis 3: Rotates the tool around the wrist axis (parallel to the Z axis).

KinPar



Parameter name		Description
Left side		
(A)	Offset to joint (Z)	Offset to joint 1 in the Z direction
(B)	Offset to joint (X)	Offset to joint 1 in the X direction
(C)	Length upper arm	Length of the upper arm
(D)	Length forearm	Length of the lower arm
(E)	Extension	Extension of the lower arm starting from its connection to the other arm
(F)	Offset	Orthogonal offset
Right side		
(G)	Offset to joint (Z)	Offset to joint 2 in the Z direction
(H)	Offset to joint (X)	Offset to joint 2 in the X direction
(I)	Length upper arm	Length of the upper arm
(J)	Length forearm	Length of the lower arm
Tool plate		
(K)	Distance left side	Tool plate, left section
(L)	Distance right side	Tool plate, right section
(M)	Offset (Z)	Flange offset in the Z direction

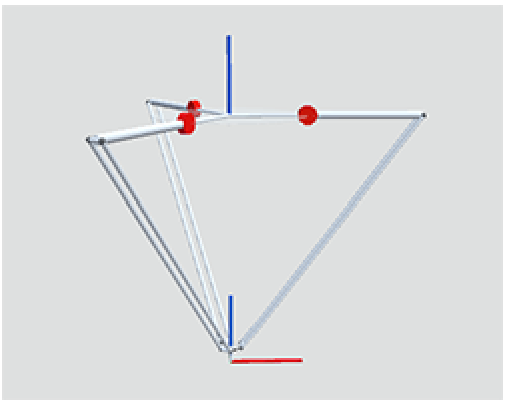
Joints

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

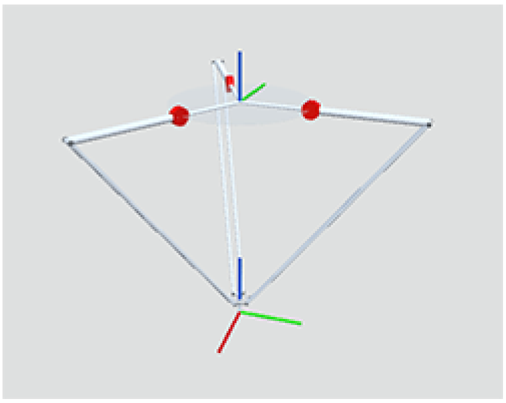
29180252/EN – 03/2020

7.7.15 TRIPOD_RRR/RRRR_M10

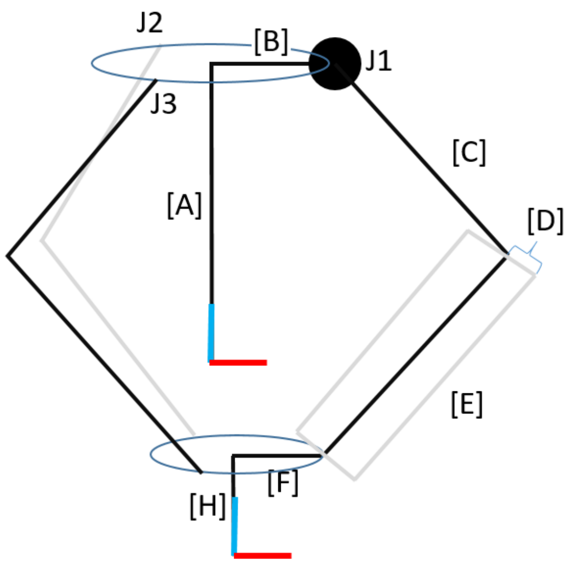
Included in the MOVIKIT® Robotics add-on MediumModels license.

TRIPOD_RRR_M10	
	TRIPOD with 3 rotary axes: • Joint axes 1 to 3: In combination, they allow translational motion of the tool (XYZ). The motion of a drive cannot be clearly assigned to the motion of a Cartesian axis.

Included in the MOVIKIT® Robotics add-on MediumModels license.

TRIPOD_RRRR_M10	
	TRIPOD with 4 rotary axes: • Joint axes 1 to 3: In combination, they allow translational motion of the tool (XYZ). The motion of a drive cannot be clearly assigned to the motion of a Cartesian axis. • Joint axis 4: Rotates the tool around the wrist axis (parallel to the Z axis).

KinPar



Parameter name		Description
Tripod kinematic parameters		
(A)	Height offset shoulder plate	Offset to the height of the shoulder joints in the Z direction
(B)	Radius shoulder plate	Radius of the circle formed by the shoulder joints
(C)	Length upper arms	Length of the upper arms
(D)	Half-width parallelogram	Half width of the parallelograms
(E)	Length forearms	Length of the lower arms
(F)	Radius tool plate	Radius of the circle formed by the intersections of the tool plate with the centerlines of the parallelograms
(G)	Horizontal offset tool plate	Flange offset in the X direction
(H)	Vertical offset tool plate	Flange offset in the Z direction

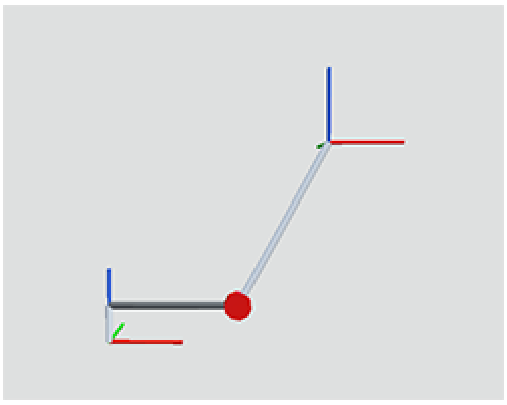
Joints

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

29180252/EN – 03/2020

7.7.16 MIXED_LR_M10

Included in the MOVIKIT® Robotics license.

MIXED_LR_M10	
	A MIXED kinematic model with 1 linear axis and 1 rotary axis • Joint axis 1: X direction • Joint axis 2: Rotation around an axis parallel to the Y axis. The flange orientation is constantly maintained by means of a parallelogram or a belt, or the tool only comprises a pen, for example; its orientation is not significant.

KinPar

Parameter name	Description
Kinematic parameters	
Base offset (Z)	Base offset in the Z direction
Arm length	Length of the arm

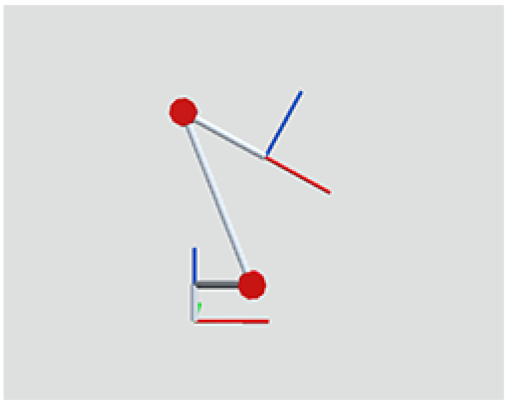
Joints

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

7.7.17 MIXED_LRR_M10

Included in the MOVIKIT® Robotics add-on MediumModels license.

MIXED_LRR_M10



A MIXED kinematic model with 1 linear axis and 2 rotary axes

- Joint axis 1: X direction
- Joint axis 2: Rotation around an axis parallel to the Y axis.
- Joint axis 3: Rotation around an axis parallel to the Y axis.

KinPar

Parameter name	Description
Kinematic parameters	
Base offset (Z)	Base offset in the Z direction
Upper arm length	Length of the upper arm
Forearm length	Length of the lower arm

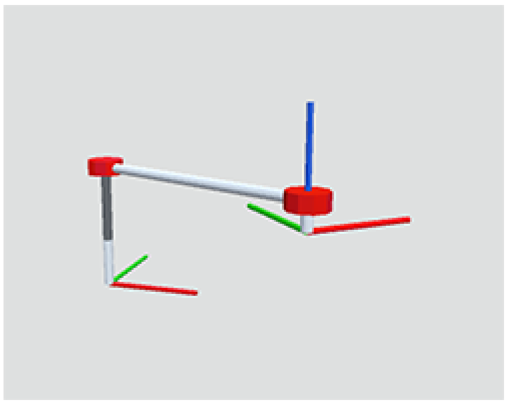
Joints

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

29180252/EN – 03/2020

7.7.18 MIXED_LRR_M20

Included in the MOVIKIT® Robotics add-on MediumModels license.

MIXED_LRR_M20	
	A MIXED kinematic model with 1 linear axis and 2 rotary axes that allows motion of the flange on the surface of a cylinder around the linear axis as well as rotation of the flange (around the axis parallel to Z). The X axis runs horizontally along the circular arc of the cylinder around the linear axis: • Joint axis 1: Z direction • Joint axis 2: Rotates the arm around the shoulder axis (parallel to the Z axis). • Joint axis 3: Rotates the tool around the wrist axis (parallel to the Z axis).

KinPar

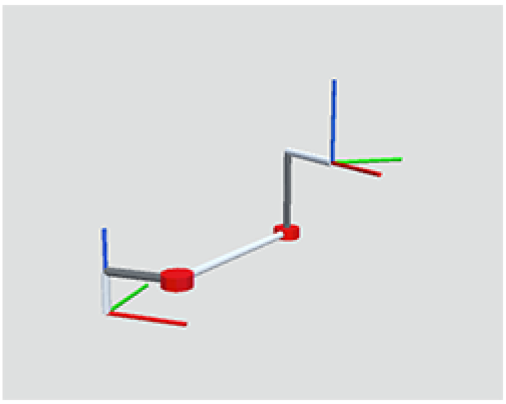
Parameter name	Description
Kinematic parameters	
Base offset (Z)	Base offset in the Z direction
Arm length	Length of the arm
Flange offset (Z)	Flange offset in the Z direction

Joints

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

7.7.19 MIXED_LRRL_M10

Included in the MOVIKIT® Robotics add-on MediumModels license.

MIXED_LRRL_M10	
	A MIXED kinematic model with 2 linear axes and 2 rotary axes: • Joint axis 1: X direction • Joint axis 2: Rotation around an axis parallel to the Z axis • Joint axis 3: Rotation around an axis parallel to the Z axis • Joint axis 4: Z direction

KinPar

Parameter name	Description
Kinematic parameters	
Base offset (Z)	Base offset in the Z direction
Upper arm length	Length of the upper arm
Forearm length	Length of the lower arm
Flange offset (Z)	Flange offset in the Z direction

Joints

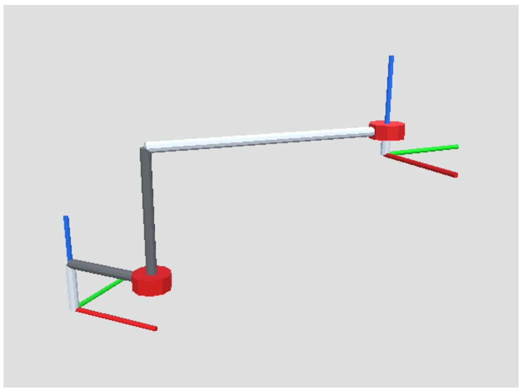
To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

29180252/EN – 03/2020

7.7.20 MIXED_LRLR_M10

Included in the MOVIKIT® Robotics add-on MediumModels license.

MIXED_LRRL_M10



A MIXED kinematic model with 2 linear axes and 2 rotary axes:

- Joint axis 1: X direction
- Joint axis 2: Rotation around an axis parallel to the Z axis
- Joint axis 3: Z direction
- Joint axis 4: Rotation around an axis parallel to the Z axis

KinPar

Parameter name	Description
Kinematic parameters	
Base offset (Z)	Base offset in the Z direction
Arm length	Length of the arm between joint axis 3 and 4
Flange offset (Z)	Flange offset in the Z direction

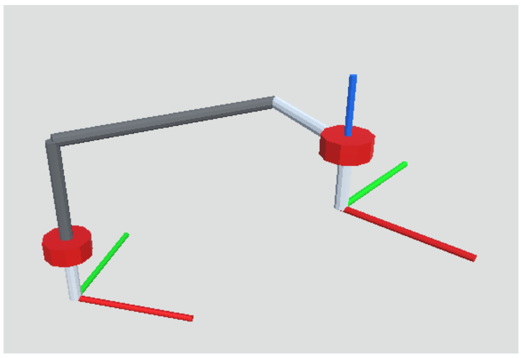
Joints

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

29180252/EN – 03/2020

7.7.21 MIXED_RLLR_M10

Included in the MOVIKIT® Robotics add-on MediumModels license.

MIXED_LRRL_M10	
	A MIXED kinematic model with 2 linear axes and 2 rotary axes: • Joint axis 1: Rotation around an axis parallel to the Z axis • Joint axis 2: Z direction • Joint axis 3: Linear axis in xy plane • Joint axis 4: Rotation around an axis parallel to the Z axis

KinPar

Parameter name	Description
Kinematic parameters	
Base offset (Z)	Base offset in the Z direction
Side offset	Offset vertical to joint axis 3 In y direction if joint axis 1 = 0°
Flange offset (Z)	Flange offset in the Z direction

Joints

To configure the emergency stop ramps, see chapter "Emergency stop ramps of the joints" (→ 🗎 60).

29180252/EN – 03/2020

7.8 Path events

Path events allow for executing an instruction at a parameterizable point in the path progression. The path events are parameterized by specifying a reference point, shifting the reference point using the *Distance* parameter, and a time offset using the *Time* parameter. For further information, refer to chapter "Path event" (→ 🗎 125).

The parameterization of the path event refers to the next path segment in the robot program and takes place in 3 steps:

1. Specifying a reference point: Starting point or end point of the next path segment in the robot program. The robot program is only displayed as error-free in the RobotMonitor when the REG PATH_EVENT registry is followed by a motion command.
   - ⇨ If reference = FALSE: Starting point of the path segment
   - ⇨ If reference = TRUE: End point of the path segment
2. Shifting the reference point using the *Distance* parameter in [mm]: The value 0 does not result in a shift of the reference point. A negative value causes the reference point to be shifted forward in the path progression, i.e. to the beginning of the path. A positive value causes the reference point to be shifted backwards in the path progression, i.e. towards the end of the path.
   - ⇨ If distance = 0: exactly at the reference point
   - ⇨ If distance is negative: in the path progression in front of the reference point
   - ⇨ If distance is positive: in the path progression behind the reference point
3. Time offset using the *Time* parameter in [ms]: The value 0 does not result in a time shift. A negative value results in a time shift before reaching the shifted reference point, i.e. a lead time. A positive value results in a time shift after reaching the shifted reference point, i.e. a delay time.
   - ⇨ If time = 0: in the shifted reference point
   - ⇨ If time negative: before reaching the shifted reference point
   - ⇨ If time positive: after reaching the shifted reference point

INFORMATION

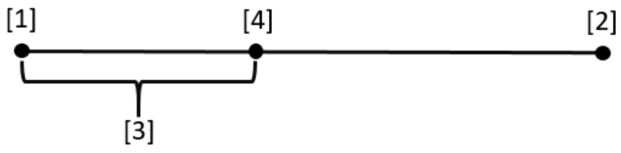
If several instructions are to be executed at the same point in the path progression, you can insert several path events with the same parameter setting into the robot program.

The examples explained in the following chapters illustrate how the described parameters work.

29180252/EN – 03/2020

7.8.1 Example 1

Path event with reference point at the starting point of the segment, shifted backwards in path progression by a positive value of the *Distance* parameter, *Time* parameter = 0.

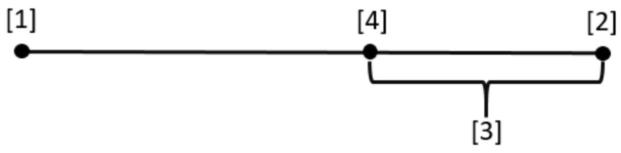


31386523275

[1] Starting point of the path segment
[2] End point of the path segment
[3] *Distance* in [mm], positive value
[4] Point at which the path event is triggered

7.8.2 Example 2

Path event with reference point at the end point of the segment, shifted forwards in path progression by a negative value of the *Distance* parameter, *Time* parameter = 0.

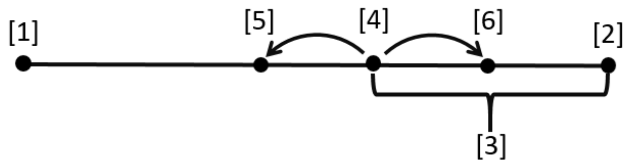


31386525707

[1] Starting point of the path segment
[2] End point of the path segment
[3] *Distance* in [mm], positive value
[4] Point at which the path event is triggered

7.8.3 Example 3

Path event with reference point at the end point of the segment, shifted forwards in path progression by a negative value of the *Distance* parameter and with different values of the *Time* parameter.



31386528139

[1] Starting point of the path segment
[2] End point of the path segment
[3] *Distance* in [mm], negative value
[4] Point at which the path event is triggered if *Time* = 0
[5] Point at which the path event is triggered if *Time* < 0
[6] Point at which the path event is triggered if *Time* > 0

The reference point and the time can be shifted across several path segments.

The shifts in the forward path progression take place up to a maximum of one stop point. Stop points are the starting point of the path and the starting points of path segments to which no blending is performed.

If the reference point shifted by the *Distance* parameter is located before the last stop point, error message 16#7E82 "PathEventPositionBeforePathBegin" will be issued.

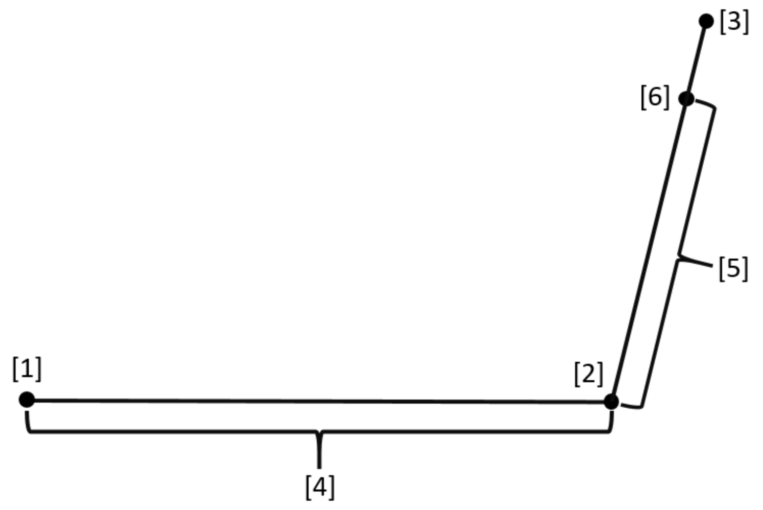
If the parameterized lead time (negative value in the *Time* parameter) is longer than the time for moving to the shifted reference point (using the *Distance* parameter), the following behavior occurs:

- When the robot (in the current coordinate system) is at a standstill, the path event is triggered and the lead time starts to expire. The robot starts the movement only when it is at the shifted reference point after the parameterized lead time has expired. During the waiting period, a corresponding message is displayed in the RobotMonitor.
- When the robot (in the current coordinate system) is in motion, the error message 16#7E83 "PathEventTimeShiftExceedsRemainingTimeInMotion" is issued.

The shifts in the path progression to the rear also take place across stop points, but at most up to a command that expects the end of the movement: "WAIT MotionDone" or "END_PROG WaitMotion".

7.8.4 Example 4

Path events with a large, positive value of the *Distance* parameter which causes the reference point to be shifted from the starting point across the segment border into the next path segment.



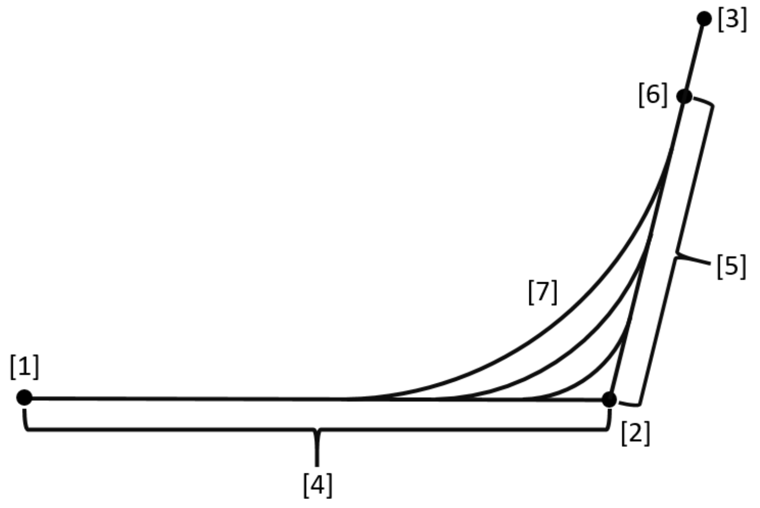
31386543371

[1]	Starting point of the path segment that is programmed next in the robot program after the REG PATH_EVENT command
[2]	End point of the path segment
[3]	End point of the subsequent path segment
[4]+[5]	*Distance* in [mm], positive value
[6]	Point at which the path event is triggered

The reference point shifted by means of the *Distance* parameter is always at the same position outside the blending range. Accordingly, the *Distance* parameter always refers to the path without blending regardless of whether the path is blended or not.

7.8.5 Example 5

Path event as in example 4 but with differently sized blending curves: Point [6] at which the path event is triggered, independent of blending.



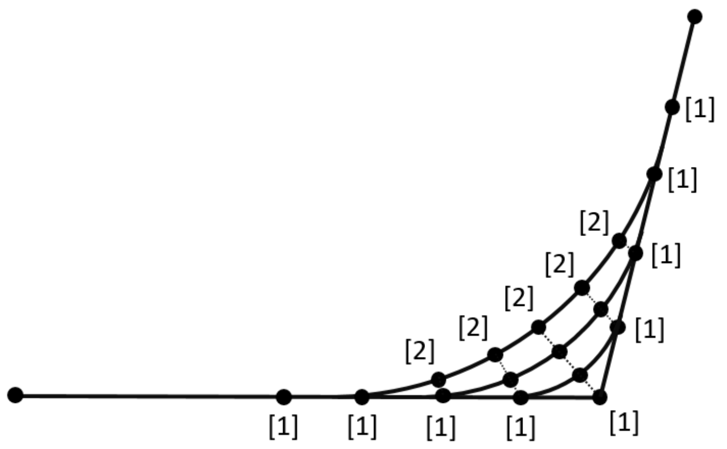
31386545803

[1]	Starting point of the path segment that is programmed next in the robot program after the REG PATH_EVENT command
[2]	End point of the path segment
[3]	End point of the subsequent path segment
[4]+[5]	*Distance* in [mm], positive value
[6]	Point at which the path event is triggered
[7]	Several differently sized blending curves

29180252/EN – 03/2020

7.8.6 Example 6

Reference points within blending range



31386548235

[1] Reference points on the path shifted using the *Distance* parameter without blending at which the respective path event is triggered
[2] Reference points shifted on the blending curves

The number of path events in a robot program is not limited. The maximum number of path events in the program is 64. If 64 path events are registered in the robot program, the program pointer will not advance until previously registered path events have been triggered and are no longer active. This situation is displayed in the RobotMonitor with an appropriate message.

The robot program does not end until all registered path events have been triggered. This means that if the shifted reference point is behind the end point of the last interpreted path segment or if a lead time has not yet expired, the robot program will not end. In these cases, the RobotMonitor will display a corresponding message.

If the robot program is paused and/or the robot is moved away from the path in jog mode, path events are still triggered at the end of an already started lead time. This situation is indicated by warning 16#17E82 "PathEventTime-ShiftElapsedNotInProgramMode" in the RobotMonitor.

When starting and stopping a robot program, all previously registered path events are deleted.

The path events refer to the setpoint path progression. Accordingly, a lag error can lead to inaccuracies when triggering path events. Increased cycle times of the *HighPrio* task also lead to reduced accuracy because the path events are triggered in the *HighPrio* task.

7.9 Touchprobe

A touchprobe event is triggered either by the triggering of a sensor or the change of state of a BOOLEAN variable.

When the touchprobe event is triggered, the current axis positions are read and are used for position measurement or sensor-based positioning. The touchprobe function of MOVIKIT® Robotics differs from the touchprobe function of the individual axes by the transformation of position values of the individual axes into a Cartesian touchprobe position and mapping them to the path of the robot.

Application cases for the touchprobe function are, for example, palletizing or depalletizing with variable or unknown height of the parts or the sensor-based execution of actions.

The actions that can be executed during a touchprobe event can be divided into those that trigger a movement (positioning) and those that do not trigger a movement (measure, Set_Boolvar, and CallFunction).

Any number of touchprobe events can be used in an SRL program, but only one event at a time. To activate a new event, the previous event must be deactivated, otherwise an error is triggered.

To use the touchprobe function, you have to activate it in the configuration menu "Additional functions" (→ 🗎 41) of the software module.

For further information on parameterizing the SRL commands in the RobotMonitor, refer to chapter "Touchprobe" (→ 🗎 127).

7.9.1 Activation/deactivation

The touchprobe function is not permanently activated. It can be activated, for example, at the start of the program or only at a certain point on the path of the robot. This prevents the triggering of the sensor in a wrong path segment.

A touchprobe event is activated by a registration using the "REG_TOUCHPROBE_EVENT" instruction in the robot program. The following setting options are available for the instruction:

- The trigger source is set using the *Source* parameter.
  – With the "InverterTouchprobe" setting, the touchprobe positions of the inverters are used. This requires a touchprobe sensor to be connected to all inverters of the robot. Additionally, all MultiMotion axes of the robot must be configured as described in chapter "Configuring "MultiMotion touchprobe"" (→ 🗎 87). The trigger position can be determined accurately (< 0.2 ms recording time).
  – With the setting "BoolVariable" as the *Source*, the digital input to which the sensor is connected must be mapped to an SRL Boolean variable in the IEC program. Additional settings in MOVISUITE® are not required. With this setting, the actual position of the inverter is used, and not the touchprobe position of the inverter. It is read in the cycle of the *HighPrio* task (Power/Progressive: cycle time ≥ 1 ms, Standard/Advanced: cycle time ≥ 5 ms).

    – When using the MOVIKIT® MultiAxisController, a simulated axis, or a virtual axis below the robot, the setting "BoolVariable" is required.

    – With the setting "BoolVariable" as the *Source*, you additionally parameterize the index of the Boolean variable in the *SourceBoolVar* parameter, and the edge that is to be triggered in the *Level* parameter.

- The *Mode* parameter is used to set the number of trigger events to be detected until deregistration. When set to "Single", one trigger is detected. When set to "Multiple", all triggers are detected until deregistration.
- The *MeasuringDirection* parameter is used to set the direction (x, y, or z) in which the sensor triggers a trigger. For example, if the height of a palletizing application is measured vertically along z direction, then set "z". The *MeasuringDirection* parameter also specifies the direction to which the distance of sensor-based positioning refers.

The touchprobe function can be activated for several path segments. Once activated, the touchprobe function remains active until ...

- The (touchprobe-) sensor is triggered in "Single" mode.
- The function is deactivated with the instruction "DEREG_TOUCHPROBE_EVENT.
- The motion pointer ("M" for "Motion") passes the "CONTINUE AFTER POSITIONING EVENT" instruction.
- The SRL program has ended or is stopped.

Configuring "MultiMotion touchprobe"

To use the touchprobe positions of the inverters as the source ("InverterTouchprobe" *source*) for the touchprobe function of MOVIKIT® Robotics, the touchprobe function of all single axes of the robot must be configured accordingly.

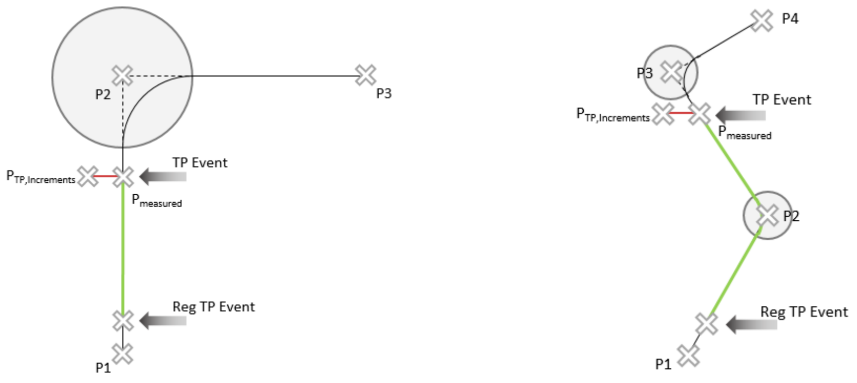
The touchprobe function of the inverters is activated in the configuration menu "Basic settings" of MOVIKIT® MultiMotion under "Functions used". The configuration of MOVIKIT® MultiMotion is then extended by the configuration menu described below. The following table provides notes on the setting options of this configuration menu with regard to using the function together with MOVIKIT® Robotics. Usually the default settings can be used. For a detailed description of the configuration menu, refer to the MOVIKIT® MultiMotion manual.

Parameter	Value
General	
Touchprobe source	Set "Inverter" as the touchprobe source.
Mode	In this case, the selection of the touchprobe mode is irrelevant because it is overwritten by the robot program.
Trigger	
Source	Source for triggering the trigger
Event	Selection for the type of edge used for triggering. This parameter cannot be set using the robot program. • Rising edge • Falling edge • Rising and falling edge
Sensor dead time rising edge	Optional setting of the dead time of the sensor in use for the rising edge at the trigger input. This time will be included in the calculation of the touchprobe event value.

7.9.2 Triggering and measuring

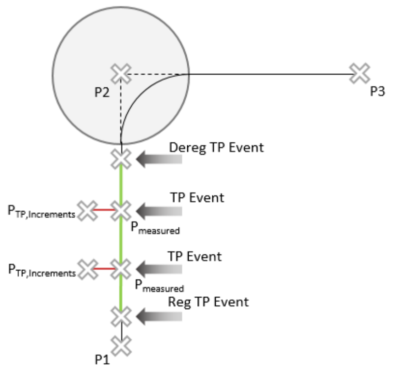
After triggering a touchprobe event, the positions are read, transformed, and projected onto the path. The determined Cartesian actual position is projected perpendicular to the *MeasuringDirection* onto the path of the robot. When using the "MEASURE" instruction, the projection point is output in the pose variable set there.

If the path segment into which is to be projected has no motion component in *MeasuringDirection*, an error is output. This is the case for example, if the *MeasuringDirection* is set to "z" but the robot moves along a straight line in x-y direction. If projection onto the current path segment is not spatially possible, an attempt is made to project the point onto one of the adjacent segments.



Path point	Description
P1 .. P4	Path points along the course of the path
Reg TP Event	Instruction for activating the touchprobe
TP event	Triggering the sensor or switching the BOOLEAN variable
$P_{TP,Increments}$	Actual position or touchprobe position of the inverters
$P_{measured}$	Measuring point projected onto the path

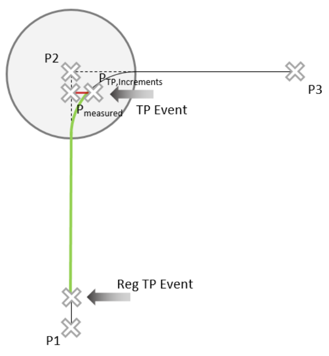
In the figures, the touchprobe is shown vertically in z-direction in "Single" *mode* with *MeasuringDirection*. The green area represents the segments in which the touchprobe function is activated. After triggering of the triggers, the measured point ($P_{TP,Increments}$) is projected onto the path ($P_{measured}$). The movement of the robot itself is not affected.



Path point	Description
P1 .. P3	Path points along the course of the path
Reg TP Event	Instruction for activating the touchprobe

Path point	Description
Dereg TP Event	Instruction for deactivating the touchprobe
TP event	Triggering the sensor or switching the BOOLEAN variable
$P_{TP,Increments}$	Actual position or touchprobe position of the inverters
$P_{measured}$	Measuring point projected onto the path

In "Multiple" *mode*, the touchprobe function remains active after a trigger is triggered (*MeasuringDirection* is in z-direction). If another trigger is triggered, the newly measured point is also projected onto the path, and $P_{measured}$ is overwritten. The event is active (green area) until a "DEREG Touchprobe_EVENT" instruction is interpreted, the program ends or is stopped.



Path point	Description
P1 .. P3	Path points along the course of the path
Reg TP Event	Instruction for activating the touchprobe
TP event	Triggering the sensor or switching the BOOLEAN variable
$P_{TP,Increments}$	Actual position or touchprobe position of the inverters
$P_{measured}$	Measuring point projected on path

If the touchprobe sensor is triggered within a blending curve, the point is projected onto the path that was interpolated without blending (figure: *MeasuringDirection* z-direction, "Single" *mode*).

7.9.3 Sensor-based positioning

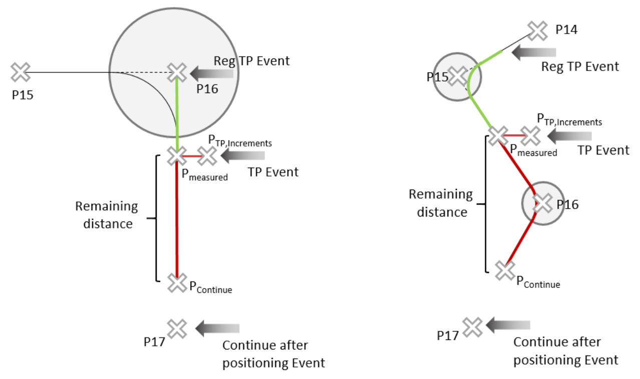
The "POSITIONING" instruction is available to initiate sensor-based positioning after a touchprobe event has been triggered. In the *RemainingDistance* parameter, you set the distance to be covered in *MeasuringDirection*.

Observe the following to use this instruction:

- The "POSITIONING" instruction is activated for a specified range (remaining distance travel range). To mark this area, it must be restricted with the instruction "CONTINUE AFTER POSITIONING EVENT".
- The travel range of the remaining distance must be completely known to calculate the remaining distance position. If the end of the range is not yet known when remaining distance travel is triggered, i.e. the program has not yet been interpreted up to the instruction and the program pointer has therefore not yet arrived at the instruction (e.g. due to a "WAIT" instruction), an error is output.
- The "CONTINUE AFTER POSITIONING EVENT" instruction must be executed after registering the touchprobe event. Otherwise an error is output.

- "POSITIONING" is always executed at the first trigger of the event ("Single" *mode*). If "Multiple" *mode* in conjunction with "POSITIONING" is set, the touchprobe function is still deactivated after starting sensor-based positioning. A warning informs the user about this.
- The starting point of sensor-based positioning is the measuring point projected onto the path.
- The set remaining distance (*RemainingDistance*) must be positive.

The remaining distance is calculated while the event is executed. The *RemainingDistance* specifies the offset to be covered in *MeasuringDirection*. The path itself, including its direction, is not changed by the *MeasuringDirection*, i.e. the robot continues along the programmed path.

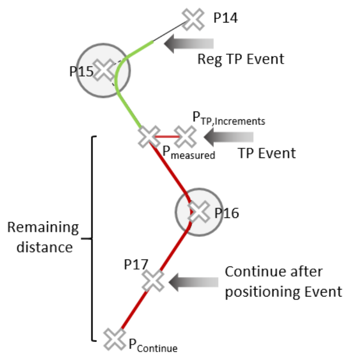


Path point	Description
P14-P17	Path points along the course of the path
Reg TP Event	Instruction for activating the touchprobe function
CONTINUE AFTER POSITIONING EVENT	Mark at which point the path should be continued after the "POSITIONING" instruction
TP event	Triggering the sensor or switching the BOOLEAN variable
$P_{TP,Increments}$	Actual position or touchprobe position of the inverters
$P_{measured}$	Measuring point projected onto the path
$P_{Continue}$	End point of sensor-based positioning
Remaining Distance	Length of the remaining distance [mm] in *MeasuringDirection*

The remaining distance results along the path segments of the remaining distance travel range (including their orientation) until the *RemainingDistance* is reached. The rotational degrees of freedom are interpolated true to path until the remaining distance is reached. Sensor-based positioning is also possible across several segments. The direction of the segments must deviate less than 90° from the *MeasuringDirection* so that there is a motion component in *MeasuringDirection*. When sensor-based positioning is performed, all path segments between the end of the remaining distance and the "CONTINUE AFTER POSITIONING EVENT" instruction are discarded.

The motion parameters set in the program for the current path segment are used for sensor-based positioning. If it is not possible to maintain the desired end point with these parameters (e.g. due to excessive speed when starting the remaining distance or a remaining distance that is too short), the motion parameters are increased up to maximally the emergency stop ramps. A warning informs the user about this. If it is not possible to maintain the set remaining distance despite emergency stop ramps, an error is output.

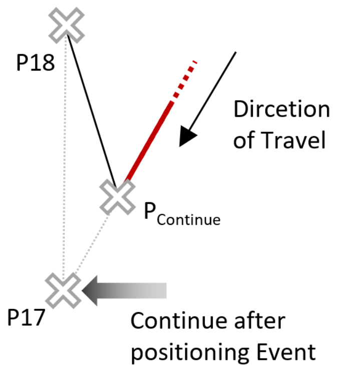
In the current software version, the movement is not accelerated during sensor-based positioning. This means that when triggering late, a very slow sensor-based positioning can occur shortly before reaching standstill. In future versions, it will be possible to accelerate also during sensor-based positioning.



Path point	Description
P14-P17	Path points along the course of the path
Reg TP Event	Instruction for activating the touchprobe function
CONTINUE AFTER POSITIONING EVENT	Mark at which point the program should be continued after the "POSITIONING" instruction
TP event	Triggering the sensor or switching the BOOLEAN variable
$P_{TP,Increments}$	Actual position or touchprobe position of the inverters
$P_{measured}$	Measuring point projected onto the path
$P_{Continue}$	End point of sensor-based positioning

Path point	Description
RemainingDistance	Length of the remaining distance [mm] in *MeasuringDirection*

If the set remaining distance is longer than the provided segments up to the "CONTINUE" mark, the last path segment is extended. In this case, the restriction applies that this segment has no rotation so the rotational degrees of freedom in the segment are not changed, otherwise the segment is not extended and an error is output.



Path point	Description
P17-P18	Path points along the course of the path
CONTINUE AFTER POSITIONING EVENT	Mark at which point the path should be continued after the "POSITIONING" instruction
$P_{Continue}$	End point of sensor-based positioning

After a "POSITIONING" instruction has been executed, further path segments may follow. However, these movement segments are not executed until positioning is completed. Next, the position after the "CONTINUE AFTER POSITIONING EVENT" instruction is approached.

7.9.4 Touchprobe as path event

The "REG_TOUCHPROBE_EVENT" and "DEREG_TOUCHPROBE_EVENT" instructions can be placed below a "REG PATH_EVENT" instruction, i.e. at a path event. Doing so lets you activate the touchprobe function on a specific path segment.

7.9.5 Using path events and touchprobe simultaneously

Path events whose stopping time has already expired will continue to be triggered independently of touchprobe positioning. However, as soon as touchprobe positioning is started, path events whose shifted reference point (*Distance* parameter) lies behind the end point of positioning are no longer triggered. This is why path events might not be triggered during touchprobe positioning.

If, for example, the vacuum of a suction gripper is to be switched on before reaching the gripping position, the path event must be parameterized in an area that lies before the end point of touchprobe positioning. There is no automatic shifting of path events with the end point of positioning.

29180252/EN – 03/2020

8 Control via the process controller

8.1 Process controller

After startup and robot programming, the software module is ready for operation. The software module can be controlled via the IEC user program on the MOVI-C® CONTROLLER by the PLC via the fieldbus, or by the operator via the RobotMonitor. These options for controlling the software module are summarized in this manual and referred to as process control or process controller.

8.2 Starting up the process controller

✓ The parameterization function test was successful. For further information, refer to chapter "Referencing axes and performing function test" (→ 🗎 48).

✓ A robot program has been created. For further information, refer to chapter "Robot program" (→ 🗎 99).

1. Program the "Process start" (→ 🗎 95).
2. Program the "Process sequence" (→ 🗎 95).
3. Program the Handling errors"" (→ 🗎 97).

8.3 Important status signals

Access

Access to the robot can be requested via the *Get Access Control* signal. If the request was successful, the *Control Active* signal is returned. For further information, refer to chapter "Access management" (→ 🗎 58).

Error

If errors or faults are present, remove the causes of error first. The error messages displayed in the RobotMonitor or IEC Editor can be used to determine the cause of error. A fault status can be resolved by setting the *Reset* signal.

Setpoints active

For controlling the robot, make sure *Setpoints Active* is set to "TRUE". If not, determine the cause:

1. Check if there is an error.
2. Check if "Enable" is set.
3. Check if access has been requested.

8.4 Prerequisites for movements of the robot

General

- Setpoints active
- Override > 0
- Prerequisites depending on the control type

Program mode

- The motion program includes motion commands for poses that do not correspond to the current pose.
- Program pause deactivated
- Program stop deactivated
- Rising edge of program start

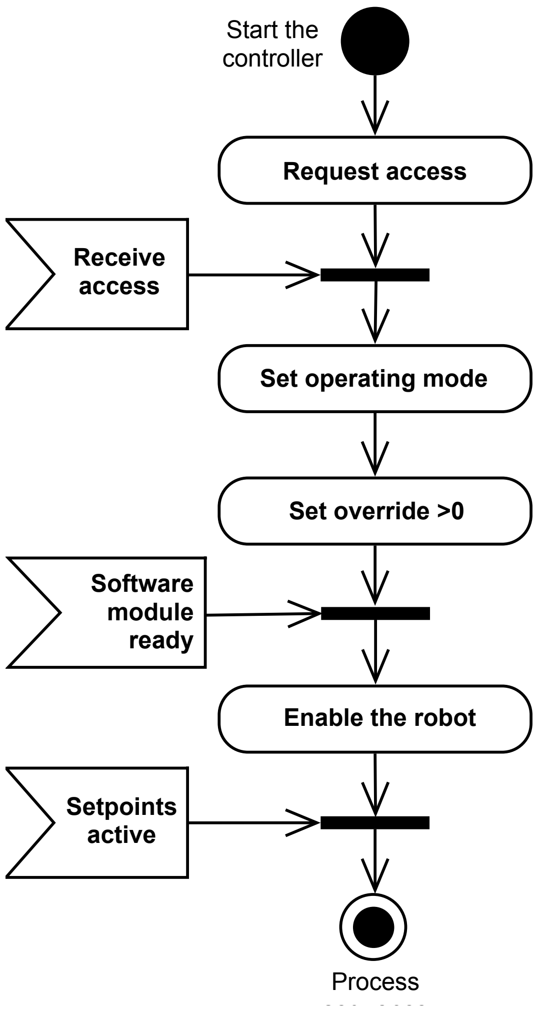
  The rising edge must follow at least one cycle after the rising edge of "Setpoint active" and after the falling edges of program pause and program stop to enable the rising edge to start the program.
- In "Manual high speed" operating mode, the program start must remain activated until the entire program has finished. For further information, refer to chapter "Program mode" (→ 🗎 55).

Jog mode

- In jog mode, at least one jog signal must be activated (positive or negative).

8.5 Process start

After switching on the system, the program start in the process controller may appear as follows. There may also be other requests depending on the application.



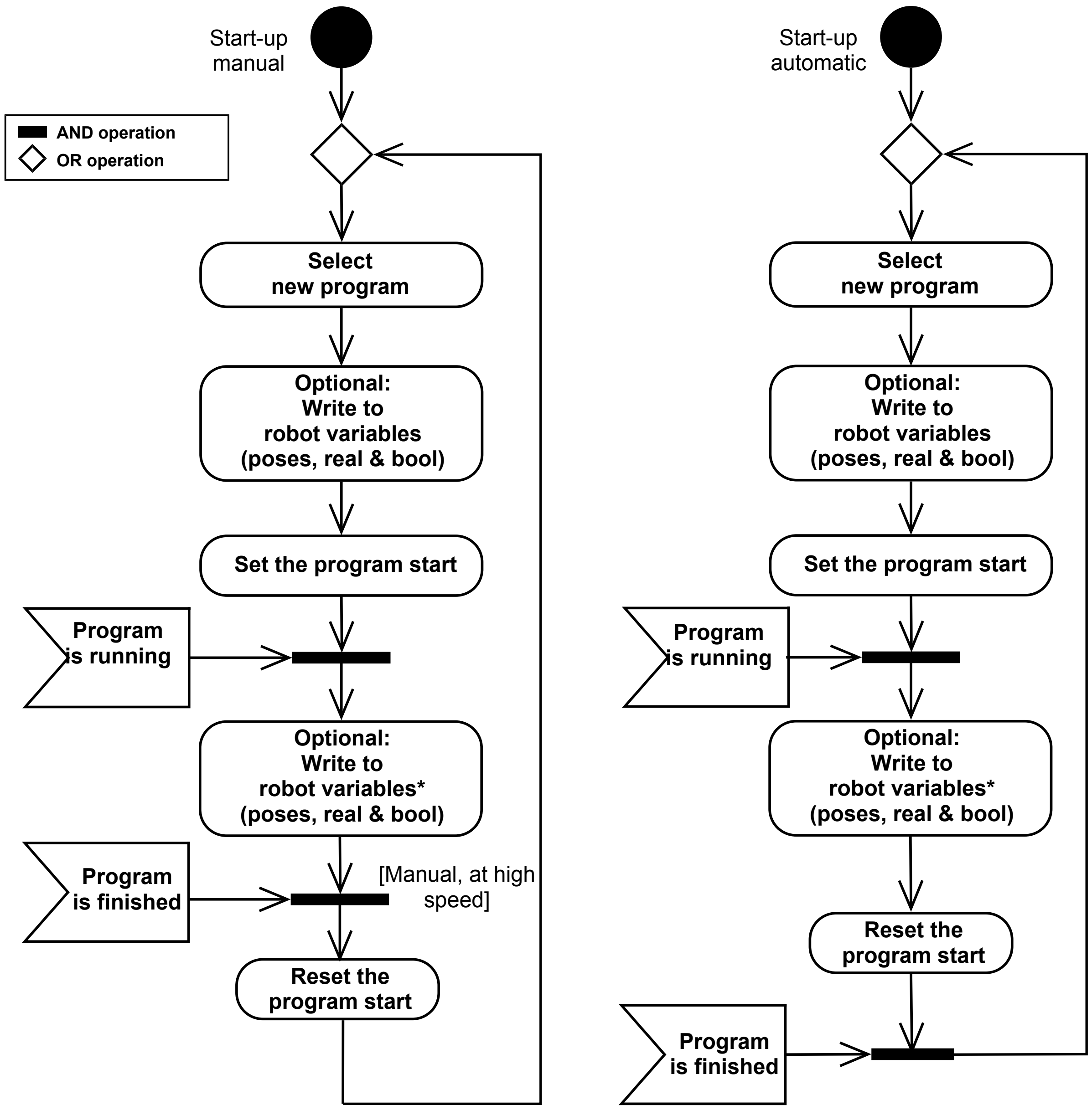
9007222588577419

8.6 Process sequence

Programs may be started as follows:

- First, specify the program number. If needed, re-parameterize the robot program variables (path poses; REAL and BOOL variables).
- Once the program is initialized, start it using a program start.
- In "Manual high speed" operating mode, the start button must remain activated until the entire program has finished processing.
- For each program sequence, you can also connect the wait signals while the program is being processed.
- When the system detects that the program is complete, the sequence starts again from the beginning.

The following flow diagram shows how you can continuously run programs. Before the displayed sequence can be performed, make sure the requirements for carrying out a motion in program mode are fulfilled; see chapter "Prerequisites for movements of the robot" (→ 🗎 94).



27021621098062219

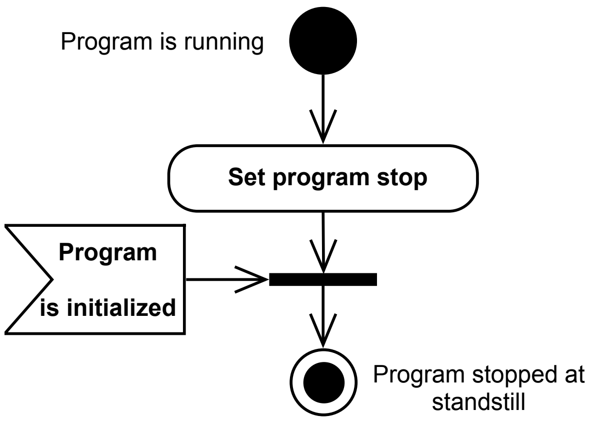
Comments

- The new values take effect once the robot executes a command that uses the variable. See chapter "Executing the robot program" (→ 🗎 111).
- The SRL command *CallFunction* can be used to write the robot variables. The *CallFunction* executes the contained IEC code clearly defined and consistent in the program flow at the position where the *CallFunction* is located in the SRL program. For example, MotionSets can also be described in the *CallFunction* (using a method in the user interface).

29180252/EN – 03/2020

8.7 Process stop

If the current program is to be stopped and started again from the beginning of the program, program stop must be executed.



31369861771

8.8 Handling errors

You can use the fault signal to determine if MOVIKIT® Robotics is in fault status. The fault can be caused by the controller or can be signaled by assigned inverters. You can detect if there is an inverter fault from the fault outputs of the single axes. In the event of a fault or error, the setpoint active signal signal is set to "FALSE".

When the software module has a fault status, the configured emergency stop ramps are used to automatically decelerate and brake. Once the cause of fault has been removed and the fault has been reset, you may continue. The fault is reset using the *Reset* signal.

To help find the cause of fault, a unique fault number and an error text are issued in the *MessageID* signal. In the event of an inverter fault, you can find more information regarding the fault number in the device documentation.

8.8.1 Limitation of the work envelope

Each robot has a work envelope within which it may move. The work envelope is a result of the kinematic model and of the limitations set by the software. These limitations are parameterized by the user with knowledge of both the environment and the mechanical components of the real robot.

An exceeded software limitation will automatically trigger an emergency stop and an error message (0x7e60). After a subsequent error reset, the robot can be moved in jog mode. The output signals (IEC: *Basic.Out.xOutOfWorkspace*, PD PO 7:1) and a warning (0x17e60) also indicate that the robot is outside the work envelope. As soon as the robot is inside the work envelope, the instructions no longer apply and a program can be started again.

The following software limitations can be parameterized for the work envelope:

Joint axis limitations

- Limit values for rotary joints and prismatic joints actuated by a drive. These values may result, for example, from the following underlying conditions:
  - Limited mobility of the joint

– Winding of cables

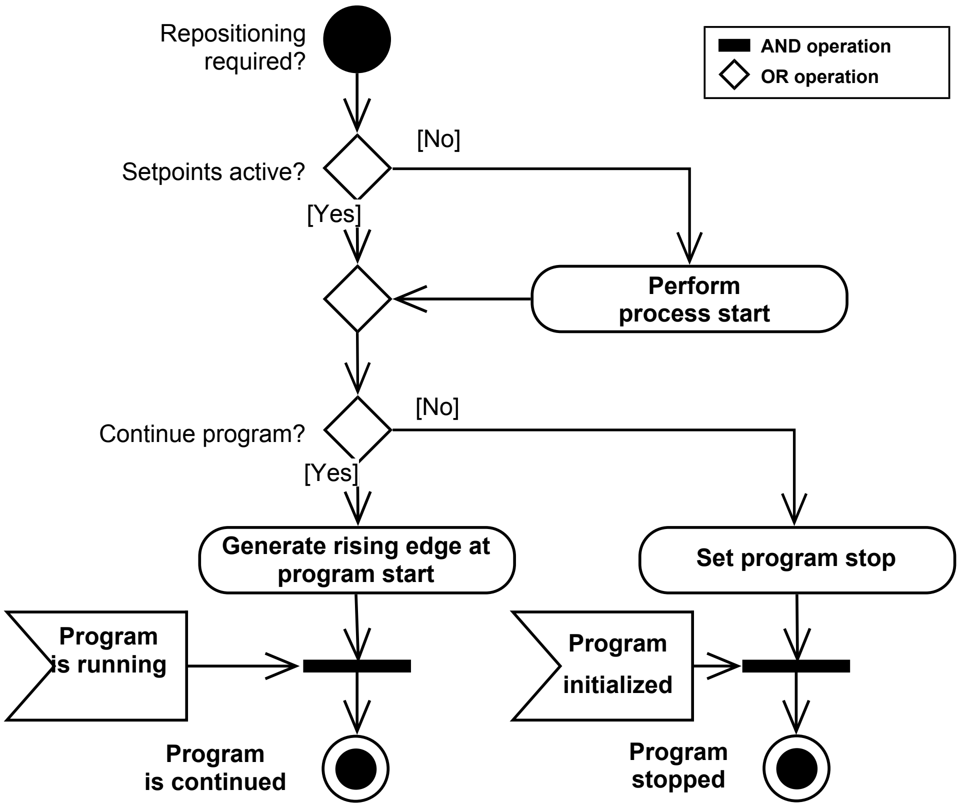
– End positions of the linear guidance system

Cartesian limits

- Limit values for the Cartesian coordinates of the tool center point and the tool orientation. Spatially speaking, the 3 translational limitations set up a square block within which the tool may move. The rotary limitation also restricts the permitted orientation of the tool.

8.9 Repositioning required

If repositioning is reported as required, there are two ways to handle this. Either cancel the existing program using program stop and start a new one (see previous chapter) or start repositioning using program start and continue the program.



31381916043

29180252/EN – 03/2020

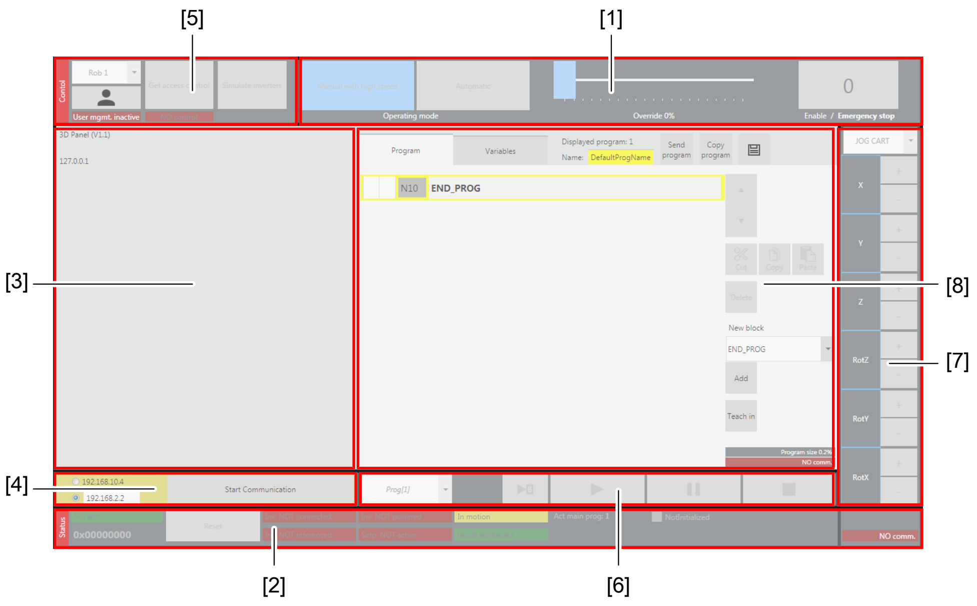
9 Robot program

This chapter describes the signals of the RobotMonitor and how to create robot programs using the SEW Robot Language (SRL). The sequence in which the signals are to be controlled is described in chapter "Control via the process controller" (→ 🗎 93).

9.1 Requirements

- The IEC project with the software module has been downloaded to the MOVI-C® CONTROLLER and started. For further information, refer to chapter "Generating an IEC project" (→ 🗎 42).
- The RobotMonitor has been "started" (→ 🗎 45) and a connection has been "established" (→ 🗎 46).

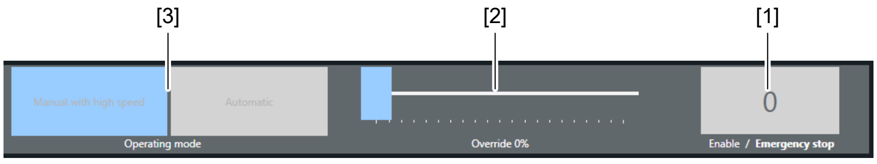
9.2 User interface



31688038411

No.	Description
[1]	"General robot control" (→ 🗎 100)
[2]	"General status and diagnostic messages of the robot" (→ 🗎 100)
[3]	"3D simulation of the robot" (→ 🗎 104)
[4]	Controlling the "communication connection" (→ 🗎 46)
[5]	"Access management/user management" (→ 🗎 46)
[6]	"Program status and control of program operation" (→ 🗎 105)
[7]	"Jog mode control" (→ 🗎 105)
[8]	Tab for "creating the robot program" (→ 🗎 107)

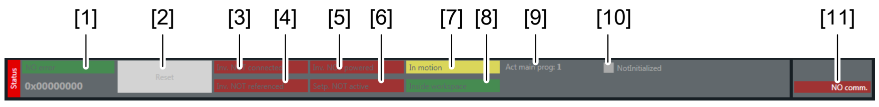
9.2.1 General robot control



31688348555

No.	Description	IEC name
[1]	Emergency stop (Enable/EmergencyS-top)	Basic.IN.xEnable_EmergencyStop
[2]	Scaling of the speed in percent (over-ride)	Basic.IN.usiOverride
[3]	Operating mode	Basic.IN.eOperatingMode

9.2.2 General status and diagnostic messages of the robot



31688374923

No.	Description	IEC name
[1]	Fault status	xError
	Right underneath: Message ID	udiMessageID
[2]	Reset fault	xReset
[3]	Connection status	Inverter.OUT.xConnected
[4]	Status or referencing	Inverter.OUT.xReferenced
[5]	Enable state	Inverter.OUT.xPowered
[6]	Status of setpoint processing	Inverter.OUT.xSetpointActive
[7]	Status of motion	Basic.OUT.xSetpointStandstill
[8]	Work envelope left	Basic.OUT.xOutOfWorkspace
[9]	Current program number	Prg.OUT.uiProgramNumber
[10]	Program state	Prg.OUT.eProgramState
[11]	Connection status of the RobotMonitor	-

9.2.3 User management

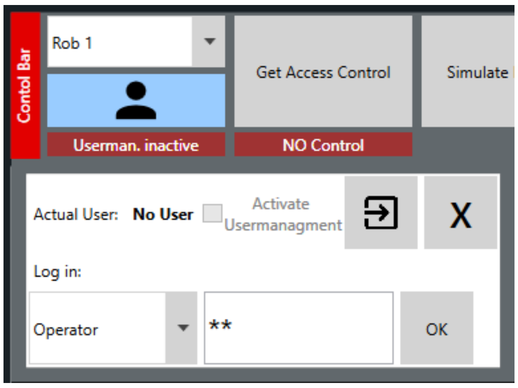
INFORMATION

User management is disabled by default. Users need not necessarily be created and configured. You can skip this step if you do not want to use the RobotMonitor on the machine.

Opening user management

✓ The RobotMonitor is open.

1. To open user management, click .
   ⇨ User management opens.



31659964171

Logging in as user

✓ User management is open.

1. Choose the appropriate user from the "Log in" choice box.
2. In the edit box next to the choice box, enter the matching password.
3. Click [OK] to confirm your entry.
   ⇨ The user name is displayed in the "Actual user" field.
4. To close user management, click the [X] button.

Logging off as user

✓ User management is open.

✓ You are logged in as user.

1. To log off the user, click .
   ⇨ *No User* is displayed in the "Actual user" field.
2. To close user management, click the [X] button.

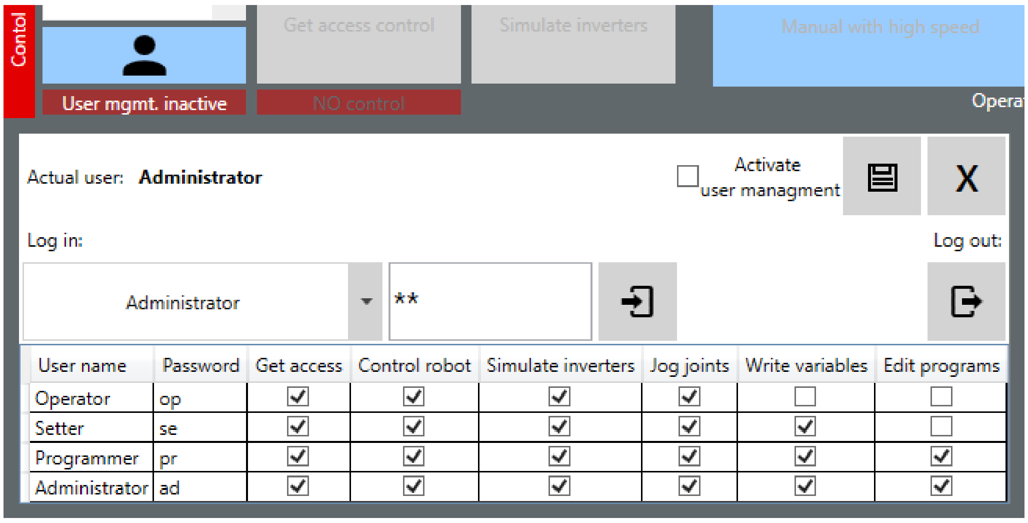
Enabling user management

INFORMATION

User management is disabled by default. The activation state is indicated underneath the button for opening user management. Only administrators can enable and disable user management.

✓ User management is open.

1. Log in to user management as administrator (default password: "ad").
   ⇨ *Administrator* is displayed in the "Actual user" field.
   ⇨ A list of all created users is displayed.



Actual user: **Administrator**

☐ Activate user managment

Log in: Administrator **

Log out:

User name	Password	Get access	Control robot	Simulate inverters	Jog joints	Write variables	Edit programs
Operator	op	✓	✓	✓	✓	☐	☐
Setter	se	✓	✓	✓	✓	✓	☐
Programmer	pr	✓	✓	✓	✓	✓	✓
Administrator	ad	✓	✓	✓	✓	✓	✓

9007225108439691

2. Enable the "Activate User Management" check box.
   ⇨ User management is now enabled. The state underneath the button for opening user management now indicates *User man. active*.
3. Log off from user management. See chapter "Logging off as user" (→ 🗎 101).
4. To close user management, click the [X] button.

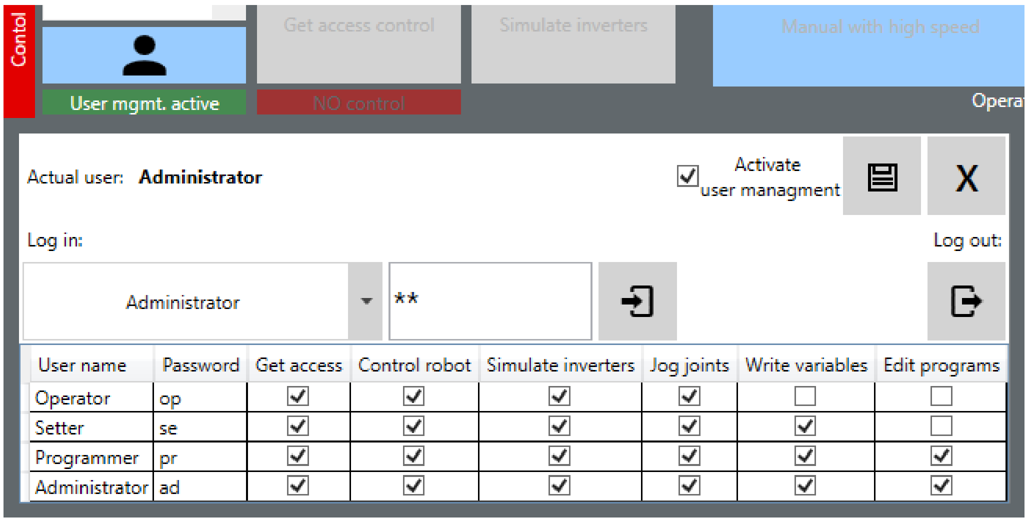
Configuring users

INFORMATION

Only administrators can configure passwords and access rights.

✓ User management is open and enabled.

1. Log in to user management as administrator (default password: "ad").
   ⇨ *Administrator* is displayed in the "Actual user" field.
   ⇨ A list of all created users is displayed.



Actual user: **Administrator**

Log in: Administrator, **

User name	Password	Get access	Control robot	Simulate inverters	Jog joints	Write variables	Edit programs
Operator	op	☑	☑	☑	☑	☐	☐
Setter	se	☑	☑	☑	☑	☑	☐
Programmer	pr	☑	☑	☑	☑	☑	☑
Administrator	ad	☑	☑	☑	☑	☑	☑

9007225112801803

2. Enable or disable the access rights of individual users by means of the check boxes. The following access rights are available:

Check box	Function
GetAccess	Request access
ControlRobot	Control a robot in jog mode and automatic mode
WriteVariables	Change program variables of a robot
EditPrograms	Change robot programs
SimulateInverters	Switches the inverters connected to the robot to simulation mode (requires "ControlRobot" permission)
JogJoints	Jogs the individual robot joints (requires "ControlRobot" permission)

3. To save the configuration, click 🖫.
   ⇨ The configured access rights for the users now take effect.
4. To close user management, click the [X] button.

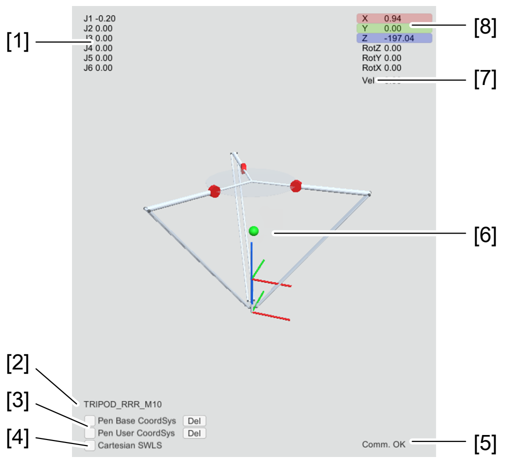
29180252/EN – 03/2020

9.2.4 3D simulation

The current position of the robots is displayed in the 3D simulation integrated in the RobotMonitor and the path of the TCP can be recorded using the "pen" tool.

You can influence the view in 3D simulation by the following actions:

- Hold the left mouse button and move the mouse: Rotate the view around the displayed green reference point.
- Hold the right mouse button and move the mouse: Move the view in the plane of the screen, i.e. the green reference point is moved. This also causes the view to be rotated with the left mouse button around the reference point you have now moved.
- Turn the mouse wheel: Zoom (increase and decrease) the view. You can zoom to the green reference point at most. If you wish to observe objects located behind the green reference point from close up, first move the reference point to that location with the right mouse button.

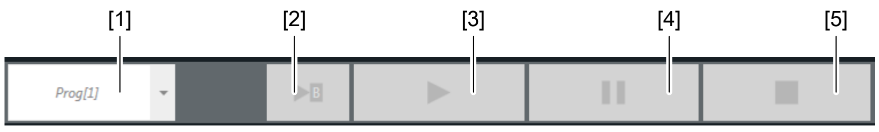


31658442891

No.	Description	IEC name
[1]	Coordinates of the joints	Basic.OUT.alrKinematicJoint
[2]	Output of current kinematic model	
[3]	Controls for operating the pen tool. You can activate pens that trace the path of the TCP. "Pen Base CoordSys" draws the path relative to the static base coordinate system of the robot. "Pen User CoordSys" draws the path relative to the possibly moving coordinate system of the robot, i.e. the drawn path moves with the coordinate system. You can delete the drawn lines using the respective "Del" buttons.	
[4]	Control to show or hide the Cartesian work envelope of the robot.	
[5]	Output whether the 3D simulation has a communication connection to the controller	
[6]	3D simulation of the kinematic model	
[7]	Output of the current translatory speed of the TCP	
[8]	Coordinates of the base	Basic.OUT.alrBase

29180252/EN – 03/2020

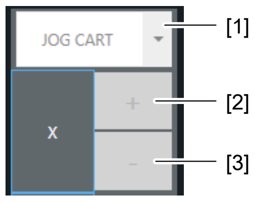
9.2.5 Program status and control of program operation



31688402059

No.	Description	IEC name
[1]	Drop-down list for the programs	Prg.IN.uiProgramNumber
[2]	Step mode	PRG.IN.eMode
[3]	Start	Prg.IN.xStart
[4]	Pause	Prg.IN.xPause
[5]	Stop	Prg.IN.xPause

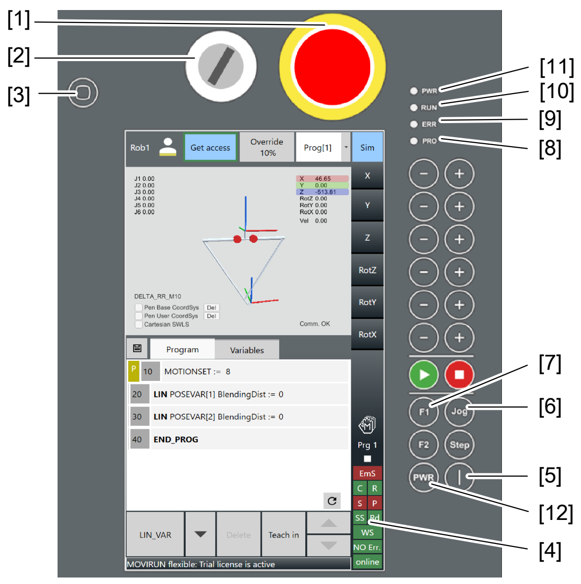
9.2.6 Jog mode control



31688415627

No.	Description	IEC name
[1]	Drop-down list of the JOG coordinate system	Jog.IN.Kinematic.eCoordin-ateSystem
[2]	Plus button for jog mode	Jog.IN.Kinematic.axPositive
[3]	Minus button for jog mode	Jog.IN.Kinematic.axNegative

9.2.7 Operator panel



31658546955

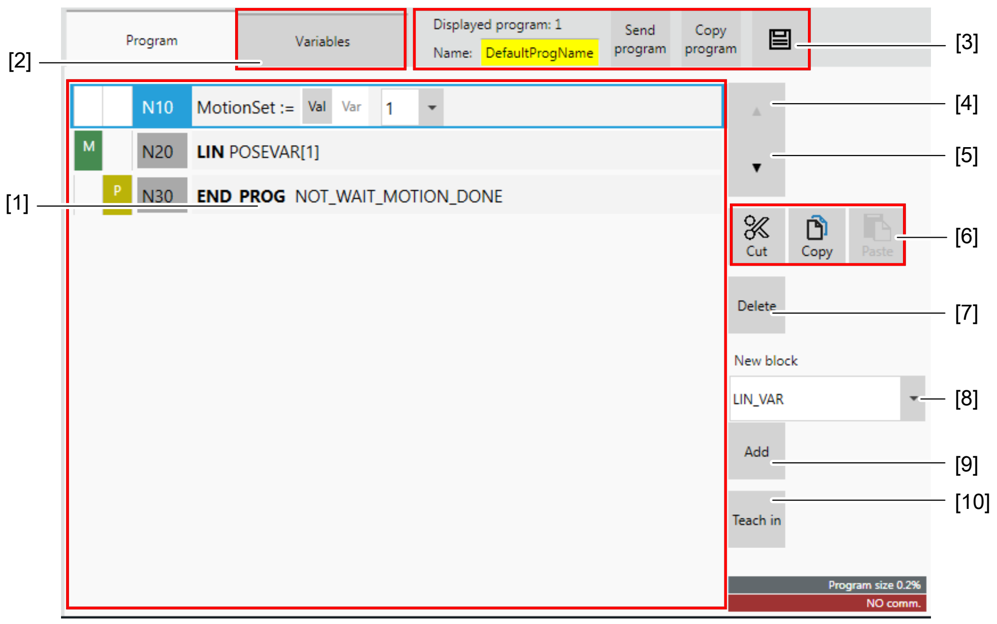
No.	Description
[1]	Emergency stop switch of the operator panel – switch must be wired externally. For additional information, refer to the documentation for the operator panel.
[2]	Key switch of the operator panel for switching the operating mode of the robot (operator mode selection switch) – switch must be wired to the MOVI-C® CONTROLLER and, if applicable, to the safety controller. For more detailed information, refer to the documentation of the operator panel and to the chapter "Using external selector switch for operating modes" (→ 🗎 159).
[3]	Button for switching between the following views of the RobotMonitor: – Split screen between 3D simulation and program display – Full screen 3D simulation – Full screen program display
[4]	Field with status information about the robot Touching the field opens a detailed status field.
[5]	Button for switching the operator panel on and off The action (shutdown, reboot, etc.) to be performed when the button is pressed can be set in the system settings of the operator panel. For more details, refer to the documentation of the operator panel.
[6]	Button for switching the JOG coordinate system
[7]	Free buttons for evaluation in the IEC program – for more information, refer to the chapter "Using free function keys of the operator panel" (→ 🗎 159).
[8]	Status LED: Program operation active
[9]	Status LED: Fault status
[10]	Status LED: Connection between RobotMonitor and MOVI-C® CONTROLLER
[11]	Status LED: Operator panel is energized
[12]	PWR: Button for enabling the robot

9.3 Creating the robot program

You can create robot programs using the editor integrated into the "Program" tab.

INFORMATION

Data loss of unsaved changes in the robot program. Be sure to save your robot program before generating the code, downloading the configuration in MOVISUITE, or restarting the controller.



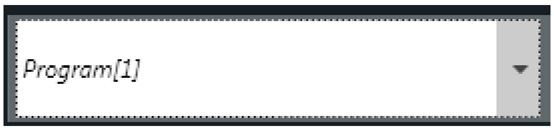
31688624139

No.	Description
[1]	Lines added to the robot program
[2]	Tab for editing "variables" (→ 🗎 130)
[3]	Options to "save" (→ 🗎 110)/"copy" (→ 🗎 109) the program
[4]	Move command up one line
[5]	Move command down one line
[6]	Cut/copy/paste command **Note:** You can also perform actions using the keyboard shortcuts Ctrl+x, Ctrl +c, and Ctrl+v.
[7]	Delete selected command
[8]	Selection of block to be added
[9]	Add selected block
[10]	Save current position (teach)

9.3.1 Creating a new program

✓ The visible program is finished or initialized. See chapter "Program mode" (→ 🗎 55).

1. Switch to the "Program" tab.
2. Change from the active program to the new program.



20621267083

⇨ The program is displayed.

3. Enter a name for the new program.
4. Insert the motion commands, assignments, and control flow elements by selecting the appropriate value from the "New Block" drop-down list and then clicking the [Add] button. Repeat this process for all motion commands to be included in your program. Refer to the subsequent chapters for further information on the commands.
5. Insert an end of program. Select the value "END_PROG" from the "New Block" drop-down list. Click the [Add] button.
6. To permanently save the program and the robot program variables on the memory card, click the save icon.

9.3.2 Editing an existing program

✓ The displayed program is finished or initialized. See chapter "Program mode" (→ 🗎 55).

1. Go to the "Program" tab.
2. Switch to the program you want to edit.

   ⇨ The program is displayed.

3. Add the required commands: Select the line below which you would like to insert the new command. Select the appropriate value from the "New Block" drop-down list. Click the [Add] button.
4. Configure the selected line.

   ⇨ You can move the selected line up or down using the arrow keys.

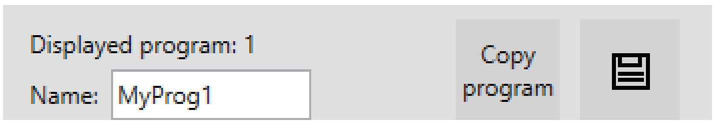
   ⇨ You can delete selected lines using the [Delete] button.

5. To permanently save the program and the robot program variables on the memory card, click the save icon.

9.3.3 Copying an existing program

✓ The displayed program is finished or initialized. See chapter "Program mode" (→ 🗎 55).

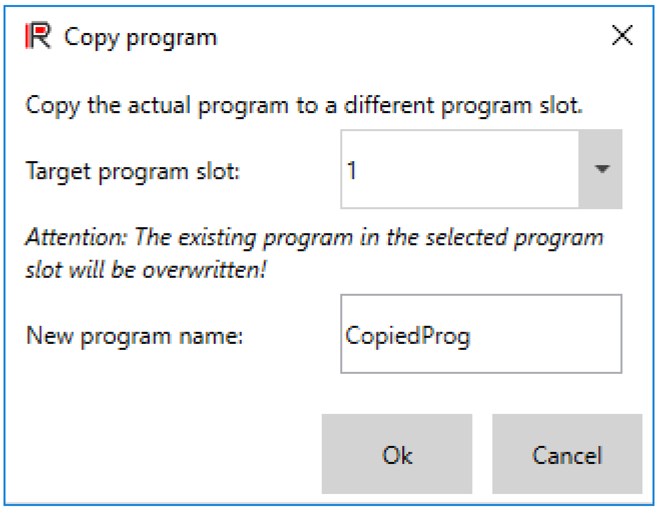
1. Go to the "Program" tab.
2. Switch to the program you want to copy.
   ⇨ The program is displayed.
3. Click the [Copy program] button.



Displayed program: 1
Name: MyProg1

18014423429246987

⇨ The "Copy program" dialog opens.



Copy program

Copy the actual program to a different program slot.

Target program slot: 1

Attention: The existing program in the selected program slot will be overwritten!

New program name: CopiedProg

31368176139

4. In the "Target program slot" choice box, select the program number to which the currently displayed program is to be copied. The existing program with this number will be overwritten.
5. Enter a name for the new program.
6. Confirm copying by clicking [OK].
   ⇨ The program is copied to the new number.

9.3.4 Adding commands

1. Select the line below which you would like to insert the new command.
2. Select the appropriate value from the "New Block" drop-down list.
3. Click the [Add] button.
4. Parameterize the command, if necessary. For further information, refer to the following chapters.

9.3.5 Loading a standard program

INFORMATION

The generated standard program overwrites the currently displayed program and the variable names.

Clicking the [Load standard program] button lets you create a standard robot program.

A dialog opens in which you can select how many standard path segments should be included in the created program. After confirming the dialog with [OK], the standard program is automatically generated and the names of the variables used in it are set.

Operating principle

A standard program contains a certain number of standard path segments. These segments can be parameterized using robot program variables. In this way you can operate the program externally through process control, e.g. via the fieldbus interface or the IEC user interface of the software module.

Each standard path segment includes the following parameters, each of which can be specified via variables:

- Target pose of the path segment
- Blending distance to the path segment
- Number of the MotionSet used for the path segment
- End signal to indicate that the program should be ended after this path segment
- Wait signal to cause the robot not to execute the path segment until the wait signal is removed

32232154379

INFORMATION

For the variables to be effective, they must be set when the program is started. Also wait signals must be set from the beginning so that the robot pauses at the desired position. Wait signals can be reset during program execution.

9.4 Saving the robot program

INFORMATION

If you do not save programs and variables, the changes made are lost and cannot be restored when resetting or rebooting the controller, when "generating the IEC project" (→ 🗎 42), or when updating the configuration data.

1. Click the [Save] button (diskette icon).

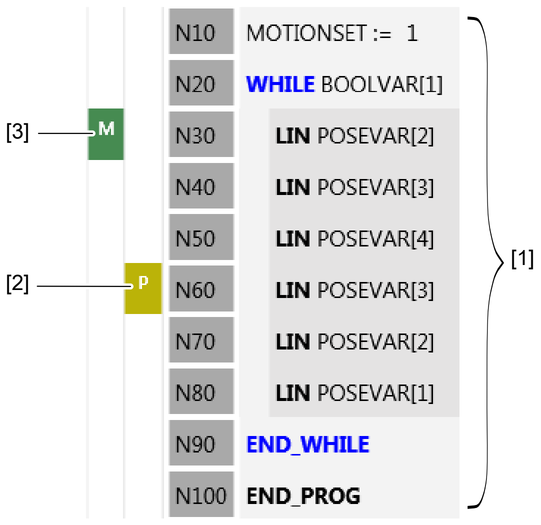
⇨ All programs and variables including motion sets of the robot are stored permanently. This means that the created programs and variables are reloaded after switching off or on the MOVI-C® CONTROLLER or after resetting the control program. The data is saved in files on the memory card of the MOVI-C® CONTROLLER and are read-in again when the MOVI-C® CONTROLLER is started again.

9.5 Executing the robot program

✓ A robot program has been created. See chapter "Creating the robot program" (→ 🗎 107).

1. Go to the "Program" tab.
2. Switch to the program to be executed.
   ⇨ The program is displayed.
3. Start the robot program by setting the "start" signal. For further information on program execution, refer to chapter "Program mode" (→ 🗎 55).

⇨ Execution of the robot program starts:



```
N10   MOTIONSET :=  1
N20   WHILE BOOLVAR[1]
N30     LIN POSEVAR[2]
N40     LIN POSEVAR[3]
N50     LIN POSEVAR[4]
N60     LIN POSEVAR[3]
N70     LIN POSEVAR[2]
N80     LIN POSEVAR[1]
N90   END_WHILE
N100  END_PROG
```

18014423429273995

[1]	Robot program commands
[2]	Program indicator ("P" for "program") Indicates the command that is currently being processed.
[3]	Motion pointer ("M" for "motion") Indicates which motion the motion control is currently processing, that is, along which path segment the robot is currently traveling.

INFORMATION

When the program is started, the "basic state" (→ 🗎 56) is established.

29180252/EN – 03/2020

9.6 Teaching poses

The robot monitor offers a "teach" function. Poses used in the robot program can be applied to the robot program (explicit teaching) or to the pose variable (variable teaching). In both cases, the robot program can then travel to that pose.

INFORMATION

The specified "Default units" (→ 🗎 21) apply.

9.6.1 Teaching a pose variable

The teach function always writes the pose which the robot is in to the desired pose variable. The following options are available:

Teaching a pose variable in the list of variables

1. Switch to the "Variables" tab.
2. From the "Variables" tab, switch to the tab of the desired pose variable.
3. Move the robot to the desired pose using jog mode.
4. From the list, select the pose variable you wish to apply.
5. Click [ActPos to Selected PosVar].
   - ⇨ The current pose of the robot is written to the selected variable. The changes take effect immediately.
   - ⇨ You can now use the pose in the robot program. For further information, refer to chapter "Adding a linear segment with a pose variable" (→ 🗎 113).
   - ⇨ You can change the variable if necessary. For further information, refer to chapter "Editing robot program variables" (→ 🗎 130).

Teaching a pose variable in the robot program

✓ A linear segment with pose variables exists in the robot program to be processed.

1. Switch to the "Program" tab.
2. Move the robot to the required pose using jog mode.
3. Select the linear segment with the pose variable you wish to apply.
4. From the drop-down list of the linear segment, select the pose variable that is to be written to.
5. Click [Edit].
6. Click [TeachIn].
   - ⇨ The current pose of the robot is written to the selected variable. The changes take effect immediately.
   - ⇨ You can change the command if necessary. For further information, refer to chapter "Adding a linear segment with a pose variable" (→ 🗎 113).

9.6.2 Explicit teaching including inserting a linear segment

1. Move the robot to the desired pose using jog mode.

2. Select the line below which you would like to insert the linear segment.
3. Click [TeachIn]
   - ⇨ A linear segment is inserted, and the current pose of the robot is applied. The changes take effect immediately.
   - ⇨ You can change the command if necessary. For further information, refer to chapter "Adding a linear segment with a pose variable" (→ 🗎 113).

9.7 Motion commands

Motion commands are commands to move to a position.

INFORMATION

The specified "Default units" (→ 🗎 21) apply.

9.7.1 Linear segments

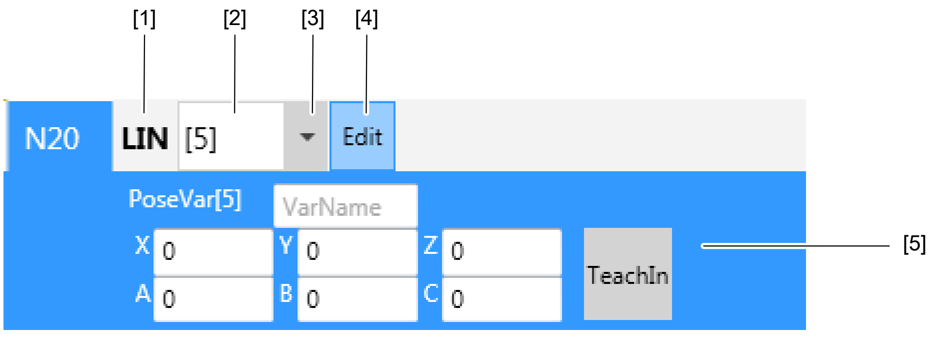
Commands for approaching a target pose on a straight line with previous blending. The target coordinates and the blending distance can be explicitly specified in the robot program or via IEC variables.

LIN BlendingDist target pose = blending distance

Adding a linear segment with a pose variable

A linear segment in which the target pose is specified via a robot program variable.

"New Block" value: "LIN_VAR".



18014423319144459

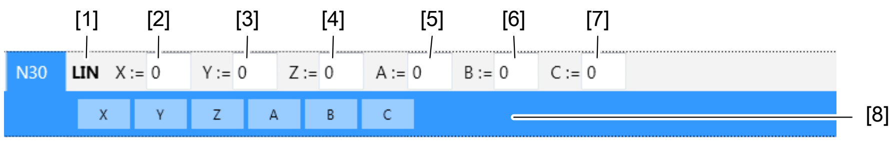
[1]	Identifier for a linear segment
[2]	Target pose
[3]	Drop-down list for replacing the target pose
[4]	Menu for changing the name and the coordinates of the target pose
[5]	Name and coordinates of the target pose

29180252/EN – 03/2020

Adding a linear segment with an explicit coordinate assignment

A linear segment in which the target pose is explicitly specified in the robot program.

"New Block" value: "LIN_EXPLICIT".



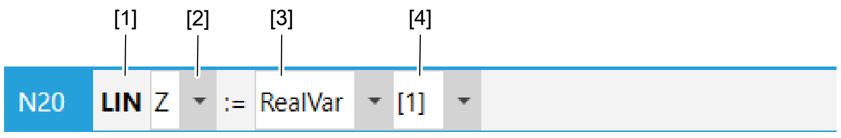
18014423320833035

[1]	Identifier for a linear segment
[2]	X coordinate of the target pose
[3]	Y coordinate of the target pose
[4]	Z coordinate of the target pose
[5]	Coordinate of the target pose for the rotation around Z
[6]	Coordinate of the target pose for the rotation around Y
[7]	Coordinate of the target pose for the rotation around X
[8]	Activate/deactivate individual coordinates If deactivated, the value remains the same as the value assigned to the previous motion command.

Adding a linear segment with single coordinate travel

Linear segment in which movement is to be carried out in direction of one coordinate axis only. The direction of movement can be selected. An explicit value, a REAL variable, or the coordinate of a POSE variable can be used as the target coordinate.

"New Block" value: "LIN_SINGLE_COORD".



9007228307326475

[1]	Identifier for a linear segment
[2]	Selection of the coordinate to be changed.
[3]	Selection whether a direct value, a REAL variable or a coordinate of a POSE variable is to be used.
[4]	Selection of the variable for the target coordinate

29180252/EN – 03/2020

9.8 Assignments

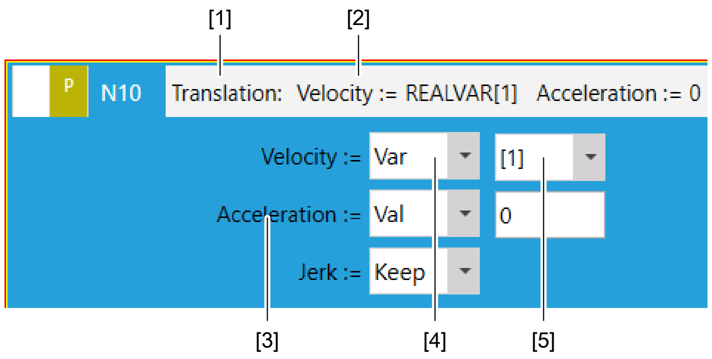
Assignments are commands that are effective until the command is called again with another parameterization.

9.8.1 Setting translatory motion properties

Command that specifies the speed, acceleration, deceleration, and jerk of subsequent translatory motion commands. Not all assignments may be set to "Keep". At least one assignment ("Value" or "Variable") must be made, else the program is not valid.

The set values are valid until they are changed again with a new call of SET_MOTION_PARA or until all motion parameters are overwritten by a complete motion parameter set with the command SET_MOTIONSET. The values within the defined motion parameter sets are not changed by this command.

"New Block" value: "SET_MOTION_PARA"



Translation: Velocity := REALVAR[1] Acceleration := 0

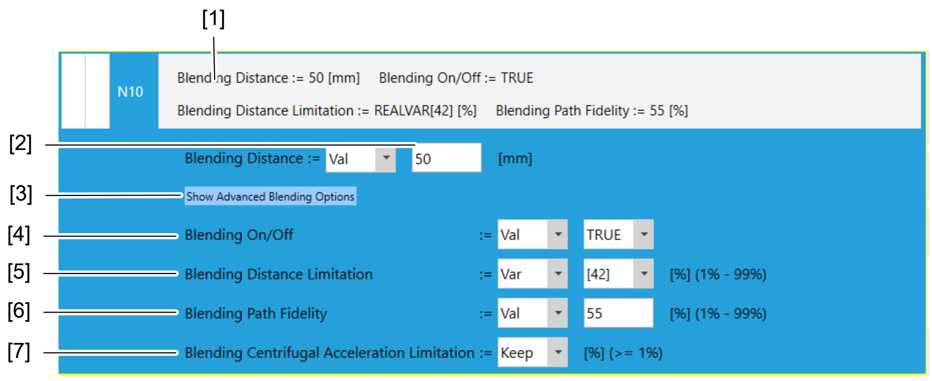
Velocity := Var [1]
Acceleration := Val 0
Jerk := Keep

31658896395

[1] Identifier for an assignment of translatory motion parameters
[2] Compact display of the assignment selection
[3] Property of the translatory motion to be written
[4] Selection of the assignment type (Var = SRL variable, Val = default value, Keep = no assignment is to be made, the previous value of the property is retained)
[5] Value of the assignment (Var = choice box with the index of the available SRL variables, Val = edit box for absolute values with the type floating point)

9.8.2 Setting parameters for blending

"New Block" value: "SET_BLENDING_PARA"



N10
Blending Distance := 50 [mm]   Blending On/Off := TRUE
Blending Distance Limitation := REALVAR[42] [%]   Blending Path Fidelity := 55 [%]

Blending Distance :=	Val	50	[mm]	
Show Advanced Blending Options				
Blending On/Off :=	Val	TRUE		
Blending Distance Limitation :=	Var	[42]	[%] (1% - 99%)	
Blending Path Fidelity :=	Val	55	[%] (1% - 99%)	
Blending Centrifugal Acceleration Limitation :=	Keep	[%] (>= 1%)		

18014423431719307

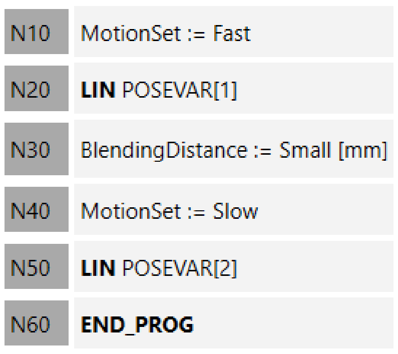
[1]	Parameters set for blending
[2]	Distance to the end point of the segment from which the new segment is to be blended. The blending distance 0 leads to an exact stop.
[3]	Show special parameters for blending
[4]	Enable/disable blending. When disabled, an exact stop is achieved.
[5]	Percentage of the length of the segment to be blended, as a limit of the blending distance limit, by default set to 50%.
[6]	Deviation from circular blending, by default set to 1% maximum jerk limitation at transition, 99% almost circular.
[7]	Limitation of the centrifugal acceleration, scaled as a percentage to the minimum of the translational accelerations of the adjacent path segments, by default set to 100%.

Blending between path segments

The following example illustrates blending from one motion segment to another motion segment:

- Motion segment 1: MotionSet fast, LIN, target position POSEVAR[1]
- Motion segment 2: BlendingDistance small, MotionSet slow, LIN, target position POSEVAR[2]

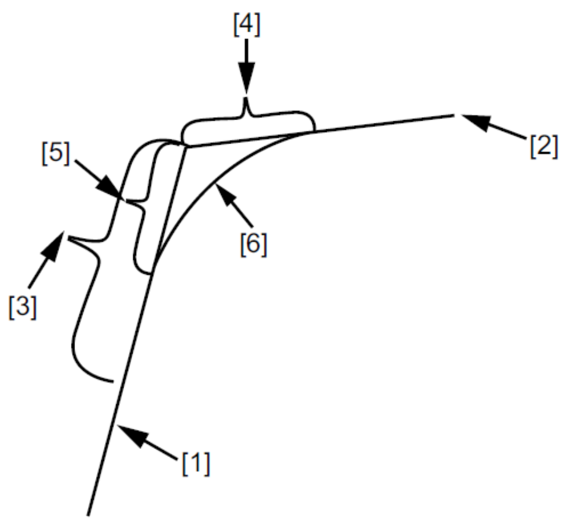
Robot program



```
N10  MotionSet := Fast
N20  LIN POSEVAR[1]
N30  BlendingDistance := Small [mm]
N40  MotionSet := Slow
N50  LIN POSEVAR[2]
N60  END_PROG
```

31458160523

Figure



31458163211

[1] Linear segment to position POSEVAR[1]
[2] Target position POSEVAR[2]
[3] Set BlendingDistance small for blending to the path segment to the target position POSEVAR[2])
[4] Segment length * limitation percentage
[5] Blending distance$_{\text{effective}}$
[6] Resulting symmetrical blending arc

Blending with branching

The following example illustrates blending from one motion segment to another motion segment depending on the BOOLEAN variable *ToTheLeft*:

- Motion segment 1: MotionSet fast, LIN, target position PoseApproach
- Motion segment 2, depending on the BOOLEAN variable *ToTheLeft*:

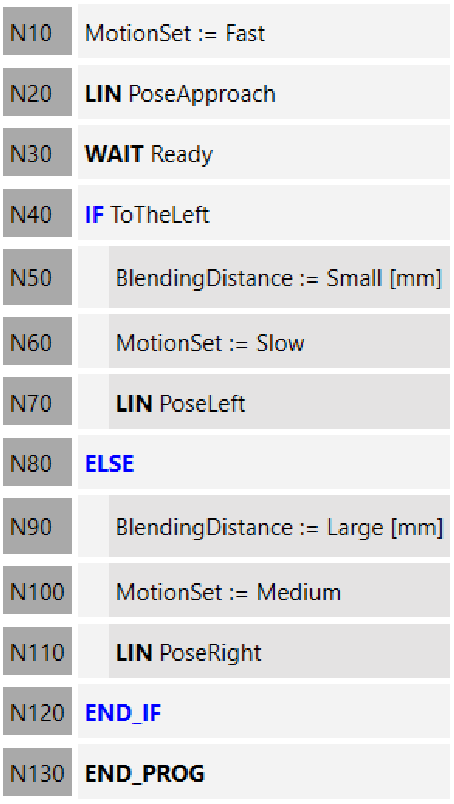
  BlendingDistance small, MOTIONSET slow, LIN, target position PoseLeft

  or

  BlendingDistance large, MOTIONSET medium, LIN, target position PoseRight

Depending on the point to be approached, *PoseLeft* or *PoseRight*, e.g. to avoid collisions with the BlendingDistance *Small* or *Large*, blending to the second motion segment is to be performed. The decision where to move to, however, is only made during the movement to the intermediate position *PoseApproach*. For this purpose, WAIT *Ready* is used to wait before the BOOLEAN variable *ToTheLeft* is evaluated.

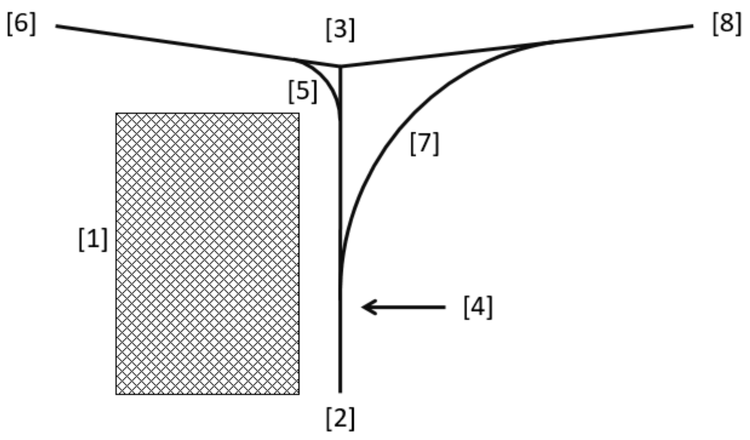
Robot program



```
N10   MotionSet := Fast
N20   LIN PoseApproach
N30   WAIT Ready
N40   IF ToTheLeft
N50       BlendingDistance := Small [mm]
N60       MotionSet := Slow
N70       LIN PoseLeft
N80   ELSE
N90       BlendingDistance := Large [mm]
N100      MotionSet := Medium
N110      LIN PoseRight
N120  END_IF
N130  END_PROG
```

31458170123

Figure



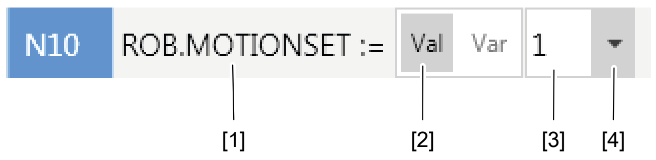
31458366859

[1] Collision object
[2] Start position
[3] Intermediate position *PoseApproach*
[4] At this point the BOOLEAN variable *Ready* becomes TRUE
[5] Small blending curve: BlendingDistance *Small*
[6] Target position *PoseLeft*
[7] Large blending curve: BlendingDistance *Large*
[8] Target position *PoseRight*

9.8.3 Adding a motion parameter set assignment

Assignment with which the parameters included in a set of motion parameters are assigned to the following motion commands. For further information, refer to the chapters "Sets of motion parameters" (→ 🗎 21), "Setting sets of motion parameters" (→ 🗎 46), and "Setting motion parameters" (→ 🗎 161).

"New Block" value: "SET_MOTIONSET"



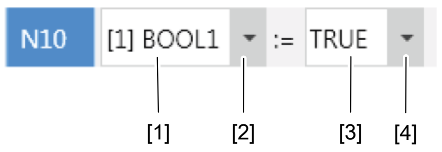
9007224177020939

[1] Identifier for the assignment of the set of motion parameters
[2] Setting whether a fixed value (VAL) or the value of a variable (VAR) is to be assigned.
[3] If [2] is set to VAL: Value that is assigned.
If [2] is set to VAR: Index of the real variable which has its value assigned.
[4] Drop-down list for setting [3]

9.8.4 Adding a Boolean assignment

Assignment in which a value is assigned to a Boolean robot program variable.

"New Block" value: "SET_BOOLVAR"



29048513163

[1] Selection of the BOOLEAN variable
[2] Drop-down list for replacing [1]
[3] Value to be assigned
[4] Drop-down list for replacing [3]

9.8.5 Adding real assignment/calculation

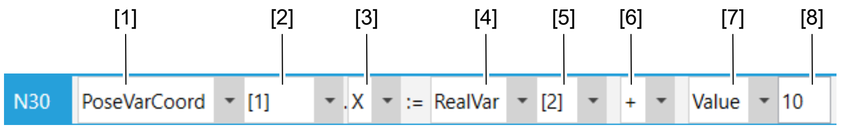
Command for calculating and assigning a value of the data type REAL. The result of the calculation can either be assigned to a REAL variable or to the coordinate of a POSE variable. You can select for the two operands of the calculation whether these are a value, a REAL variable or the coordinate of a POSE variable. Examples:

POSE[1].Z := REAL[1] + 10

REAL[1] := REAL[2] + REAL[3]

POSE[1].Z := POSE[2].Z + REAL[2]

"New Block" value: "CALC_REALVAR"



29052676747

[1] Selection whether the calculation result is to be assigned to the coordinate of a POSE variable or of a REAL variable.
[2] Selection of the target variable. Depending on what is selected in field [1], you can choose a POSE variable or a REAL variable.
[3] Selection of the coordinate of the selected POSE variable. Only visible when a "PoseVarCoord" is selected in field [1].
[4] Type of the first operand: Selection whether the first operand is to be a value, a REAL value or the coordinate of a POSE variable.
[5] Selection of the first operand. If you have specified in field [4] that the first operand is to be a value, you can enter this value here.
[6] Selection of the operator (+, -, *, /) to be used for the two operands.
[7] Type of the second operand: Selection whether the second operand is to be a value, a REAL value or the coordinate of a POSE variable.
[8] Selection of the second operand. If you have specified in field [7] that the second operand is to be a value, you can enter this value here.

9.9 Control structures

9.9.1 Conditional statement

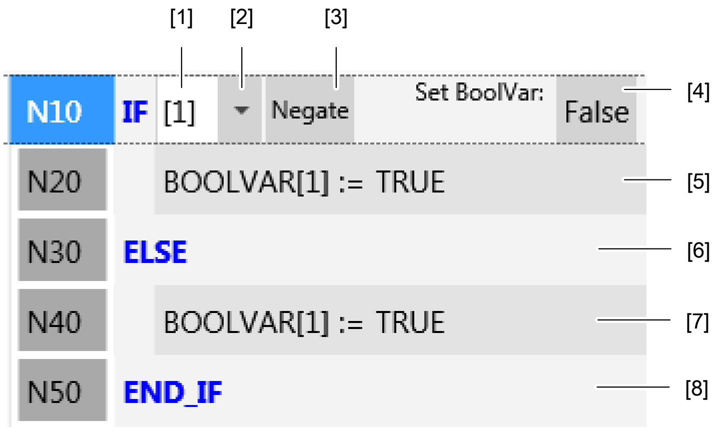
If the condition is met, the statements of the IF part will be executed. If the condition is not met, the statements of the ELSE part will be executed.

IF	Condition
	Statements of the IF part
ELSE	
	Statements of the ELSE part
END_IF	

Adding an IF condition

"New Block" value: "IF"

Move the start and the end of the IF condition to the desired program lines.



```
N10  IF [1] ▾ Negate   Set BoolVar: False
N20      BOOLVAR[1] := TRUE
N30  ELSE
N40      BOOLVAR[1] := TRUE
N50  END_IF
```

18014423432738955

[1] Argument for the IF condition: Boolean variable
[2] Drop-down list for replacing [1]
[3] Setting to negate [1] as an argument for the condition (a NOT will be inserted between IF and [1]).
[4] Switch to set the value for [1] ("TRUE" or "FALSE")
[5] Lines to be executed if [1] is met.
[6] Line with the ELSE part of the IF condition
[7] Lines to be executed if [1] is not met.
[8] Line with the IF condition end

Adding an ELSE condition

✓ An IF condition exists in the program.

1. Select the line between IF and END_IF below which you would like to insert the ELSE part.
2. Select the value "ELSE" from the drop-down list "New Block".
3. Click the [Add] button.
4. Parameterize the statements of the ELSE part.

9.9.2 Loop

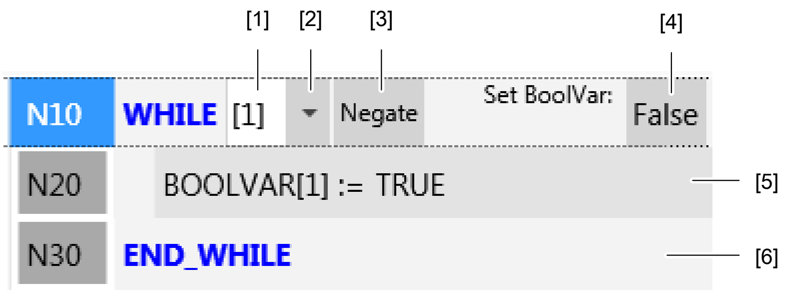
Executes the statements until the condition is no longer fulfilled.

WHILE	Condition
	Statements
END_WHILE	

Adding a WHILE loop

"New Block" value: "WHILE".

Move the start and the end of the WHILE loop to the desired program lines.



```
N10  WHILE [1]
N20      BOOLVAR[1] := TRUE
N30  END_WHILE
```

18014423432865803

[1]	Argument for the WHILE loop: Boolean variable
[2]	Drop-down list for replacing [1]
[3]	Setting to negate [1] as an argument for the condition (a NOT will be inserted between IF and [1]).
[4]	Switch to set the value for [1] ("TRUE" or "FALSE")
[5]	Lines to be repeated as long as [1] is met.
[6]	Line with WHILE loop end

9.9.3 Wait command

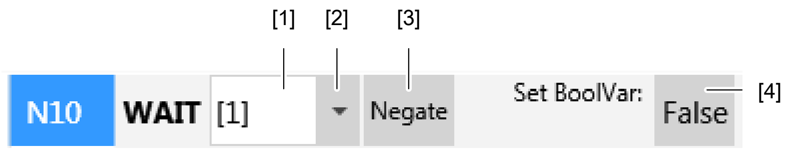
The system remains at the WAIT command until the condition is met.

WAIT condition

The condition can be a Boolean variable, a time interval, or a status signal, e.g. "end of motion" (MOTION_DONE).

Wait to add a Boolean variable

"New Block" value: "WAIT_BOOLVAR"

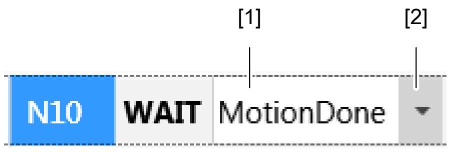


18014423432886411

[1] Argument for the wait command: Boolean variable
[2] Drop-down list for replacing [1]
[3] Setting to negate [1] as argument for the condition.
[4] Switch to set the value for [1] ("TRUE" or "FALSE")

Wait to add a motion

"New Block" value: "WAIT_MOTION"



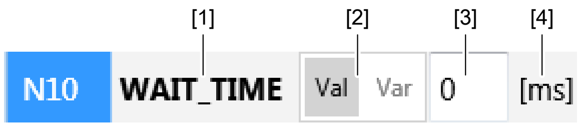
18014423432894987

[1] Argument for the wait command: end of motion
- Motion_Done: Robot has executed all travel commands and is in idle state, or motion is on the last path segment.
- MotionInLastSeg: Robot is currently executing the last travel command.

[2] Drop-down list for replacing [1]

Insert waiting for dead time to expire

Waiting time starts after the motion command has ended before WAIT_TIME. "New Block" value: WAIT_TIME



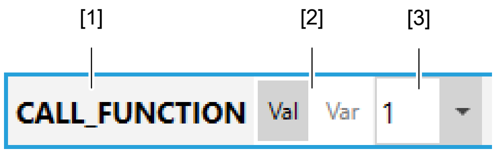
18014423432959755

[1] Identifier for the wait command with expiry of a dead time
[2] Setting whether a fixed value (VAL) or the value of a variable (VAR) is to be used.
[3] If [2] is set to VAL: Value that is used.
If [2] is set to VAR: Index of the real variable that has its value used.
[4] SI of the delay (ms = milliseconds)

9.9.4 IEC function call

Command for calling an IEC function that can be programmed in the IEC Editor by the user. This command calls the IEC function during each cycle of the high-priority cyclical task (TaskHighPrio) until the IEC function returns the return value "TRUE". Only then is the next command executed.

The CallFunction can be used, for example, to create a consistent process image at a clearly defined point in the program sequence. For example, positions and MotionSets (using a method in the user interface) can be described that are to be used in the following SRL code.



18014423433087499

[1] Identifier for the IEC function call
[2] Selection whether the number of the function to be called should be entered directly or read from a variable.
[3] Transfer parameter for the function call so that various functions can be called using a case distinction in the IEC function.

How the *CallFunction* is created in the IEC program and how it is connected to the robot is described in chapter "IEC function call for the robot program" (→ 🗎 142).

9.9.5 End of program

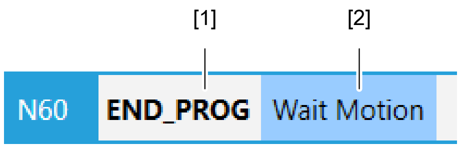
Command for ending the program

After a program has been ended with this command, the next program can be started. The program end command can also be used to end the program prematurely. It must end at least after the last command, but may also be used several times.

Adding end of program

"New Block" value: "END_PROG"

You can select whether the program end command should wait until the robot movement is finished before the program is ended or whether the program should be ended immediately when program execution (program pointer) has reached the program end command. Without waiting for the movement to end, it is possible, for example, to continuously feed in movement segments.



31659655563

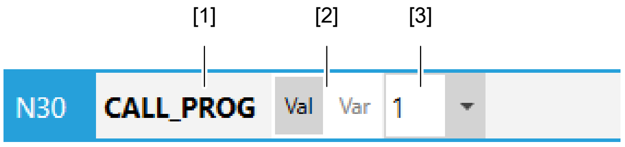
[1] Identifier for program end command
[2] Button to enable/disable waiting for the movement to end

9.9.6 Subroutine call

This command calls and executes another robot program. After the called program is finished, the original program is continued in the line following the CALL_PROG block. The maximum nesting depth for the case that programs called by another program again contain program calls is 10.

Adding a call to another robot program

"New Block" value: "CALL_PROG".



31659767435

[1]	Identifier for calling another program
[2]	Selection whether the number of the program to be called should be entered directly or read from a variable.
[2]	Edit box for the number of the program to be called

9.9.7 Path event

The REG PATH_EVENT command can be used to register an instruction to be executed at a certain event depending on the path progression. You can define the path event based on path and/or time.

For more detailed information, refer to chapters "Path events" (→ 🗎 22) and "Functional description" (→ 🗎 81).

When the path event occurs, one of the following instructions can be executed: SET_BOOLVAR, CALL_FUNCTION, REG_TP_POSITIONING, REG_TP_MEASURE, DEREG_TP. The instruction to be executed must be in the line below the REG PATH_EVENT command and is marked by an indentation. Only one instruction can be programmed.

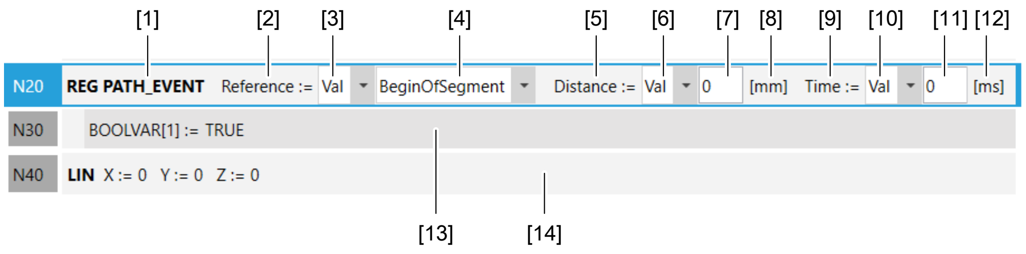
The parameters in the REG PATH_EVENT command refer to the next motion command in the robot program. The robot program is only displayed as correct in the RobotMonitor if REG PATH_EVENT is followed by a motion command. However, other instructions may be programmed between REG PATH_EVENT and the motion command, such as another REG PATH_EVENT, which then all refer to the same subsequent motion command. In this way it is possible to execute several instructions at the same path event using several REG PATH_EVENT instructions with the same parameterization.

In the special situation that all motion commands following REG PATH_EVENT in the same robot program are not executed, e.g. because of an IF branch, and the robot program was called as a subroutine (CALL_PROG), REG PATH_EVENT refers to the first motion command in the calling robot program. In contrast, every program stop and every program start (basic state) deletes all registered and not yet triggered path events.

The IEC code in the CALL_FUNCTION instruction must return Done = TRUE when triggered by a path event within one cycle. This can, for example, trigger a concurrent process in the IEC.

Adding a path event

"New Block" value: "REG_PATH_EVENT"



```
N20 REG PATH_EVENT  Reference := Val  BeginOfSegment  Distance := Val  0  [mm]  Time := Val  0  [ms]
N30     BOOLVAR[1] := TRUE
N40 LIN X := 0  Y := 0  Z := 0
```

31659788811

[1]	Identifier for registering a path event
[2]	Identifier for the reference point
[3]	Selection whether the reference point should be entered directly, read from a variable, or retained from a previous command
[4]	Edit box for the reference point
[5]	Identifier for the distance parameter
[6]	Selection whether the distance value should be entered directly, read from a variable, or retained from a previous command
[7]	Edit box for the distance parameter
[8]	Unit of the distance parameter
[9]	Identifier for the time parameter
[10]	Selection whether the value for the time parameter should be entered directly, read from a variable, or retained from a previous command
[11]	Edit box for the time parameter
[12]	Unit of the time parameter
[13]	Instruction to be executed when the event registered in the previous line occurs, in the example: Set a BOOLEAN variable to TRUE. This line is skipped in the normal program sequence.
[14]	Path segment to which REG PATH_EVENT refers

A path event that has been registered (program interpreter has accepted the subsequent motion command) is marked with a rotating circle. As soon as the path event is triggered, it is marked with a green tick. When starting and stopping the program, the circle and tick are deleted. If a path event is registered again, e.g. in a loop or in subroutines, the display changes from the tick back to the rotating circle. If a path event has already been triggered and the robot is not allowed to start yet due to a lead time, the tick and the rotating circle are displayed simultaneously; the rotating circle disappears when the robot starts moving.

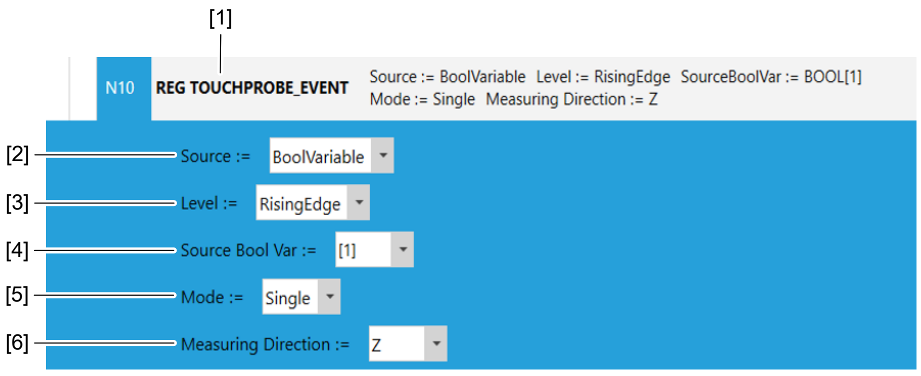
The points at which the path events are triggered are displayed in the 3D simulation when the pen is switched on. Due to the low-priority communication between MOVI-C® CONTROLLER and PC, minor inaccuracies can occur in the 3D display.

9.9.8 Touchprobe

Registering a touchprobe event

The touchprobe function is activated and parameterized using the instruction REG_TOUCHPROBE_EVENT.

"New Block" value: "REG_TP_EVENT".



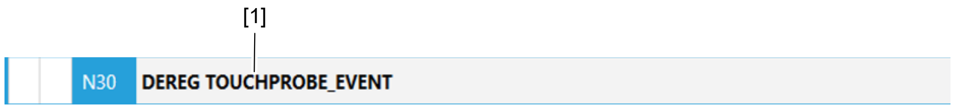
31690470539

[1] Identifier for the instruction
[2] Used touchprobe trigger source: You can choose between "BoolVariable" (BOOL touchprobe) and "InverterTouchprobe" (inverter touchprobe)
[3] Only for BOOL touchprobe: Edge used for triggering the touchprobe event. You can choose between "RisingEdge", "FallingEdge", and "RisingFallingEdge" (each edge change).
For inverter touchprobe, the edge is set by configuring the individual axes in MOVISUITE®.
[4] Only for BOOL touchprobe: SRL BOOLEAN variable used for triggering the touchprobe event
[5] Execution mode of the touchprobe: You can choose between "Single" (touchprobe is deactivated after triggering the event) and "Multiple" (touchprobe remains activated after triggering the event). On each trigger, the instruction assigned to the event is executed)
[6] Direction in which the touchprobe sensor measures or the BOOLEAN variable is switched, e.g. the z-direction (height/stroke) during palletizing.
With subsequent "POSITIONING" instruction: Effective direction of the length of the remaining distance

Deregistering a touchprobe event

An activated touchprobe function is deregistered using the DEREG_TOUCHPROBE_EVENT instruction. This instruction can be used, for example, if measurements are only to be performed in a certain area of the path.

"New Block" value: "DEREG_TP_EVENT".



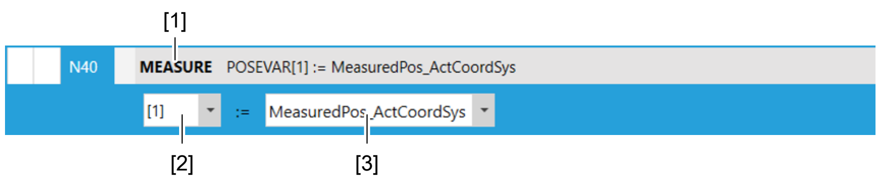
31690744843

[1] Identifier for the instruction

Performing touchprobe measurement

The instruction "REG_TP_MEASURE" is used to write the measured position into an SRL pose variable after triggering a touchprobe event. A "MEASURE" instruction may only be used under a "REG_TOUCHPROBE_EVENT" instruction.

"New Block" value: "REG_TP_MEASURE".



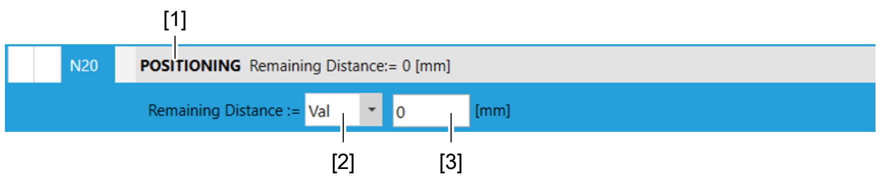
31690791307

[1] Identifier for the instruction
[2] Selection of the SRL pose variable in which the measured position should be stored.
[3] Selection of the coordinate system in which the measured position is to be stored. Selection options: "MeasuredPos_ActCoordSys" and "Measured-Pos_BaseCoordSys"

Performing sensor-based positioning

The "REG_TP_POSITIONING" instruction is used to initiate sensor-based positioning after a touchprobe event has been triggered. This positioning is performed along the path segments by the length *Remaining Distance* in the direction of the *MeasuringDirection*. If necessary, the last motion segment is extended. A "POSITIONING" instruction may only be used under a "REG_TOUCHPROBE_EVENT" instruction. After a "POSITONING" instruction, a "CONTINUE AFTER POSITIONING EVENT" instruction is mandatory.

"New Block" value: "REG_TP_POSITIONING".



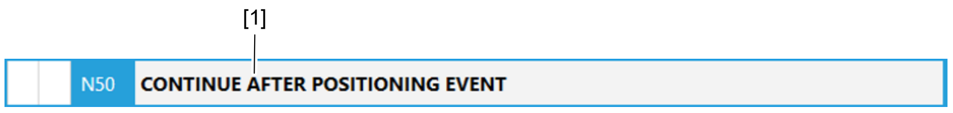
31690803723

[1] Identifier for the instruction
[2] Option for setting the *Remaining Distance*:
Selection options: "Var", "Val", "Keep"
[3] Specification of the *Remaining Distance* in [mm].

Specifying the next program position

The "CONTINUE AFTER POSITIONING EVENT" instruction is used to mark the point in the program sequence up to which path segments belong to the remaining distance travel range. When sensor-based positioning is started, the program jumps to the line after the "CONTINUE AFTER POSITIONING EVENT" instruction.

"New Block" value: "TP_CONTINUE_AFTER_POS".



31690812171

[1] Identifier for the instruction

If the motion reaches the motion segment after the "CONTINUE AFTER POSITIONING EVENT" instruction (indicated by the motion pointer "M") and no touchprobe event has yet occurred, the touchprobe function is disabled.

9.10 Variables

A list of variables is provided that can be used in the robot program. These variables can be written and read from the IEC and from the robot program.

Variables Robot program	Description
Bool	Variables with the name and value of the Boolean variable.
Real	Variables with the name and value of the floating point variable.
Pose	Variables with the names and coordinates of the poses.
MotionSet	Variables with the sets of motion parameters. See also the chapters "Sets of motion parameters" (→ 🗎 21), "Setting sets of motion parameters" (→ 🗎 46), and "Setting motion parameters" (→ 🗎 161). The specified "Default units" (→ 🗎 21) apply.

9.10.1 Editing robot program variables

✓ The robot monitor has started and is connected.

1. Switch to the "Variables" tab.
2. Switch to the tab of the desired robot program variables.
   ⇨ The list of the desired robot program variables is shown. One line corresponds to one variable.
3. In the "Name" column, enter an individual name for the variable. This name will be displayed in the robot program. If this field is empty, only the number of the robot program will be displayed.
4. In the "Value" column, enter the desired values for the variable.
5. As long as the values have not taken effect, the line is marked yellow. To have the values take effect, make sure the cursor is positioned in the respective edit box and click the enter key, or select the line and click the [Send selected variable (POSE/BOOL/REAL)] button. Doing so stores the selected line in the controller's volatile memory.
6. To save the variables and the program on the memory card permanently, click the save icon.

10 IEC programming

This chapter describes the signals of the user interface for programming in the IEC. The sequence in which the signals are to be controlled is described in chapter "Control via the process controller" (→ 🗎 93).

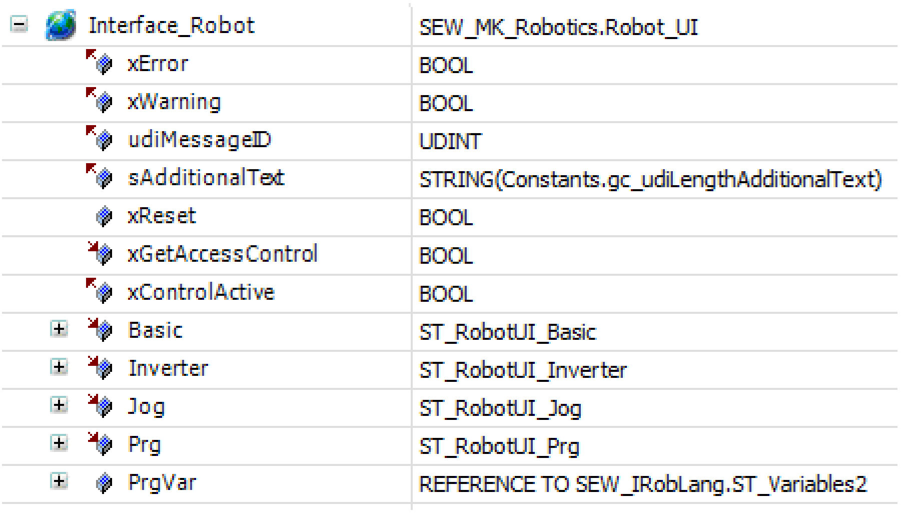
10.1 Opening the IEC project

- If an IEC project has already been generated, select the entry [IEC Editor] under "Tools" from the context menu of the MOVI-C® CONTROLLER in MOVISUITE®.
- If no IEC project has been generated, follow the steps described in the "Generating an IEC project" (→ 🗎 42) chapter.

10.2 User interface

The user interface is implemented in the IEC program by an instance in the global variable list *SEW_GVL*.

The following figure shows the interface in the IEC Editor:



Interface_Robot	SEW_MK_Robotics.Robot_UI
xError	BOOL
xWarning	BOOL
udiMessageID	UDINT
sAdditionalText	STRING(Constants.gc_udiLengthAdditionalText)
xReset	BOOL
xGetAccessControl	BOOL
xControlActive	BOOL
Basic	ST_RobotUI_Basic
Inverter	ST_RobotUI_Inverter
Jog	ST_RobotUI_Jog
Prg	ST_RobotUI_Prg
PrgVar	REFERENCE TO SEW_IRobLang.ST_Variables2

18014423142566411

Following this variable structure, the individual variables are described in more detail in the following chapters.

10.3 Diagnostics

Variables for reporting and writing errors and warnings.

Variable name	Description
xError	Data type – BOOL
	• TRUE – Error present • FALSE – No error present
xWarning	Data type – BOOL
	• TRUE – Warning present • FALSE – No warning present
udiMessageID	Data type: UDINT
	Message ID number
sAdditionalText	Data type: STRING
	Additional message text
xReset	Data type – BOOL
	• TRUE – Reset messages • FALSE – Do not reset messages

10.4 Access management

Variables for managing access permissions.

Variable name	Description
xGetAccessControl	Data type – BOOL
	• TRUE – Request access • FALSE – Return access
xControlActive	Data type – BOOL
	• TRUE – Access granted • FALSE – Access denied

Access via user interface (UserInterface):

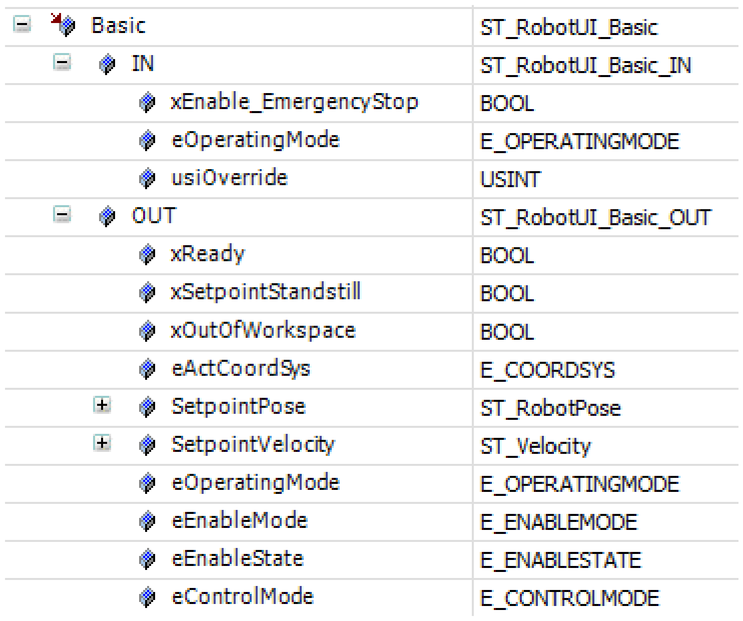
An instance requests access by setting *xGetAccessControl* to "TRUE". If *xControlActive* returns a "TRUE" value, access has been granted and is now permitted.

Access via other function blocks in the user program:

If another function block in control mode wants to access the device interface at the same time as the user interface (UserInterface), the device interface decides which one receives write permission based on the priority of the function block. The user interface (UserInterface) has the highest priority. This means if the user interface (UserInterface) has control access, then *xControlActive* returns "FALSE" to all other function blocks.

10.5 Basic functions (basic)

Interface in the IEC Editor



Basic	ST_RobotUI_Basic
IN	ST_RobotUI_Basic_IN
xEnable_EmergencyStop	BOOL
eOperatingMode	E_OPERATINGMODE
usiOverride	USINT
OUT	ST_RobotUI_Basic_OUT
xReady	BOOL
xSetpointStandstill	BOOL
xOutOfWorkspace	BOOL
eActCoordSys	E_COORDSYS
SetpointPose	ST_RobotPose
SetpointVelocity	ST_Velocity
eOperatingMode	E_OPERATINGMODE
eEnableMode	E_ENABLEMODE
eEnableState	E_ENABLESTATE
eControlMode	E_CONTROLMODE

18014423225395339

10.5.1 IN

Variable name	Description
xEnable_EmergencyStop	Data type – BOOL
	• TRUE – Enable software module • FALSE – The software module executes an emergency stop using the set emergency stop ramp.
eOperatingMode	Data type: E_OPERATINGMODE
	"Operating mode" (→ 🗎 50) • Manual_WithHighSpeed • Automatic Namespace: *SEW_MK_Robotics.SEW_IRobHPub*
usiOverride	Data type: USINT
	Speed scaling in percent for all types of controls

10.5.2 OUT

Variable name	Description
xReady	Data type – BOOL
	• TRUE – Software module is ready for operation (ready for enable) • FALSE – Software module is not ready for operation. Not all requirements for successful enable are fulfilled (error message with the cause is reported if enable is nevertheless carried out).

Variable name	Description
xSetpointStandstill	Data type: BOOL
	• TRUE – Target pose is in idle state/standstill. • FALSE – Target pose is not in idle state/standstill. The robot is moving.
xOutOfWorkspace	Data type – BOOL
	• TRUE – Robot is outside the allowed work envelope • FALSE – Robot is within the allowed work envelope
eActCoordSys	Data type: E_COORDSYS
	Coordinate system in which the path is currently being interpolated (Base, Joint, World, User). Namespace: *SEW_MK_Robotics.SEW_IRobMoCPub*
usiConstellation	Data type: USINT
	Setpoint of the constellation of the kinematics
eOperatingMode	Data type: E_OPERATINGMODE
	"Operating mode" (→ 🗎 50) • Manual_WithHighSpeed • Automatic Namespace: *SEW_MK_Robotics.SEW_IRobHPub*
eEnableMode	Data type: E_ENABLEMODE
	Current enable mode. "Enable mode" (→ 🗎 51) • EmergencyStop_Axes • EmergencyStop • ApplicationStop • Enable Namespace: *SEW_MK_Robotics.SEW_IRobHPub*
eEnableState	Data type – E_ENABLESTATE
	Enable state. "Enable state" (→ 🗎 53) • NoAccessToAxesRequested • EmergencyStoppedAxes • EmergencyStoppingAxes • WaitingForSetpointActive • EmergencyStoppingOnPath • PositionHoldControl • ApplicationStoppingOnPath • WaitingForMotionCommand • PathMotion Namespace: *SEW_MK_Robotics.SEW_IRobHPub*

Variable name	Description
eControlMode	Data type: E_CONTROLMODE
	Type of control. "Type of control" (→ 🗎 54) • Inactive • JogMode • ProgramMode Namespace: *SEW_MK_Robotics.SEW_IRobHPub*

SetpointPose

Variable name	Description
alrActCoordSys	Data type: ARRAY [1..6] OF LREAL
	Target pose in the current coordinate system (Basic.Out.eActCoordSys)
alrBase	Data type: ARRAY [1..6] OF LREAL
	Target position of the base coordinate system
alrKinematicJoint	Data type: ARRAY [1..6] OF LREAL
	Target pose of the joint axes of the kinematic model
alrAdditionalJoint	Data type: ARRAY [1..6] OF LREAL
	Target pose of the additional joint axes (additional joint axes are not included in the current range of functions).
usiConstellation	Data type: USINT
	Setpoint of the constellation of the kinematics

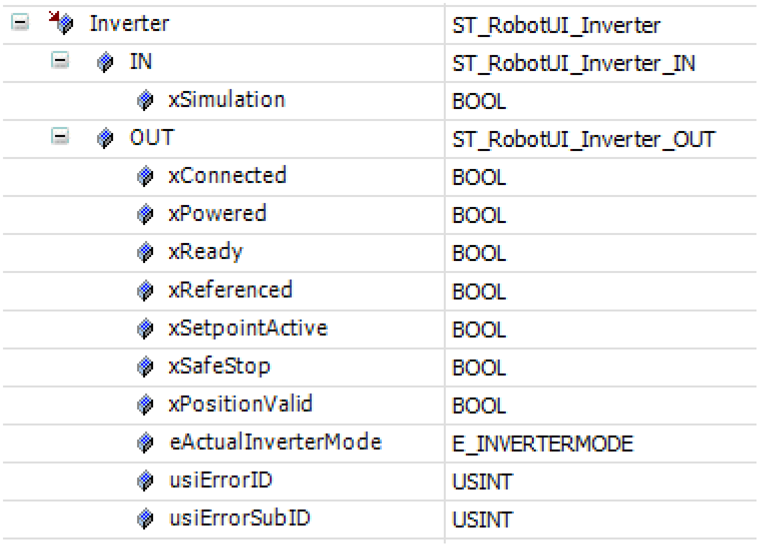
SetpointVelocity

ActCoordSys

Variable name	Description
lrTranslation	Data type: LREAL – Floating-point number
	Translation speed of the TCP in the current coordinate system.

10.6 Inverter functions (Inverter)

Interface in the IEC Editor



Inverter	ST_RobotUI_Inverter
IN	ST_RobotUI_Inverter_IN
xSimulation	BOOL
OUT	ST_RobotUI_Inverter_OUT
xConnected	BOOL
xPowered	BOOL
xReady	BOOL
xReferenced	BOOL
xSetpointActive	BOOL
xSafeStop	BOOL
xPositionValid	BOOL
eActualInverterMode	E_INVERTERMODE
usiErrorID	USINT
usiErrorSubID	USINT

18014423226531979

10.6.1 IN

For further information, refer to chapter "Simulation" (→ 🗎 59).

Variable name	Description
xSimulation	Data type – BOOL
	• TRUE – Simulate the frequency inverter of the software module (e.g. when testing without hardware). • FALSE – Do not simulate the frequency inverter.

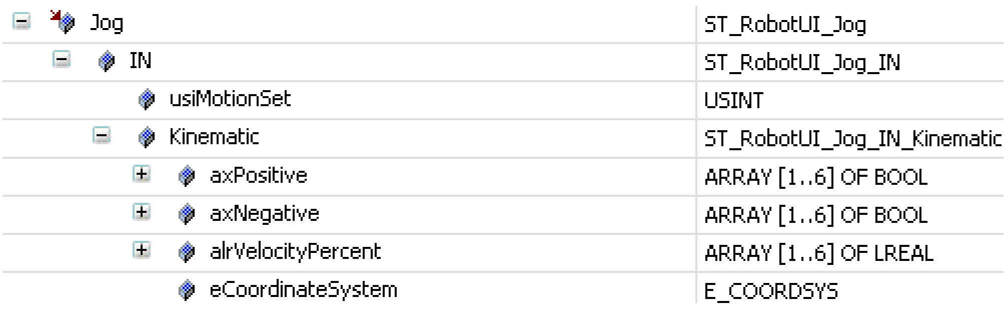
10.6.2 OUT

Variable name	Description
xConnected	Data type – BOOL
	• TRUE – Communication connection with controller • FALSE – No communication connection with controller
xPowered	Data type – BOOL
	• TRUE – Output stages enabled (provide output voltage) • FALSE – Output stages not enabled
xReady	Data type – BOOL
	• TRUE – Ready for control by the controller • FALSE – Not ready for control by the controller
xReferenced	Data type: BOOL
	• TRUE – Referenced • FALSE – Not referenced

Variable name	Description
xSetpointActive	Data type – BOOL
	• TRUE – Setpoints are processed • FALSE – Setpoints are not processed
xSafeStop	Data type – BOOL
	• TRUE – Axis is at standstill (STO active) • FALSE – Axis is not at a standstill (STO is not active).
xPositionValid	Data type: BOOL
	• TRUE – The position of the encoder is valid. • FALSE – There is an encoder fault or error. (E.g. due to a bird strike)
eActualInverterMode	Data type: E_INVERTERMODE
	Operating mode of the inverter (FCB of the inverter): • Unknown • Default • OutputDisabled (FCB 01) • ManualMode (FCB 04) • Stop (FCB 02) • Homing (FCB 12) • JogMode (FCB 20) • BrakeTest (FCB 21) • Positioning (FCB 09) • PositioningInterpolated (FCB10) • Velocity (FCB 05) • VelocityInterpolated (FCB 06) • Torque (FCB 07) • TorqueInterpolated (FCB 08) • MotorParamMeasurement (FCB 25) • PosHoldCtrl (FCB 19) • RotorPosIdentification (FCB 18) • ApplicationStop (FCB 13) • EmergencyStop (FCB 14) • UserStop (FCB 26) *Library: SEW DeviceHandler Interfaces*
usiErrorID	Data type: USINT
	Error ID
usiErrorSubID	Data type: USINT
	Suberror ID

10.7 Jog

Interface in the IEC Editor



Jog	ST_RobotUI_Jog
IN	ST_RobotUI_Jog_IN
usiMotionSet	USINT
Kinematic	ST_RobotUI_Jog_IN_Kinematic
axPositive	ARRAY [1..6] OF BOOL
axNegative	ARRAY [1..6] OF BOOL
alrVelocityPercent	ARRAY [1..6] OF LREAL
eCoordinateSystem	E_COORDSYS

9007223971732619

For further information, refer to chapter "Jog mode" (→ 🗎 54).

10.7.1 IN

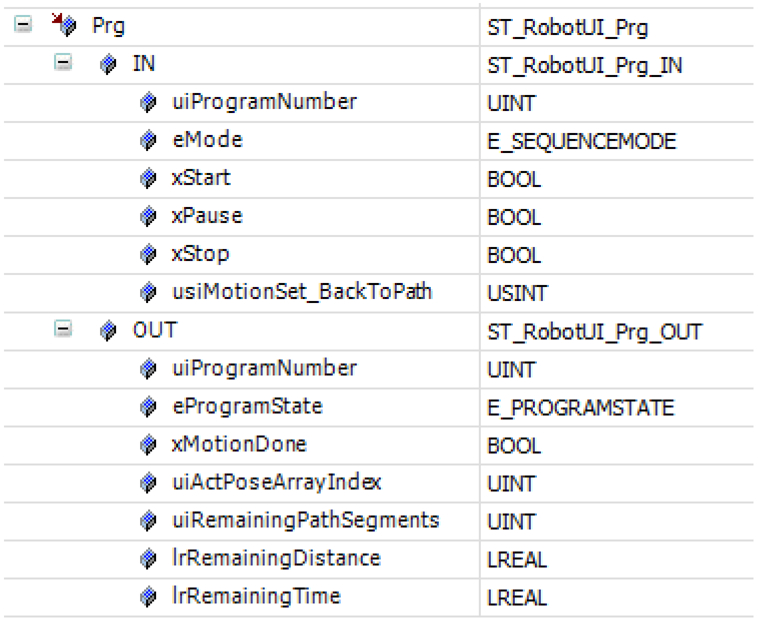
Variable name	Description
usiMotionSet	Data type: USINT
	Set of motion parameters for jog mode

Kinematics

Variable name	Description
axPositive[1..6]	Data type: ARRAY [1..6] OF BOOL
	Jog in a positive direction of the respective axis of the reference coordinate system (Jog.IN.Kinematic.eCoordinateSystem).
axNegative[1..6]	Data type: ARRAY [1..6] OF BOOL
	Jog in a negative direction of the respective axis of the reference coordinate system (Jog.IN.Kinematic.eCoordinateSystem).
alrVelocityPercent[1..6]	Data type: ARRAY [1..6] OF LREAL
	Jog speed scaling in percent of the respective axis of the reference coordinate system (Jog.IN.Kinematic.eCoordinateSystem)
eCoordinateSystem	Data type: E_COORDSYS
	Reference coordinate system in which jogging takes place. Currently supported: Joint axes ("Joint") and Cartesian axes of the base coordinate system ("Base"). Namespace: *SEW_MK_Robotics.SEW_IRobMoCPub*

10.8 Program control (Prg)

Interface in the IEC Editor



Prg	ST_RobotUI_Prg
IN	ST_RobotUI_Prg_IN
uiProgramNumber	UINT
eMode	E_SEQUENCEMODE
xStart	BOOL
xPause	BOOL
xStop	BOOL
usiMotionSet_BackToPath	USINT
OUT	ST_RobotUI_Prg_OUT
uiProgramNumber	UINT
eProgramState	E_PROGRAMSTATE
xMotionDone	BOOL
uiActPoseArrayIndex	UINT
uiRemainingPathSegments	UINT
lrRemainingDistance	LREAL
lrRemainingTime	LREAL

18014423226476683

For further information, refer to chapter "Program mode" (→ 🗎 55).

10.8.1 IN

Variable name	Description
uiProgramNumber	Data type: UINT
	Number of the main program that is currently being executed. This does not exclude the possibility that a subroutine is currently being executed.
eMode	Data type: E_SEQUENCEMODE
	Type of program sequence (AUTO, STEP_BLOCK) Chapter "Program mode" (→ 🗎 55). Namespace: *SEW_MK_Robotics.SEW_Rob.SEW_IRobLang*
xStart	Data type: BOOL
	• TRUE – Start program processing. • FALSE – In "Manual high speed" operating mode: Pause program processing.
xPause	Data type – BOOL
	• TRUE – Perform application stop and pause program execution • FALSE – Do not pause program execution
xStop	Data type – BOOL
	• TRUE – Perform application stop and stop program execution • FALSE – Do not stop program execution

Variable name	Description
usiMotionSet_BackToPath	Data type: USINT
	Set of motion parameters for repositioning on the path

10.8.2 OUT

Variable name	Description
uiProgramNumber	Data type: UINT
	Number of the main program that is currently being executed. This does not exclude the possibility that a subroutine is currently being executed.
eProgramState	Data type: E_PROGRAMSTATE
	Program execution state. "Program mode" (→ 🗎 55) • NotInitialized • Initialized • BeingExecuted • Paused • Finished • BackToPathRequired • BackToPathActive Namespace: *SEW_MK_Robotics.SEW_IRobHPub*
xMotionDone	Data type – BOOL
	• TRUE – All motion jobs and path events are executed with delay time • FALSE – Not all motion jobs have been executed yet
uiActPoseArrayIndex	Data type: UINT
	List index of the pose variable (PrgVar.astPoseValues) that is used as the target pose in the current motion command. In case of a motion command that does not use a pose variable, the index output is 0.
uiRemainingPathSegments	Data type: UINT
	Number of path segments that have been considered in motion planning and have not yet been traveled.
uiRemainingDistance	Data type: UINT
	Remaining distance of the path segments that are considered in the motion planning and have not yet been traveled.
uiRemainingTime	Data type: UINT
	Remaining time of the path segments that are considered in the motion planning and have not yet been traveled.

Note:

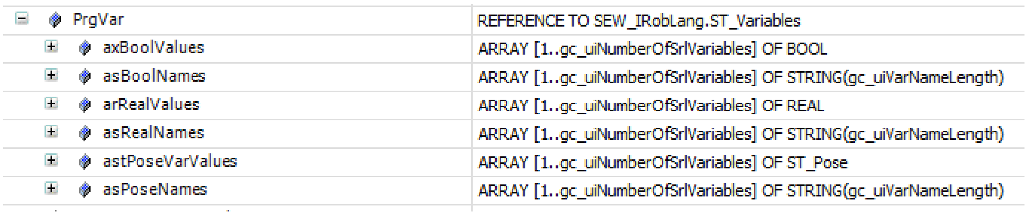
For the *RemainingPathSegments*, *RemainingDistance* and *RemainingTime*, the motion controller only considers the path segments that have already been passed by the robot program (see program pointer).

10.9 Program variables (PrgVar)

INFORMATION

The program variables are not mapped in the *Main* task (by MapIn/MapOut). They operate directly on the reference of the internal variables. Once the assignment is executed in the user program, the assigned value is effective for all commands executed in the robot program from now on. The program variables are excluded from access management.

Interface in the IEC Editor



PrgVar	REFERENCE TO SEW_IRobLang.ST_Variables
axBoolValues	ARRAY [1..gc_uiNumberOfSrlVariables] OF BOOL
asBoolNames	ARRAY [1..gc_uiNumberOfSrlVariables] OF STRING(gc_uiVarNameLength)
arRealValues	ARRAY [1..gc_uiNumberOfSrlVariables] OF REAL
asRealNames	ARRAY [1..gc_uiNumberOfSrlVariables] OF STRING(gc_uiVarNameLength)
astPoseVarValues	ARRAY [1..gc_uiNumberOfSrlVariables] OF ST_Pose
asPoseNames	ARRAY [1..gc_uiNumberOfSrlVariables] OF STRING(gc_uiVarNameLength)

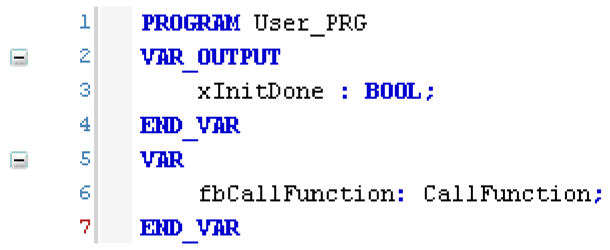
9007223971741835

Variable name	Description
axBoolValues	Data type: ARRAY OF BOOL
	List of values of the Boolean variables
asBoolNames	Data type: ARRAY OF STRING
	List of names of the Boolean variables
arRealValues	Data type: ARRAY OF REAL
	List of values of the floating point variables
asRealNames	Data type: ARRAY OF STRING
	List of names of the floating point variables
astPoseValues	Data type: ARRAY OF ST_Pose
	List of values of the pose variables
asPoseNames	Data type: ARRAY OF STRING
	List of names of the pose variables

10.10 IEC function call for the robot program

To call an IEC function from the robot program, create the following function in the IEC program:

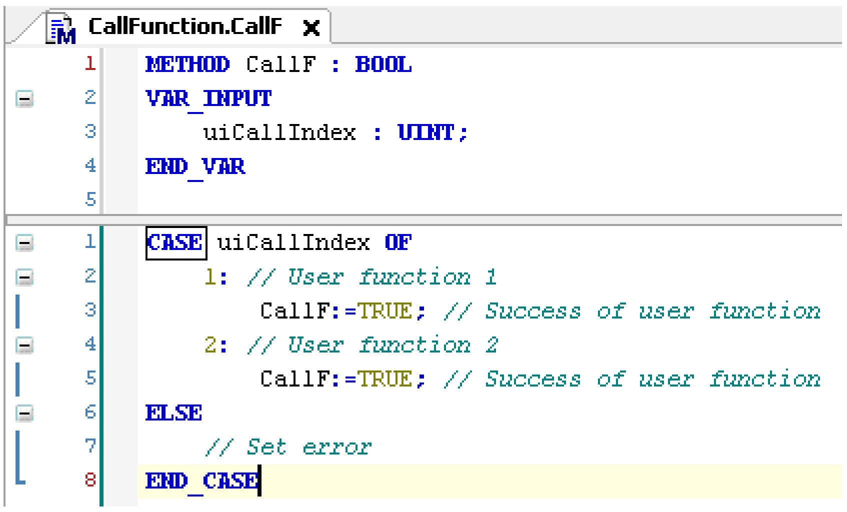
1. Create a new function block (Context menu / Add object / POU… / Type: function block).
2. Implement the following interface using the function block: *SEW_MK_Robotics.SEW_IRobLangPub.ICallF*
3. Implement all interfaces (context menu of the function block/Implement interfaces…).
4. Create an instance of the function block in the *User_PRG* program.



```
PROGRAM User_PRG
VAR_OUTPUT
    xInitDone : BOOL;
END_VAR
VAR
    fbCallFunction: CallFunction;
END_VAR
```

9007225120208267

5. Connect the function block with the robot by adding the following line of code in *User_PRG.Init* (replace "INSTANCE NAME" by the name of the robot in the MOVISUITE® project): `INSTANCENAME.fbProgramInterpreter.LinkICallF(fbCallFunction);`
6. Using the *CallF* method, program a case distinction that depends on the *uiCallIndex* variable.



CallFunction.CallF

```
METHOD CallF : BOOL
VAR_INPUT
    uiCallIndex : UINT;
END_VAR

CASE uiCallIndex OF
    1: // User function 1
        CallF:=TRUE; // Success of user function
    2: // User function 2
        CallF:=TRUE; // Success of user function
ELSE
    // Set error
END_CASE
```

25865677067

7. Program the desired function into the appropriate branch of the case distinction. The next SRL command is always only processed after setting *CallF* to "TRUE".

For further information on calling the IEC function in the RobotMonitor, refer to chapter "IEC function call" (→ 🗎 124).

29180252/EN – 03/2020

11 Process data assignment

This chapter describes the signals of the process data interface via fieldbus. The sequence in which the signals are to be controlled is described in chapter "Control via the process controller" (→ 🗎 93).

11.1 Fieldbus interface

The software module offers a standardized fieldbus interface for actuation from a higher-level controller.

INFORMATION

The fieldbus interface must be activated via the "Fieldbus interface" (→ 🗎 37) configuration menu after the configuration of the software module has been completed.

INFORMATION

The current version of the software module does not support referencing and unreferenced jogging of the axes via the fieldbus. Robots with absolute encoders function without problems after they have been referenced once using manual mode in MOVISUITE®. For further questions and an application solution for referencing and unreferenced jog mode via fieldbus, contact SEW-EURODRIVE.

11.1.1 Fieldbus profiles

INFORMATION

In the current version, the "MOVIKIT® fieldbus monitor" (→ 🗎 156) only supports the fieldbus profile "Standard profile for positioning". The fieldbus profile "Flexible, parameterizable profile" is not supported.

Standard profile for positioning

INFORMATION

For the "standard profile for positioning" it is necessary that a standard program with the corresponding number of path segments has been loaded. See also chapter "Loading a standard program" (→ 🗎 110).

The standard profile for positioning can be used to easily position the software module from the higher-level controller. The path is managed by the higher-level controller and transmitted to the software module. The interface and the SRL program are predefined.

The path is composed of path segments. Each path segment is defined by a target pose, a blending distance, and a motion parameter set. See chapter "Loading a standard program" (→ 🗎 110). The path is covered at the start. Next, further paths can be transmitted and started. For further information on control, refer to the chapter "Process sequence" (→ 🗎 95).

The path consists of motion sequences (without stopping) that are specified by the higher-level controller during runtime through path points and blending ranges. Various enables control when the robot performs which movements. The speed and acceleration at which the robot travels along these path segments is defined by preconfigurable sets of motion parameters. The higher-level controller specifies the number of the motion parameter set to be used for each path segment. For example, data sets for rapid speed, creep speed or gripping movements can be defined.

Flexible parameterizable profile

With the flexible, parameterizable profile, the software module can be controlled in various ways by the higher-level controller via fieldbus. The robot program can be programmed by the user in the RobotMonitor and the used robot program variable can be applied as input or output signals to the fieldbus. With this profile, table positioning of the robot is also possible.

As with the "standard profile for positioning", the first 4 process data words are predefined. The optional diagnostic modules, the standard path segments as well as other robot program variables (poses, real variables, Boolean variables) can be configured individually. See "Fieldbus interface" (→ 🗎 37).

11.1.2 Consistent data transmission

INFORMATION

For consistent data transfer to the MOVI-C® CONTROLLER, the process data must be configured accordingly in the higher-level controller. Consistency blocks must be used in the master that match the selected process data profile. It is recommended to have the robot instance and the consistent block start with the same word.

INFORMATION

The current software version does not support the consistent transfer of data over several consistent blocks. For further questions and an application solution for consistent data transfer over several consistent blocks via the fieldbus, contact SEW-EURODRIVE.

The MOVI-C® CONTROLLER supports consistent data blocks up to 64 process data words (see the MOVI-C® CONTROLLER manual). This ensures consistent data transfer for the "standard profile for positioning" with up to 10 standard path segments. Consistent data transfer for the fieldbus profile "Flexible, parameterizable profile" must be considered individually.

11.1.3 Control words 1-4

Control words 1 to 4 are present in each fieldbus profile and cannot be adjusted. Each robot instance on the fieldbus starts with these 4 process data words.

Word	Bit	In	Description
PO 1	0	Enable/emergency stop	Enable software module
	…	…	…
	8	Acknowledge fault/message	Reset fault/messages
	9	Simulation	Simulate frequency inverter of the software module
	…	…	…
	15	MOVIKIT® Handshake In	This signal is copied internally to status word bit 15 (MOVIKIT® Handshake Out). If the copying operation fails ("Handshake Out" signal remains constant with changing "Handshake In" signal), the device-internal processing of the software module is disrupted.
PO 2	0	Automatic/manual	Setpoint of the "operating mode" (→ 🗎 50)
	…	…	
	2	Program start	Start program execution
	3	Program pause	Perform application stop and pause program execution
	4	Program stop	Perform application stop and stop program execution
	…	…	

Word	Bit	In	Description
PO 3	0	Jog joint axis/Cartesian axis 1 positive	Jog in positive direction of the respective axis of the reference coordinate system (Bit 12)
	1	Jog joint axis/Cartesian axis 2 positive	
	2	Jog joint axis/Cartesian axis 3 positive	
	3	Jog joint axis/Cartesian axis 4 positive	
	4	Jog joint axis/Cartesian axis 5 positive	
	5	Jog joint axis/Cartesian axis 6 positive	
	6	Jog joint axis/Cartesian axis 1 negative	Jog in negative direction of the respective axis of the reference coordinate system (Bit 12)
	7	Jog joint axis/Cartesian axis 2 negative	
	8	Jog joint axis/Cartesian axis 3 negative	
	9	Jog joint axis/Cartesian axis 4 negative	
	10	Jog joint axis/Cartesian axis 5 negative	
	11	Jog joint axis/Cartesian axis 6 negative	
	12	Jog coordinate system: JCS/BCS	Reference coordinate system in which jogging takes place. FALSE – Base coordinate system (BCS) TRUE – Joint coordinate system (JCS)
	…	…	
PO 4	0-7	Program number	Number of the main program to be executed
	8-15	Override	Speed scaling in percent of the programmed speed for all types of control

11.1.4 Status words 1-4

Status words 1 to 4 are present in each fieldbus profile and cannot be adjusted. Each robot instance on the fieldbus starts with these 4 process data words.

Word	Bit	In	Description
PI 1	0	Ready	Software module is ready for operation
	1	STO enable (STO active)	TRUE – Axis is not at a standstill (STO is not active) FALSE – Axis is at standstill (STO active)
	2	Output stage enable	Output stages enabled
	…	…	
	4	Motors turning (motor standstill)	TRUE – Target pose is not in idle state The robot is moving. FALSE – Target pose in idle state
	5	Referenced	Inverter is referenced
	6	Setpoint active	Setpoints are processed
	…	…	
	8	Module fault	A fault occurred
	9	Module warning	A warning is present
	…	…	
	15	MOVIKIT® Handshake Out	This signal is copied internally from control word bit 15 MOVIKIT® Handshake In). If the copying operation fails ("Handshake Out" signal remains constant with changing "Handshake In" signal), the device-internal processing of the software module is dis-rupted.
PI 2	0	Automatic/manual	Current "operating mode" (→ 🗎 50). FALSE – Manual TRUE – Automatic
	…	…	
	2	Program is initialized	Status of "program execution" (→ 🗎 55).
	3	Program is executed	
	4	Program stopped	
	5	Program is completed	
	6	Repositioning is required	
	7	Repositioning is activated	
	…	…	

Word	Bit	In	Description
PI 3	0-15	Message ID	Message identification number (fault, warning, information). The bits PI 1:8 and PI 1:9 can be used to determine the severity of the message: Fault: Only PI 1:8 is "TRUE" Warning: Only PI 1:9 is "TRUE" Information: PI 1:8 and 1:9 are "FALSE"
PI 4	0-7	Program number	Number of the main program that is currently being executed. **NOTE:** This does not exclude the possibility that a subroutine is currently being executed.
	8	Jog	"Type of control" (→ 🗎 54)
	…	…	
	10	Program	
	…	…	

29180252/EN – 03/2020

11.1.5 Standard profile for positioning

The length of the "standard profile for positioning" depends on the maximum number of standard path segments:

- n ≤ 2: Length = 16
- n ≥ 2: Length = 4+6×n

Process output data

INFORMATION

For the "standard profile for positioning" it is necessary that a standard program with the corresponding number of path segments has been loaded. See also chapter "Loading a standard program" (→ 🗎 110).

The following table shows the process output data from the higher-level controller to the software module for control via fieldbus.

Word	Function	Explanation
PO 1-4	Control words 1-4	Basic signals for controlling a robot via fieldbus, see chapter "Control words 1-4" (→ 🗎 145).
PO 5-10	Standard path segment 1	The standard path segments, each of which is 6 process data words long, are available from PO 5 onwards. The number of path segments depends on the configured "maximum number of standard path segments" (→ 🗎 37).
PO 11-16	Standard path segment 2	
etc.	etc.	

Standard path segments in the standard profile

INFORMATION

The numbering of the process data does not refer to the absolute number in the fieldbus interface but is only the relative number in the standard path segment.

The shape of the standard path segment is a straight line (LIN).

Word	Bit	In	Description
PO 1	0-3	Set of motion parameters	Selection of the motion parameter set for this path segment via the 4 bits as NIBBLE.
	4	Wait point	TRUE – Stopping at the previous pose is forced. It must be set to "TRUE" since program start. FALSE – Enable for traveling this path segment if the path has not already been terminated by a path end (bit 5).
	5	Path end	TRUE – End program at this target pose. Only the first path end (across all path segments in the entire telegram) is effective. The signal must remain set to "TRUE" between program start and program end.
	…	…	
PO 2	0-15	Blending distance	Blending distance between the last path segment and this path segment (i.e. at the target pose of the last path segment)
PO 3-6	0-15	Target pose	Specification of the target position for this path segment (X, Y, Z, and A coordinates)

Process input data

The following table shows the process input data from the software module to the higher-level controller for control via fieldbus.

Word	Function	Explanation
PI 1-4	Status words 1-4	Basic signals for diagnosing the robot via fieldbus. See chapter "Status words 1-4" (→ 🗎 147).
PI 5-16	Module for motion diagnostics	Signals for diagnosing the robot motion. See chapter "Module for motion diagnostics in the standard profile" (→ 🗎 151).

Module for motion diagnostics in the standard profile

Word	Bit	In	Description
5	0	Work envelope left	The robot is outside the allowed work envelope.
	1	Current motion task executed	All motion jobs and path events are executed with delay time.
	…	…	
	8-15	Constellation	Setpoint of the constellation of the kinematics
6	0-15	Translational speed	Translation speed of the TCP in the current coordinate system
7	0-15	Target pose in BCS (X coordinate)	Target position of the base coordinate system
8	0-15	Target pose in BCS (Y coordinate)	
9	0-15	Target pose in BCS (Z coordinate)	
10	0-15	Target pose in BCS (A coordinate)	
11	0-7	Currently approached pose	Number of currently covered path segment.
	8-15	Remaining segments	Number of path segments that have been considered in motion planning and have not yet been traveled.
12	0-15	Target distance	Remaining distance of the path segments that are considered in the motion planning and have not yet been traveled.
13	0-15	Target pose in JCS (J1 coordinate)	Target pose of the joint axes of the kinematic model
14	0-15	Target pose in JCS (J2 coordinate)	
15	0-15	Target pose in JCS (J3 coordinate)	
16	0-15	Target pose in JCS (J4 coordinate)	

11.1.6 Flexible parameterizable profile

Process output data

Word	Function	Explanation
PO 1-4	Control words 1-4	Basic signals for controlling the robot via fieldbus. See "Control words 1-4" (→ 🗎 145).
From PO 5	Optional process data modules	From PO 5, optional process data modules can additionally be "configured" (→ 🗎 37). The following order applies: – Standard path segments – Further poses (robot program variables) – Further real variables (robot program variables) – Further Boolean variables (robot program variables)

Process input data

Word	Function	Explanation
PI 1-4	Status words 1-4	Basic signals for diagnosing the robot via fieldbus. See "Status words 1-4" (→ 🗎 147).
From PO 5	Optional process data modules	From PO 5, optional process data modules can additionally be "configured" (→ 🗎 37). The following order applies: – Status word for motion diagnostics – Further poses (robot program variables) – Further real variables (robot program variables) – Further Boolean variables (robot program variables)

Poses

As the pose size is configured on the fieldbus, it varies between 2 and 12 process data words. However, the default pose size is 4 process data words (X, Y, Z, A). In the standard path segments, the motion diagnostics module and the poses of the other robot program variables, all poses are mapped according to the configuration.

29180252/EN – 03/2020

Optional module for motion diagnostics with variable size

Word	Bit	Function	Description
PI 5	0	Work envelope left	The robot is outside the allowed work envelope
	1	Current motion task executed	All motion jobs and path events are executed with delay time
	…		
	8-15	Constellation	Setpoint of the constellation of the kinematics
PI 6	0-15	Translational speed	Translation speed of the TCP in the current coordinate system.
PI 7 to 6+p*	0-15	Target pose in BCS	Target position of the base coordinate system
PI 7+p*	0-7	Currently approached pose	List index of the pose variable (*PrgVar.astPoseValues*) that is used as the target pose in the current motion command. In case of a motion command that does not use a pose variable, the index output is 0.
	8-15	Remaining segments	Number of path segments that have been considered in motion planning and have not yet been traveled.
PI 8+p*	0-15	Target distance	Remaining distance of the path segments that are considered in the motion planning and have not yet been traveled.
PI 9+p* to 8+2p*	0-15	Target pose in JCS	Target pose of the joint axes of the kinematic model

* p = pose size, see chapter "Poses" (→ 🗎 152). Default value: 4.

Standard path segments with variable size

INFORMATION

The numbering of the process data does not refer to the absolute number in the fieldbus interface but is only the relative number in the standard path segment.

Word	Bit	In	Description
PO 1	0-3	Set of motion parameters	Selection of the motion parameter set for this path segment via the 4 bits as NIBBLE.
	4	Wait point	TRUE – Stopping at the previous pose is forced FALSE – Enable for traveling this path segment if the path has not already been terminated by a path end (bit 5).
	5	End point	TRUE – End program at this target position Only the first path end (across all path segments in the entire telegram) is effective.
	…	…	
PO 2	0-15	Blending distance	Blending distance between the last path segment and this path segment (i.e. at the target pose of the last path segment)
PO 3 to 2+p*	0-15	Target pose	Target pose specification for this path segment

* p = pose size, see chapter "Poses" (→ 🗎 152). Default value: 4.

Other robot program variables

The robot program variables (poses, reals, bools) can be mapped to the fieldbus as process input data word as well as process output data word by simple configuration. See chapter "More poses" (→ 🗎 39). These are appended to the end of the fieldbus interface. The signals are mapped as space-savingly as possible.

Example

First the poses are mapped, then the real variables, and then the Boolean variables. For data types smaller than WORD, several real variables are mapped into one WORD (in the following example real 2 and 3). The Boolean variables fill up the WORDs started by the real variables (in the example Booleans 1-4) and are then continued in the next WORD (in the example Booleans 1-4).

Configuration:

- 1 pose with X as DWORD, Y, Z and A as WORD
- 3 real variables as WORD, BYTE and NIBBLE
- 8 Boolean variables

Fieldbus interface:

Word	Bit	Function
1-2	0-15	Pose: X-coordinate as DWORD
3	0-15	Pose: Y-coordinate as WORD
4	0-15	Pose: Z-coordinate as WORD
5	0-15	Pose: A-coordinate as WORD
6	0-15	Real 1 as WORD
7	0-7	Real 2 as BYTE
	8-11	Real 3 as NIBBLE
	12-15	Bool 1-4
8	0-3	Bool 5-8
	4-15	Not used

12 Diagnostics

12.1 MOVIKIT® fieldbus monitor

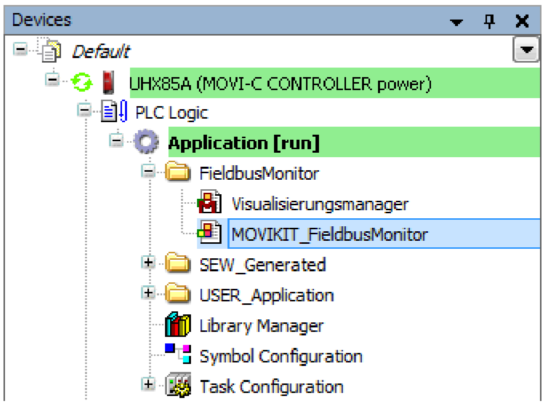
The MOVIKIT® fieldbus monitor is a tool in the IEC Editor for monitoring and controlling the fieldbus interface. The MOVIKIT® fieldbus monitor accesses solely the data of the fieldbus interface and represents the process input and process output data exchanged between the higher-level controller and the software module.

INFORMATION

You have to import the MOVIKIT® fieldbus monitor to being able to monitor and control the fieldbus interface. For further information, refer to the chapter "Importing the MOVIKIT® fieldbus monitor" (→ 🗎 44).

Do the following to open the tool:

1. In the MOVISUITE® project, open the context menu of the MOVI-C® CONTROLLER and select [IEC Editor] from the "Tools" submenu.
   ⇨ The IEC Editor opens.
2. Open the [Online] menu and click on [Login].
3. In the device tree, double-click the "MOVIKIT_FieldbusMonitor" node. (Path: `Default > SPS-Logik > Application [run] > FieldbusMonitor`)


Devices
Default
UHX85A (MOVI-C CONTROLLER power)
PLC Logic
Application [run]
FieldbusMonitor
Visualisierungsmanager
MOVIKIT_FieldbusMonitor
SEW_Generated
USER_Application
Library Manager
Symbol Configuration
Task Configuration

9007227578769547

⇨ The MOVIKIT® fieldbus monitor is opened in a new tab.

⚠ WARNING

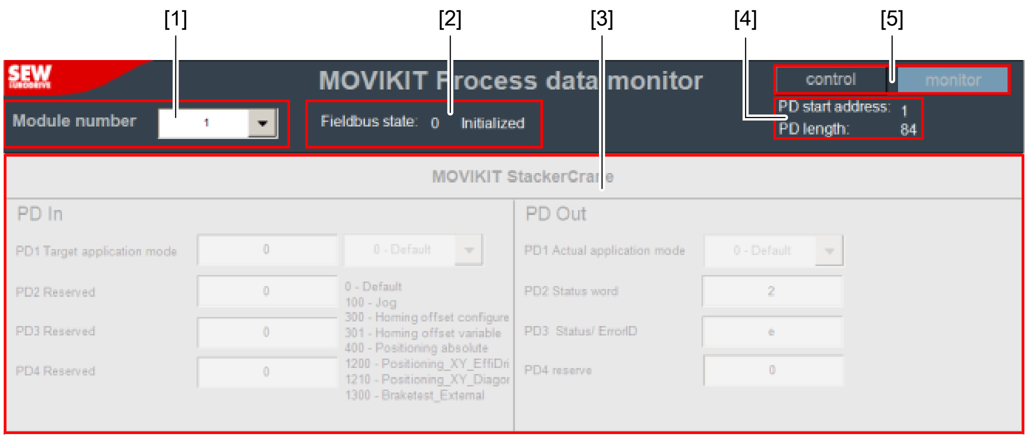
Unexpected system behavior if the communication between PC and MOVI-C® CONTROLLER is interrupted because the specified setpoints continue to take effect until the connection to the IEC Editor is interrupted automatically and the IEC Editor is logged off.

Death, severe injuries or damage to property

- In control mode, make sure that the drive can be stopped at any time by means of emergency stop measures.

12.1.1 User interface

The user interface consists of the following sections:



Module number 1

Fieldbus state: 0 Initialized

PD start address: 1
PD length: 84

MOVIKIT StackerCrane

PD In			PD Out	
PD1 Target application mode	0	0 - Default	PD1 Actual application mode	0 - Default
PD2 Reserved	0	0 - Default 100 - Jog 300 - Homing offset configure 301 - Homing offset variable 400 - Positioning absolute 1200 - Positioning_XY_EffiDri 1210 - Positioning_XY_Diagor 1300 - Braketest_External	PD2 Status word	2
PD3 Reserved	0		PD3 Status/ ErrorID	e
PD4 Reserved	0		PD4 reserve	0

18014426835536651

No.	Description
[1]	Number of the software module that is to be monitored or controlled. If several software modules are present, the sequence depends on the start address specified in the fieldbus configuration of the software module.
[2]	Status information of the fieldbus.
[3]	Visualization of the process data and control elements for controlling the bits
[4]	Start address and process data length of the software module selected under [1].
[5]	Button for toggling between "Monitor" and "Control". In "Control" mode, you can test functions of the software module without set-points from the higher-level controller. Control bits and process data words are directly applied to another edit box by pressing the enter key or clicking on it with the mouse.

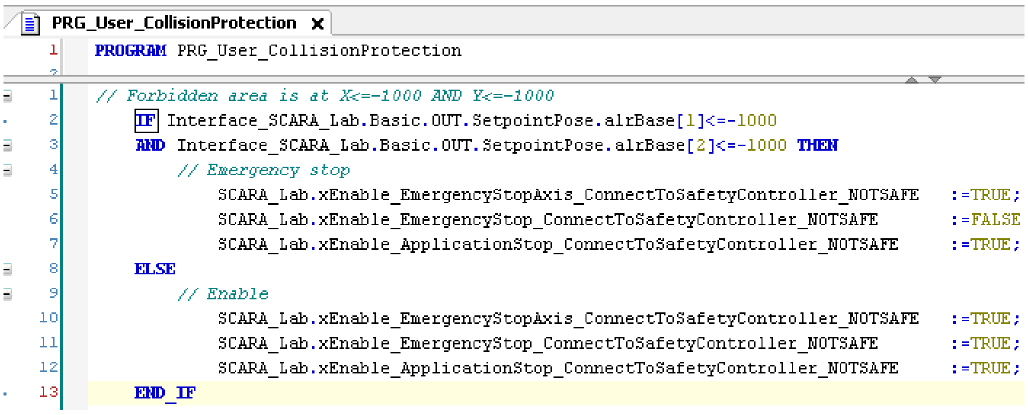
13 Application examples

13.1 Non-safe monitoring of robot behavior

For implementing non-safe monitoring (e.g. collision monitoring) of the robot behavior, you can use the enable mode that is always effective regardless of the authorized user (HMI, UI, PD, etc.).

For implementing collision monitoring by means of an IF condition, program the monitoring with the following behavior:

If the system violates the permitted behavior, emergency stop is triggered. If the system adheres to the permitted behavior, enable is issued.


PRG_User_CollisionProtection

```
PROGRAM PRG_User_CollisionProtection
```

```
// Forbidden area is at X<=-1000 AND Y<=-1000
    IF Interface_SCARA_Lab.Basic.OUT.SetpointPose.alrBase[1]<=-1000
    AND Interface_SCARA_Lab.Basic.OUT.SetpointPose.alrBase[2]<=-1000 THEN
        // Emergency stop
            SCARA_Lab.xEnable_EmergencyStopAxis_ConnectToSafetyController_NOTSAFE   :=TRUE;
            SCARA_Lab.xEnable_EmergencyStop_ConnectToSafetyController_NOTSAFE       :=FALSE
            SCARA_Lab.xEnable_ApplicationStop_ConnectToSafetyController_NOTSAFE     :=TRUE;
    ELSE
        // Enable
            SCARA_Lab.xEnable_EmergencyStopAxis_ConnectToSafetyController_NOTSAFE   :=TRUE;
            SCARA_Lab.xEnable_EmergencyStop_ConnectToSafetyController_NOTSAFE       :=TRUE;
            SCARA_Lab.xEnable_ApplicationStop_ConnectToSafetyController_NOTSAFE     :=TRUE;
    END_IF
```

9007222992193547

13.2 Switching to single axis control

Axis control must be switched from the robot to the single axes for referencing, non-referenced jogging, and for other single-axis functions. The robot must be in a safe state when switching control to the single axes.

Cancel enable before switching over control. If the robot is enabled and access to a lower-level single axis is requested, the robot enters fault state. The robot brakes to an emergency stop with the remaining axes axis-by-axis.

13.2.1 Switching from robot to single axis

1. Cancel enable for the robot (*xEnable_EmergencyStop* = "FALSE").
2. Check to see that the setpoint is no longer active (*xSetpointActive* = "FALSE")
3. Request access to the lower-level axis (*xGetAccessControl* = "TRUE").
   ⇨ Access to the single axis is granted (*xControlActive* = "TRUE").

⇨ The single axis can be controlled.

13.2.2 Switching from single axis to robot

1. Revoke the access right to the lower-level axis (*xGetAccessControl* = "FALSE").
2. Access to the single axis is revoked (*xControlActive* = "FALSE").

⇨ The robot can be controlled.

13.3 Using external selector switch for operating modes

If you want to specify the operating mode using an external selection switch (e.g. via the key switch on an operator panel) and not via a control source of the software module (UserInterface, RobotMonitor, fieldbus interface), do the following:

1. Create an interface variable that can be used to directly specify the operating mode of the robot. You can do this in the user program (User_PRG) in the IEC program:
   ⇨ `itfOperatingMode:SEW_MK_Robotics.SEW_IRobHPub.IOperating-Mode_ConnectToSafetyController_NOTSAFE:= MyRobot.fbOperat-ingModeHandler;`
   ⇨ "MyRobot" corresponds to the name of the robot node in MOVISUITE®.
2. Assign the externally specified operating mode to the variable:
   ⇨ `itfOperatingMode.eOperatingMode_ConnectToSafetyControl-ler_NOTSAFE:= SEW_MK_Robotics.SEW_IRobHPub.E_Operating-Mode.Automatic;`
   ⇨ `itfOperatingMode.eOperatingMode_ConnectToSafetyControl-ler_NOTSAFE:= SEW_MK_Robotics.SEW_IRobHPub.E_Operating-Mode.Manual_WithHighSpeed;`

INFORMATION

As soon as the operating mode has been selected once via this mechanism, the operating mode selected in the other control sources (UserInterface, RobotMonitor, fieldbus interface) is ignored and has no effect.

13.4 Using free function keys of the operator panel

The operator panel has function keys that you can use as required, e.g. to write digital outputs. The function keys are only transferred to the MOVI-C® CONTROLLER if the RobotMonitor is running on the control panel, is connected to the controller, and has access to the robot.

The states of the function keys can be read out in the following variables:

- `HMI_MyRobot.fbMonitor.InSignals.xUserFunctionKey1`
- `HMI_MyRobot.fbMonitor.InSignals.xUserFunctionKey2`

"MyRobot" corresponds to the name of the robot node assigned by the user in MOVISUITE®.

If the BOOLEAN variable is TRUE, this means that the button is currently pressed. If the BOOLEAN variable is FALSE, this means that the button is currently not pressed.

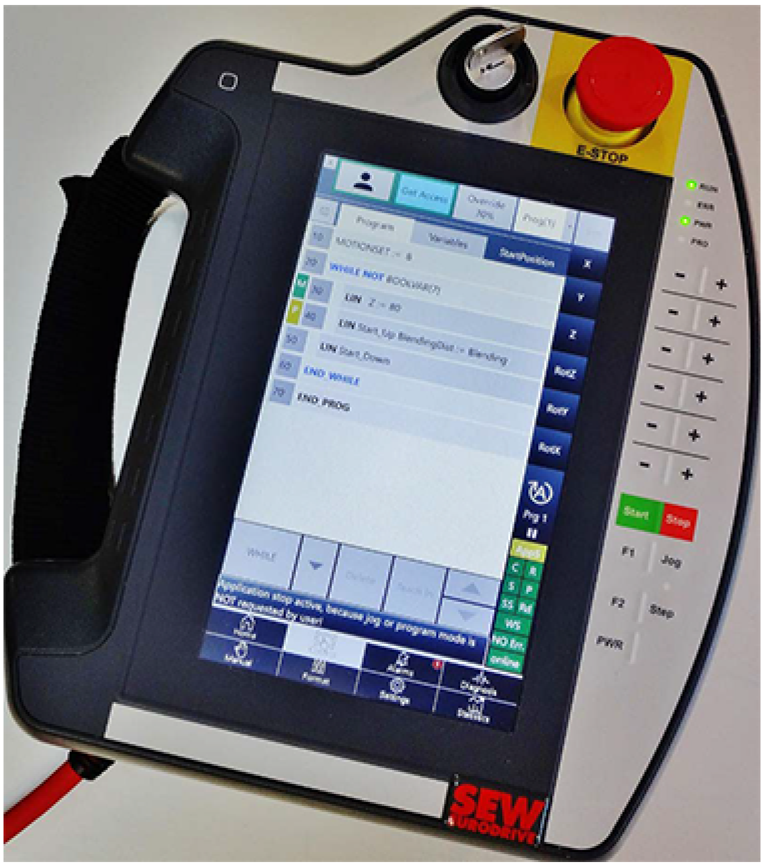
13.5 Combining the RobotMonitor with visualization

The RobotMonitor can run independently on a device with Windows operating system or can be used in combination with machine visualization or system visualization. For this purpose, the RobotMonitor can be started, for example, by pressing a key in the visualization. If the RobotMonitor has already been started, it will not be started again when the button is pressed again, but will be moved to the foreground. To avoid that the RobotMonitor covers the entire visualization and you can switch back to the visualization, you can specify the size of the RobotMonitor (in px) at startup using the following arguments:

– WindowWidth 600 – WindowHeight 920

If the RobotMonitor is not in the foreground, it does not use the hardware buttons of the operator panel and these can be used by the visualization.

In the following example, the RobotMonitor was combined with a PackML visualization. In the lower area, you can see the control elements of the visualization. The RobotMonitor is located above this area. Pressing a control in the visualization automatically moves the RobotMonitor into the background, and the visualization is displayed completely. Pressing a button in the visualization lets you start the RobotMonitor again and moves it into the foreground.



32292574603

13.6 Setting motion parameters

13.6.1 Speed

Rule of thumb:

- Distance/required target time (approach upwards from this value)

Correct value – Minimum of ...

- Maximum speed to which the mechanics, the tool, and the product may be subjected.
- Determine the speed on the tool that is possible with the maximum motor speeds (depending on the gear ratio, the position of the kinematics and consequently of the lever arms)

13.6.2 Acceleration

Rule of thumb:

- Speed × 10 (means the speed should be achieved in 100 ms)

Correct value – Minimum of ...

- Maximum acceleration to which the mechanics, the tool, and the product may be subjected.
- Determine which acceleration is possible on the tool using the maximum motor torques and gear unit input torques (this depends on the gear ratio, the position of the kinematics and consequently the lever arms, as well as on the inertias).

13.6.3 Jerk

Rule of thumb:

- Acceleration × 10 or, in the case of rigid mechanics, speed × 100 (means the acceleration should be reached in 100 ms or in 10 ms).

Correct value – Minimum of ...

- Maximum jerk to which the mechanics, the tool, and the product may be subjected.
- Maximum jerk at which the tool and the robot mechanics are not stimulated to oscillate.

13.7 Optimizing the cycle time

If the required cycle time is not reached, perform the following steps:

1. Make sure that identical poses are not approached twice.
2. Increase the jerk or the acceleration in the configuration. You should also do this for parameters that you assume are not used with the current operating mode, because your settings may affect the resulting motion parameters due to synchronization. See also "Sets of motion parameters" (→ 🗎 21) and "Setting motion parameters" (→ 🗎 161).
3. Use the signal *SEW_GVL_Internal.INSTANCENAME.fbMotionControl.Out.stProg.lrLimitedSpeedDueToCentrifugalAcc* to check whether the path speed is limited due to the centrifugal force acceleration (0 = not limited).
4. Increase the blending distance if this is possible with the interference contours. See chapter "Blending" (→ 🗎 19).

   ⇨ The cycle time is achieved.

   ⇨ If the cycle time is still not reached, contact SEW-EURODRIVE Service.

14 Fault management

14.1 Troubleshooting

14.1.1 The robot is not carrying out any motion

Example

The robot is not carrying out any motion.

Remedy

- Check and eliminate the general causes:
  - No access requested
  - No access granted
  - Wrong robot instance selected
  - No "Enable" is set
  - *Override* is set to the value "0"
  - A fault or error has occurred
  - Setpoints not active
  - Inverter without current supply
  - Inverter not connected
- If the robot is in jog mode, check and eliminate the following causes:
  - Operating mode is not "Manual high speed"
  - No jog signal activated
- If the robot is in program mode, check and eliminate the following causes:
  - Program pause activated
  - Program stop activated
  - No rising edge of program start detected
  - In "Manual high speed" operating mode
    Program start not activated
  - Target position(s) same as current position
- Check and eliminate additional causes:
  - Specified speed is too low
  - Specified acceleration is too low
  - Specified jerk is too low

14.1.2 Communication not possible between RobotMonitor and MOVI-C® CONTROLLER

Example

Communication with the robot is not possible. The message "NO Communication" is displayed.

Remedy

1. Check if "Start Communication" has been pressed.
2. Check if the set IP address of the MOVI-C® CONTROLLER is correct. You can find the set IP address in MOVISUITE® or in the IEC Editor.
3. Open the global variable list*SEW_GVL_Internal* in the IEC Editor.
4. Navigate to variable *HMI_InstanceName*.
5. Check this variable and the subordinate blocks *fbSimu3D*, *fbMonitor*, and *fb-ProgramEditor* for error messages.
6. In the device tree o the IEC Editor, double-click the object *Device*.
7. Open the "Log" tab.
8. Select the entry "MOS" as logger.
9. Click the [Update] button.
10. Check the messages for errors coming from the communication module of the software module, and eliminate the cause.
    ⇨ If the RobotMonitor is still not connected, contact SEW-EURODRIVE Service.

14.1.3 Robot program not found

Example

The robot program is not found. An error message of "File not Found: srlfiles/.../Prog1.crc. Please click on "Save" in robot monitor and restart PLC!" is displayed.

Remedy

✓ The "GetControl" signal has been activated and a feedback message of "Control Active" is displayed.

1. In the RobotMonitor, click the [Save programs and variables on memory card] diskette button.
2. Reset the error using the "Reset" signal.

14.2 Fault codes

14.2.1 Robotics errors

Dec	Hex.	Error	Library SEW_MOS_...
32 000	0x7d00	InterfaceNotValid	Robot_Itfs
32001	0x7d01	NoValidRobotKinematicModel_NrOfJoints	Robot_Itfs
32002	0x7d02	KinematicModelNotLinked	Robot_Itfs
32003	0x7d03	ActualRobotPoseInvalid	Robot_Itfs
32004	0x7d04	InternalInitializingStateWrong	Robot_Itfs
32005	0x7d05	ReplacementOfKinematicModelNotSupported	Robot_Itfs
32064	0x7d40	InterfaceNotValid	RobotHandler_Itfs
32065	0x7d41	EnableOutOfBoundaries_Safety	RobotHandler_Itfs
32066	0x7d42	InternalWrongEnableMode	RobotHandler_Itfs
32067	0x7d43	ProgramAndJogModeAtSameTime	RobotHandler_Itfs
32068	0x7d44	Just_SeqModeAuto_in_OpModeAuto	RobotHandler_Itfs
32069	0x7d45	ProgramChangeNotPossible	RobotHandler_Itfs
32070	0x7d46	InternalProgramInitNotPossible	RobotHandler_Itfs
32071	0x7d47	ProgramNumberChangedWhileExecution	RobotHandler_Itfs
32072	0x7d48	Inverter_NotConnected	RobotHandler_Itfs
32073	0x7d49	Inverter_SafeStop	RobotHandler_Itfs
32074	0x7d4a	Inverter_NotReady	RobotHandler_Itfs
32075	0x7d4b	Inverter_NotReady_TooLong	RobotHandler_Itfs
32076	0x7d4c	Inverter_NotReferenced	RobotHandler_Itfs
32077	0x7d4d	JogOnlyInManuelReducedSpeed	RobotHandler_Itfs
32078	0x7d4e	AxisGroupInverterModeWrong	RobotHandler_Itfs
32079	0x7d4f	AxisGroupInverterModeWrongWhileInterp	RobotHandler_Itfs
32080	0x7d50	AxisGroupInverterModeWrongTooLong	RobotHandler_Itfs
32081	0x7d51	InternalRefToMotionSetInvalid	RobotHandler_Itfs
32082	0x7d52	InternalResetProgOfMotionControlNotPossible	RobotHandler_Itfs
32083	0x7d53	ModuleNotNeeded	RobotHandler_Itfs
32084	0x7d54	ManualWithReducedSpeedNotSupported	RobotHandler_Itfs
32085	0x7d55	InternalSetUpNotPossible	RobotHandler_Itfs
32086	0x7d56	ChangeOfSimulationNotAllowed	RobotHandler_Itfs
32087	0x7d57	NoAxesLinkedAndNoSimulation	RobotHandler_Itfs
32088	0x7d58	ProgramInterpreterStateNotKnown	RobotHandler_Itfs
32089	0x7d59	ControlModeNotKnown	RobotHandler_Itfs
32090	0x7d5a	NoAccessToSubordinatedAxes	RobotHandler_Itfs
32096	0x7d60	RobotHMI_LinkRobotError_TargetIndexOutOfRange	RobotHMI_Itfs
32097	0x7d61	RobotHMI_LinkRobotError_RobotInterfaceInvalid	RobotHMI_Itfs

29180252/EN – 03/2020

Dec	Hex.	Error	Library SEW_MOS_...
32098	0x7d62	RobotHMI_SwitchRobotError_AtLeastOneInterfaceInvalid	RobotHMI_Itfs
32099	0x7d63	RobotHMI_SwitchRobotError_InvalidRobotInterface	RobotHMI_Itfs
32100	0x7d64	RobotHMI_SwitchRobotError_RequestedIndexOutOfRange	RobotHMI_Itfs
32101	0x7d65	RobotHMI_InvalidComState	RobotHMI_Itfs
32102		RobotHMI_ErrorInGuiCom	RobotHMI_Itfs
32103	0x7d67	RobotHMI_ReceiveToolVersionTimeOut	RobotHMI_Itfs
32104	0x7d68	RobotHMI_ConnectSubModulesTimeOut	RobotHMI_Itfs
32128	0x7d80	InvalidArrayIndex	RobotLanguage_Itfs
32129	0x7d81	InvalidInterface	RobotLanguage_Itfs
32130	0x7d82	InvalidReference	RobotLanguage_Itfs
32131	0x7d83	InvalidOperation	RobotLanguage_Itfs
32132	0x7d84	InvalidVarType	RobotLanguage_Itfs
32133	0x7d85	InvalidValueType	RobotLanguage_Itfs
32134	0x7d86	InvalidVarIndex	RobotLanguage_Itfs
32135	0x7d87	InvalidModuleType	RobotLanguage_Itfs
32136	0x7d88	InvalidBlock	RobotLanguage_Itfs
32137	0x7d89	InvalidProgramNumber	RobotLanguage_Itfs
32138	0x7d8a	NotImplementedInThisVersion	RobotLanguage_Itfs
32139	0x7d8b	ExecutionNotPossible	RobotLanguage_Itfs
32140	0x7d8c	InvalidOperand	RobotLanguage_Itfs
32141	0x7d8d	ReadDataError	RobotLanguage_Itfs
32142	0x7d8e	WriteDataError	RobotLanguage_Itfs
32143	0x7d8f	InvalidCircParameter	RobotLanguage_Itfs
32144	0x7d90	InvalidVariableValue	RobotLanguage_Itfs
32145	0x7d91	InvalidCommand	RobotLanguage_Itfs
32146	0x7d92	InvalidWordNumber	RobotLanguage_Itfs
32147	0x7d93	InvalidState	RobotLanguage_Itfs
32148	0x7d94	InvalidVarLength	RobotLanguage_Itfs
32149	0x7d95	InvalidInput	RobotLanguage_Itfs
32150	0x7d96	EventNumberOverflow	RobotLanguage_Itfs
32151	0x7d97	EventBuffer	RobotLanguage_Itfs
32152	0x7d98	EventExecution	RobotLanguage_Itfs
32160	0x7da0	RobLangEditor_ISRL_Data_NotConnected	RobotLanguage_Editor_Itfs
32161	0x7da1	RobLangEditor_IRobMoCOutForEditor_NotConnected	RobotLanguage_Editor_Itfs
32162	0x7da2	RobLangEditor_IProgramInterpreter_NotConnected	RobotLanguage_Editor_Itfs
32163	0x7da3	RobLangEditor_GuiCommError	RobotLanguage_Editor_Itfs
32164	0x7da4	RobLangEditor_InvalidRef_SRL_Data	RobotLanguage_Editor_Itfs

Dec	Hex.	Error	Library SEW_MOS_...
32165	0x7da5	RobLangEditor_InvalidRef_ProgInterpreterOut	RobotLanguage_Editor_Itfs
32169	0x7da9	RobLangEditor_ReceivedData_ProgIndexOutOfRange	RobotLanguage_Editor_Itfs
32170	0x7daa	RobLangEditor_ReceivedData_ProgTooLong	RobotLanguage_Editor_Itfs
32171	0x7dab	RobLangEditor_ReceivedData_SetBoolVarValue_Index-OutOfRange	RobotLanguage_Editor_Itfs
32172	0x7dac	RobLangEditor_ReceivedData_SetBoolVarName_Index-OutOfRange	RobotLanguage_Editor_Itfs
32173	0x7dad	RobLangEditor_ReceivedData_SetRealVarValue_Index-OutOfRange	RobotLanguage_Editor_Itfs
32174	0x7dae	RobLangEditor_ReceivedData_SetRealVarName_Index-OutOfRange	RobotLanguage_Editor_Itfs
32175	0x7daf	RobLangEditor_ReceivedData_SetPoseVarValue_Index-OutOfRange	RobotLanguage_Editor_Itfs
32177	0x7db1	RobLangEditor_ReceivedData_SetPoseVarName_Index-OutOfRange	RobotLanguage_Editor_Itfs
32178	0x7db2	RobLangEditor_ReceivedTelegramTypeNotSupported	RobotLanguage_Editor_Itfs
32179	0x7db3	RobLangEditor_Send_NameRequestNotSupported	RobotLanguage_Editor_Itfs
32180	0x7db4	RobLangEditor_Send_ValueRequestNotSupported	RobotLanguage_Editor_Itfs
32181	0x7db5	RobLangEditor_Send_RequestNotSupported	RobotLanguage_Editor_Itfs
32182	0x7db6	InterfaceError	RobotLanguage_Editor_Itfs
32183	0x7db7	RobLangEditor_InvalidRef_MotionSet	RobotLanguage_Editor_Itfs
32184	0x7db8	RobLangEditor_IndexOutOfRange	RobotLanguage_Editor_Itfs
32185	0x7db9	RobLangEditor_ResetComNotSuccessfull	RobotLanguage_Editor_Itfs
32186	0x7dba	RobLangEditor_ReceivedData_RequestedProgIndexOutO-fRange	RobotLanguage_Editor_Itfs
32187	0x7dbb	RobLangEditor_InvalidRef_ActivePathEvents	RobotLanguage_Editor_Itfs
32188	0x7dbc	RobLangEditor_InvalidRef_TriggeredPathEvents	RobotLanguage_Editor_Itfs
32192	0x7dc0	RobMon_InvalidInterface	RobotMonitor_Itfs
32193	0x7dc1	RobMon_GuiCommError	RobotMonitor_Itfs
32194	0x7dc2	RobMon_InvalidRef	RobotMonitor_Itfs
32195	0x7dc3	RobMon_InvalidOperatingModeReceived	RobotMonitor_Itfs
32196	0x7dc4	RobMon_InvalidJogCoordSysReceived	RobotMonitor_Itfs
32197	0x7dc5	RobMon_LinkRobotNotSuccessFull	RobotMonitor_Itfs
32224	0x7de0	InterfaceNotValid	RobotUI_Itfs
32225	0x7de1	SetMotionParamSet_OnlyIfInactive	RobotUI_Itfs
32256	0x7e00	Internal_SeeDetailsInLogbook	RobotMotionControl_Itfs
32257	0x7e01	ConfigurationReadFileFailed	RobotMotionControl_Itfs
32258	0x7e02	ConfigurationReadParameterFailed	RobotMotionControl_Itfs
32259	0x7e03	ConfigurationVersionNotSupported	RobotMotionControl_Itfs

29180252/EN – 03/2020

Dec	Hex.	Error	Library SEW_MOS_...
32260	0x7e04	ConfigurationNotValid	RobotMotionControl_Itfs
32261	0x7e05	ErrorResetOnlyPossibleAfterProgramStop	RobotMotionControl_Itfs
32272	0x7e10	LicenseNotAvailable	RobotMotionControl_Itfs
32273	0x7e11	LicenseNotRequested	RobotMotionControl_Itfs
32274	0x7e12	LicenseCorrupted	RobotMotionControl_Itfs
32275	0x7e13	JointEStopDecelerationOutOfRange	RobotMotionControl_Itfs
32276	0x7e14	JointEstopJerkOutOfRange	RobotMotionControl_Itfs
32277	0x7e15	JointLimitSwitchOutOfRange	RobotMotionControl_Itfs
32278	0x7e16	RotationEStopDecelerationOutOfRange	RobotMotionControl_Itfs
32279	0x7e17	RotationEStopJerkOutOfRange	RobotMotionControl_Itfs
32280	0x7e18	TranslationEStopDecelerationOutOfRange	RobotMotionControl_Itfs
32281	0x7e19	TranslationEStopJerkOutOfRange	RobotMotionControl_Itfs
32282	0x7e1a	CartesianLimitSwitchOutOfRange	RobotMotionControl_Itfs
32288	0x7e20	AxisJointTransformationNotLinked	RobotMotionControl_Itfs
32289	0x7e21	AxisJointTransformationInterfaceZero	RobotMotionControl_Itfs
32290	0x7e22	AxisJointTransformationSetupFailed	RobotMotionControl_Itfs
32291	0x7e23	AxisJointTransformationFailed	RobotMotionControl_Itfs
32292	0x7e24	KinematicModelNotLinked	RobotMotionControl_Itfs
32293	0x7e25	KinematicModelInterfaceZero	RobotMotionControl_Itfs
32294	0x7e26	ForwardKinematicsFailed	RobotMotionControl_Itfs
32295	0x7e27	ForwardKinematicsInterfaceCorrupted	RobotMotionControl_Itfs
32296	0x7e28	InverseKinematicsFailed	RobotMotionControl_Itfs
32297	0x7e29	InverseKinematicsInterfaceCorrupted	RobotMotionControl_Itfs
32304	0x7e30	MotionControlNotSetUp	RobotMotionControl_Itfs
32305	0x7e31	MotionSetNumberInvalid	RobotMotionControl_Itfs
32306	0x7e32	JointAccelerationOutOfRange	RobotMotionControl_Itfs
32307	0x7e33	JointDecelerationOutOfRange	RobotMotionControl_Itfs
32308	0x7e34	JointJerkOutOfRange	RobotMotionControl_Itfs
32309	0x7e35	JointVelocityOutOfRange	RobotMotionControl_Itfs
32310	0x7e36	RotationAccelerationOutOfRange	RobotMotionControl_Itfs
32311	0x7e37	RotationDecelerationOutOfRange	RobotMotionControl_Itfs
32312	0x7e38	RotationJerkOutOfRange	RobotMotionControl_Itfs
32313	0x7e39	RotationVelocityOutOfRange	RobotMotionControl_Itfs
32314	0x7e3a	TranslationAccelerationOutOfRange	RobotMotionControl_Itfs
32315	0x7e3b	TranslationDecelerationOutOfRange	RobotMotionControl_Itfs
32316	0x7e3c	TranslationJerkOutOfRange	RobotMotionControl_Itfs
32317	0x7e3d	TranslationVelocityOutOfRange	RobotMotionControl_Itfs

Dec	Hex.	Error	Library SEW_MOS_...
32320	0x7e40	MotionTypeInvalid	RobotMotionControl_Itfs
32321	0x7e41	BlendingDistanceNegative	RobotMotionControl_Itfs
32322	0x7e42	CircleCenterCoordinatesLeadingToRadiusZero	RobotMotionControl_Itfs
32323	0x7e43	CircleCenterCorrectionLargerThanThreshold	RobotMotionControl_Itfs
32324	0x7e44	CircleCenterCorrectionThresholdNotPositive	RobotMotionControl_Itfs
32325	0x7e45	CircleCenterEqualToStartOrEndPoint	RobotMotionControl_Itfs
32326	0x7e46	CircleCoordinatesUnclear	RobotMotionControl_Itfs
32327	0x7e47	CircleDirectionInvalid	RobotMotionControl_Itfs
32328	0x7e48	CircleRadiusZero	RobotMotionControl_Itfs
32329	0x7e49	CircleTypeInvalid	RobotMotionControl_Itfs
32330	0x7e4a	CircleTypeRadiusAngleRequiresPositiveRadius	RobotMotionControl_Itfs
32331	0x7e4b	PlaneInvalid	RobotMotionControl_Itfs
32332	0x7e4c	TangentPerpendicularToPlane	RobotMotionControl_Itfs
32336	0x7e50	CoordinateSystemNotSupported	RobotMotionControl_Itfs
32337	0x7e51	UserCoordinateSystemInterfaceZero	RobotMotionControl_Itfs
32338	0x7e52	UserCoordinateSystemInterfaceCorrupted	RobotMotionControl_Itfs
32339	0x7e53	UserCoordinateSystemTypeNotSupported	RobotMotionControl_Itfs
32340	0x7e54	UserCoordinateSystemVelocityOutOfRange	RobotMotionControl_Itfs
32341	0x7e55	UserCoordinateSystemAccelerationOutOfRange	RobotMotionControl_Itfs
32352	0x7e60	OutOfWorkspace	RobotMotionControl_Itfs
32368	0x7e70	NoSolutionFound	RobotMotionControl_Itfs
32384	0x7e80	PathEventInterfaceOfInterpreterZero	RobotMotionControl_Itfs
32385	0x7e81	PathEventQueryInterfaceNotSuccessful	RobotMotionControl_Itfs
32386	0x7e82	PathEventPositionBeforePathBegin	RobotMotionControl_Itfs
32387	0x7e83	PathEventTimeShiftExceedsRemainingTimeInMotion	RobotMotionControl_Itfs
32400	0x7E90	TouchProbeInterfaceZero	RobotMotionControl_Itfs
32401	0x7E91	TouchProbeQueryInterfaceNotSuccessful	RobotMotionControl_Itfs
32402	0x7E92	TouchProbePositioningTriggerBeforeContinue	RobotMotionControl_Itfs
32403	0x7E93	TouchProbeForLinearInterpolation	RobotMotionControl_Itfs
32404	0x7E94	TouchProbePosRequiresTransInMeasuringDirection	RobotMotionControl_Itfs
32405	0x7E95	TouchProbePositioningDistanceTooShort	RobotMotionControl_Itfs
32406	0x7E96	TouchProbePositioningEndInNonTangentialBlending	RobotMotionControl_Itfs
32407	0x7E97	TouchProbePositioningEndBeforeActualSetpoint	RobotMotionControl_Itfs
32408	0x7E98	TouchProbeMultipleSimultaneousContinue	RobotMotionControl_Itfs
32512	0x7f00	Internal_SeeDetailsInLogbook	RobotMathematics_Itfs
32544	0x7f20	Internal_SeeDetailsInLogbook	RobotProfileGeneration
32545	0x7f21	VelocityZeroOrNegative	RobotProfileGeneration

29180252/EN – 03/2020

Dec	Hex.	Error	Library SEW_MOS_...
32546	0x7f22	AccelerationZeroOrNegative	RobotProfileGeneration
32547	0x7f23	AccelerationGreaterThanRapidDeceleration	RobotProfileGeneration
32548	0x7f24	JerkZeroOrNegative	RobotProfileGeneration
32549	0x7f25	JerkGreaterThanRapidJerk	RobotProfileGeneration
32550	0x7f26	SetPositionNotReachableWithSettings	RobotProfileGeneration
32551	0x7f27	DecelerationZeroOrNegative	RobotProfileGeneration
32552	0x7f28	DecelerationGreaterThanRapidDeceleration	RobotProfileGeneration
32553	0x7f29	SetPositionZero	RobotProfileGeneration
32576	0x7f40	Internal_SeeDetailsInLogbook	RobotAxisJointTransformation_Itfs
32577	0x7f41	ConfigurationReadFileFailed	RobotAxisJointTransformation_Itfs
32578	0x7f42	ConfigurationReadParameterFailed	RobotAxisJointTransformation_Itfs
32579	0x7f43	ConfigurationVersionNotSupported	RobotAxisJointTransformation_Itfs
32580	0x7f44	ConfigurationNotValid	RobotAxisJointTransformation_Itfs
32581	0x7f45	AxisGroupPositioningInterpolatedNotLinked	RobotAxisJointTransformation_Itfs
32582	0x7f46	AxisGroupPositioningInterpolatedInterfaceZero	RobotAxisJointTransformation_Itfs
32583	0x7f47	ParameterValueInvalid	RobotAxisJointTransformation_Itfs
32584	0x7f48	TouchProbeValueMultidimensionalNotLinked	RobotAxisJointTransformation_Itfs
32585	0x7f49	TouchProbeValueMultidimensionalInterfaceZero	RobotAxisJointTransformation_Itfs
32608	0x7f60	Internal_SeeDetailsInLogbook	RobotKinematicModel_Itfs
32609	0x7f61	ConfigurationReadFileFailed	RobotKinematicModel_Itfs
32610	0x7f62	ConfigurationReadParameterFailed	RobotKinematicModel_Itfs
32611	0x7f63	ConfigurationVersionNotSupported	RobotKinematicModel_Itfs
32612	0x7f64	ConfigurationNotValid	RobotKinematicModel_Itfs
32613	0x7f65	ConfiguredModelDiffersFromLinked	RobotKinematicModel_Itfs
32614	0x7f66	ParameterValueOutOfRange	RobotKinematicModel_Itfs
32624	0x7f70	PositionNotPossible	RobotKinematicModel_Itfs
32625	0x7f71	OrientationNotPossible	RobotKinematicModel_Itfs
32626	0x7f72	PoseNotReachable	RobotKinematicModel_Itfs
32627	0x7f73	CartesianAssignmentNotAllowed	RobotKinematicModel_Itfs
32628	0x7f74	ConstellationNotSupported	RobotKinematicModel_Itfs

Dec	Hex.	Error	Library SEW_MOS_...
32629	0x7f75	ConstellationNotAllowedInPose	RobotKinematicModel_Itfs
32800	0x8020	InvalidInterface	RobotTouchprobe_Itfs
32801	0x8021	SEW_IRobLangNotLinked	RobotTouchprobe_Itfs
32802	0x8022	SEW_ITouchProbeProfileNotLinked	RobotTouchprobe_Itfs
32803	0x8023	SEW_ITransformNotLinked	RobotTouchprobe_Itfs
32804	0x8024	SEW_IExecuteEventNotLinked	RobotTouchprobe_Itfs
32805	0x8025	LinkableModuleNotUsed	RobotTouchprobe_Itfs
32806	0x8026	InvalidSource	RobotTouchprobe_Itfs
32807	0x8027	InvalidMode	RobotTouchprobe_Itfs
32808	0x8028	InvalidLevel	RobotTouchprobe_Itfs
32809	0x8029	InvalidMeasuringDirection	RobotTouchprobe_Itfs
32810	0x802A	InvalidSourceVarIndex	RobotTouchprobe_Itfs
32811	0x802B	SeveralTouchProbesAtSameTime	RobotTouchprobe_Itfs
32812	0x802C	CounterOverflowTouchProbeCounter	RobotTouchprobe_Itfs
32813	0x802D	CounterOverflowActiveTPEvent_ChangeCounter	RobotTouchprobe_Itfs
32814	0x802E	ErrorResetOnlyPossibleAfterProgramStop	RobotTouchprobe_Itfs
32815	0x802F	TransformationNotSuccessful	RobotTouchprobe_Itfs
32816	0x8030	ExecuteEventNotSuccessful	RobotTouchprobe_Itfs
32817	0x8031	ContinueBeforeRegister	RobotTouchprobe_Itfs
33024	0x8100	InterfaceNotValid	RobotPD
33025	0x8101	ErrorReadingConfigFile	RobotPD
33026	0x8102	ErrorReadingParameter	RobotPD
33027	0x8103	RequiredLengthOfProfileToLarge	RobotPD
33028	0x8104	uiProcessDataProfile_NoConsistency	RobotPD
33029	0x8105	InternalProfileLengthCheckDefect	RobotPD
33030	0x8106	InvalidReferenceOnProgramVariables	RobotPD
33031	0x8107	InternalConfigCheckInvalid	RobotPD
33032	0x8108	ValueOutOfBoundsOfDatatype	RobotPD
33033	0x8109	WrongConfiguredSizeOnBusOfReal	RobotPD
33280	0x8200	Simu3D_InvalidInterface	Simu3D_Itfs
33281	0x8201	Simu3D_InvalidRef	Simu3D_Itfs
33282	0x8202	Simu3D_GuiCommError	Simu3D_Itfs
33283	0x8203	Simu3D_AddUserCS_IndexOutOfRange	Simu3D_Itfs
33284	0x8204	Simu3D_UserCS_GetTransformNotSuccessful	Simu3D_Itfs
33285	0x8205	Simu3D_NumberOfTransmittedKinParsToSmall	Simu3D_Itfs
33286	0x8206	Simu3D_TimeoutModelAcknowledge	Simu3D_Itfs
33287	0x8207	Simu3D_TimeoutConfigAcknowledge	Simu3D_Itfs

29180252/EN – 03/2020

14.2.2 Robotics messages

Dec	Hex.	Message	Library SEW_MOS_...
97600	0x17d40	WhishedEnableModeNotPossible	RobotHandler_Itfs
97601	0x17d41	WaitingForActiveControlMode	RobotHandler_Itfs
97602	0x17d42	WaitingForEnable	RobotHandler_Itfs
97609	0x17d49	Inverter_SafeStop	RobotHandler_Itfs
97610	0x17d4a	RisingEdgeOfStartIgnoredBecauseProgramPause	RobotHandler_Itfs
97611	0x17d4b	RisingEdgeOfStartIgnoredBecauseProgramStop	RobotHandler_Itfs
97612	0x17d4c	RisingEdgeOfStartIgnoredBecauseProgramNotYetInitial-ized	RobotHandler_Itfs
97613	0x17d4d	RisingEdgeOfStartIgnoredBecauseRobotNotReadyTo-Move	RobotHandler_Itfs
97614	0x17d4e	RisingEdgeOfStartIgnoredBecauseProgramIsBeeingEx-ecuted	RobotHandler_Itfs
97632	0x17d60	RobotHMI_MsgHmiConnected	RobotHMI_Itfs
97635	0x17d63	RobotHMI_ReceiveToolVersionTimeOut	RobotHMI_Itfs
97636	0x17d64	RobotHMI_ConnectionLossDuringSubmodulesConnect	RobotHMI_Itfs
97792	0x17e00	Internal_SeeDetailsInLogbook	RobotMotionControl_Itfs
97808	0x17e10	JogKinematicJointDOFNotExisting	RobotMotionControl_Itfs
97809	0x17e11	JogAdditionalJointDOFNotExisting	RobotMotionControl_Itfs
97810	0x17e12	JogCartesianDOFNotExisting	RobotMotionControl_Itfs
97811	0x17e13	JogKinematicJointStoppedDueToSWLS	RobotMotionControl_Itfs
97812	0x17e14	JogAdditionalJointStoppedDueToSWLS	RobotMotionControl_Itfs
97813	0x17e15	JogCartesianStoppedDueToSWLS	RobotMotionControl_Itfs
97824	0x17e20	LimitedSpeedDueToCentrifugalAcc	RobotMotionControl_Itfs
97888	0x17e60	OutOfWorkspace	RobotMotionControl_Itfs
97920	0x17e80	PathEventRegistrationDelayedMaximumNumberReached	RobotMotionControl_Itfs
97921	0x17e81	PathEventTimeShiftDelaysMotion	RobotMotionControl_Itfs
97922	0x17e82	PathEventTimeShiftElapsedNotInProgramMode	RobotMotionControl_Itfs
97923	0x17e83	PathEventStillActiveAfterMotionEnd	RobotMotionControl_Itfs
98080	0x17f20	UseOfIncreasedDeceleration	RobotProfileGeneration
98081	0x17f21	UseOfIncreasedJerk	RobotProfileGeneration
98082	0x17f22	UseOfIncreasedDecelerationAndIncreasedJerk	RobotProfileGeneration
98112	0x17f40	Internal_SeeDetailsInLogbook	RobotAxisJointTransforma-tion_Itfs
98144	0x17f60	Internal_SeeDetailsInLogbook	RobotKinematicModel_Itfs
98336	0x18020	PositioningWhileMultipleMode	RobotTouchprobe_Iffs
98816	0x18200	Simu3D_TimeoutModelAcknowledge	Simu3D_Itfs
98817	0x18201	Simu3D_TimeoutConfigAcknowledge	Simu3D_Itfs

Dec	Hex.	Message	Library SEW_MOS_...
98818	0x18202	Simu3D_FailedFBCallsInGuiCom	Simu3D_Itfs

14.2.3 Subordinate software module errors

Dec	Hex.	Error	Library SEW_MOS_...
25664	0x6440	eSEW_LicMgr_GetInfo	IECLicenseManager
25665	0x6441	eSEW_LicMgr_GetInfo_PerfClass	IECLicenseManager
25666	0x6442	eSEW_LicMgr_CheckAndReportRuntime	IECLicenseManager
25667	0x6443	eSEW_LicMgr_SecretChallenge	IECLicenseManager
25668	0x6444	eSEW_LicMgr_NoRuntime	IECLicenseManager
25669	0x6445	eSEW_LicMgr_NoValidRuntime	IECLicenseManager
25670	0x6446	eSEW_LicMgr_CheckLicense	IECLicenseManager
25671	0x6447	eSEW_LicMgr_ConsumeLicense	IECLicenseManager
25672	0x6448	eSEW_LicMgr_ReportMissingLicense	IECLicenseManager
25673	0x6449	eSEW_LicMgr_FileReloadWatcher	IECLicenseManager
25674	0x644a	eSEW_LicMgr_ConfirmToken	IECLicenseManager
25696	0x6460	InterfaceNotValid	ErrorHandling_Itfs
25697	0x6461	SubordinatedFBArrayFull	ErrorHandling_Itfs
25698	0x6462	ErrorIDZero	ErrorHandling_Itfs
25699	0x6463	MessageIDZero	ErrorHandling_Itfs
25700	0x6464	FBHasAlreadyAnSuperordinatedFB	ErrorHandling_Itfs
25701	0x6465	SubordinatedFBAlreadyAdded	ErrorHandling_Itfs
25702	0x6466	MessageIDisEqualToErrorID	ErrorHandling_Itfs
25703	0x6467	MessageIDEqualvocal	ErrorHandling_Itfs
25704	0x6468	CompletionOfAdditionalTextFailed	ErrorHandling_Itfs
25728	0x6480	MessageBufferFull	LoggingAdapter_Itfs
26112	0x6600	ConfigFileNotFound	AxisConfig_Itfs
26113	0x6601	ConfigFileNotOpened	AxisConfig_Itfs
26114	0x6602	ConfigFileNotClosed	AxisConfig_Itfs
26115	0x6603	ConfigDataNotRead	AxisConfig_Itfs
26116	0x6604	ConfigParameterNotFound	AxisConfig_Itfs
26117	0x6605	ConfigParameterNotValid	AxisConfig_Itfs
27136	0x6a00	DeviceError	DeviceAdapter_Itfs
27137	0x6a01	DeviceHandlerError	DeviceAdapter_Itfs
27168	0x6a20	eSEW_OSCHandler_InterfaceNotValid	DeviceAdapter_OSC71B
27169	0x6a21	eSEW_OSCHandler_TimeoutSendSync	DeviceAdapter_OSC71B
27170	0x6a22	eSEW_OSCHandler_InvalidCanId	DeviceAdapter_OSC71B
27171	0x6a23	eSEW_OSCHandler_InvalidInteface	DeviceAdapter_OSC71B

Dec	Hex.	Error	Library SEW_MOS_...
27172	0x6a24	eSEW_OSCHandler_Error	DeviceAdapter_OSC71B
27173	0x6a25	eSEW_OSCHandler_InvalidDataAddress	DeviceAdapter_OSC71B
27174	0x6a26	eSEW_OSCHandler_CanNotReady	DeviceAdapter_OSC71B
27175	0x6a27	eSEW_OSCHandler_DeviceNotReady	DeviceAdapter_OSC71B
27176	0x6a28	eSEW_OSCHandler_NotInitialized	DeviceAdapter_OSC71B
27177	0x6a29	eSEW_OSCHandler_CanIdAlreadyRegistered	DeviceAdapter_OSC71B
27178	0x6a2a	eSEW_OSCHandler_InvalidCanAddress	DeviceAdapter_OSC71B
27179	0x6a2b	eSEW_OSCHandler_NumberCanPDOToLarge	DeviceAdapter_OSC71B
27180	0x6a2c	eSEW_OSCHandler_TimeoutSendPDO	DeviceAdapter_OSC71B
27181	0x6a2d	eSEW_OSCHandler_TimeoutRequestSDO	DeviceAdapter_OSC71B
27182	0x6a2e	eSEW_OSCHandler_TimeoutHeartbeat	DeviceAdapter_OSC71B
27183	0x6a2f	eSEW_OSCHandler_EmergencyTelegram	DeviceAdapter_OSC71B
27184	0x6a30	eSEW_OSCHandler_SDOAbortCode	DeviceAdapter_OSC71B
27185	0x6a31	eSEW_OSCHandler_TimeoutRequestSBUSParam	DeviceAdapter_OSC71B
27186	0x6a32	eSEW_OSCHandler_MVLAbortCode	DeviceAdapter_OSC71B
27187	0x6a33	eSEW_OSCHandler_TimeoutMvlPD	DeviceAdapter_OSC71B
27188	0x6a34	eSEW_OSCHandler_InvalidPDLenght	DeviceAdapter_OSC71B
27189	0x6a35	eSEW_OSCHandler_InvalidPDSegment	DeviceAdapter_OSC71B
27712	0x6c40	eSEW_ExSourc_GetConfig	SyncExtSource_Itfs
27713	0x6c41	eSEW_ExSourc_NotLinked_SendObject	SyncExtSource_Itfs
27714	0x6c42	eSEW_ExSourc_ValueOutOfLimits	SyncExtSource_Itfs
27715	0x6c43	eSEW_ExSourc_VZ1Filter	SyncExtSource_Itfs
27716	0x6c44	eSEW_ExSourc_AverageFilter	SyncExtSource_Itfs
27717	0x6c45	eSEW_ExSourc_DeltaValueToLarge	SyncExtSource_Itfs
27718	0x6c46	eSEW_ExSourc_NotLinked_Persistent	SyncExtSource_Itfs
27719	0x6c47	eSEW_ExSourc_NotLinked_SyncExtSourceVal-ues	SyncExtSource_Itfs
27720	0x6c48	eSEW_ExSourc_NotLinked_ConfigData	SyncExtSource_Itfs
28224	0x6e40	eSEW_ParamHandler_Request	ParameterHandler
28225	0x6e41	eSEW_ParamHandler_Response	ParameterHandler
28226	0x6e42	eSEW_ParamHandler_NoDeviceLink	ParameterHandler
28672	0x7000	eSEW_FH_ASM_Result	FileHandler
28673	0x7001	eSEW_FH_TimeOut	FileHandler
28674	0x7002	eSEW_FH_FileNotHere	FileHandler
28675	0x7003	eSEW_FH_RTS_Result	FileHandler
28704	0x7020	eSEW_PLCGetInfo	Util
28705	0x7021	eSEW_DeltaValueToLarge	Util

Dec	Hex.	Error	Library SEW_MOS_...
28706	0x7022	eSEW_VZ1Filter	Util
28707	0x7023	eSEW_AverageFilter	Util
28708	0x7024	eSEW_ModuloMax_ModuloMin	Util
28709	0x7025	eSEW_ValueOutOfLimits	Util
28710	0x7026	eSEW_NotInitialized	Util
30208	0x7600	SoftwareLimitSwitchNotValid	InterpolationModes_Itfs
30209	0x7601	ModuloLimitsNotValid	InterpolationModes_Itfs
30210	0x7602	PresetPositionNotValid	InterpolationModes_Itfs
30211	0x7603	ReferenceOffsetNotValid	InterpolationModes_Itfs
30212	0x7604	ReferenceOffsetOutOfModuloLimit	InterpolationModes_Itfs
30213	0x7605	HomingStartPositionNotValid	InterpolationModes_Itfs
30214	0x7606	ModuloModeNotSupported	InterpolationModes_Itfs
30215	0x7607	AxisNotReferenced	InterpolationModes_Itfs
30216	0x7608	TargetPositionNotValid	InterpolationModes_Itfs
30217	0x7609	TravelDistanceNotValid	InterpolationModes_Itfs
30218	0x760a	TargetPositionOutOfSoftwareLimitSwitch	InterpolationModes_Itfs
30219	0x760b	VelocityStopPositionNotValid	InterpolationModes_Itfs
30220	0x760c	MasterResolutionOutsideLimits	InterpolationModes_Itfs
30221	0x760d	MasterModuloOutsideLimits	InterpolationModes_Itfs
30222	0x760e	SlaveModuloOutsideLimits	InterpolationModes_Itfs
30223	0x760f	NumeratorDenominatorOutsideLimits	InterpolationModes_Itfs
30224	0x7610	MasterPositionOutsideLimits	InterpolationModes_Itfs
30225	0x7611	MasterTimeBaseOutsideLimits	InterpolationModes_Itfs
30226	0x7612	SlaveTimeBaseOutsideLimits	InterpolationModes_Itfs
30227	0x7613	SoftwareLimitPositive_Reached	InterpolationModes_Itfs
30228	0x7614	SoftwareLimitNegative_Reached	InterpolationModes_Itfs
30229	0x7615	ApplicationLimitDeceleration	InterpolationModes_Itfs
30230	0x7616	ApplicationLimitAcceleration	InterpolationModes_Itfs
30231	0x7617	ApplicationLimitVelocityPositive	InterpolationModes_Itfs
30232	0x7618	ApplicationLimitVelocityNegative	InterpolationModes_Itfs
30233	0x7619	InterfaceNotLinked	InterpolationModes_Itfs
30234	0x761a	ProfileGeneratorInternalError	InterpolationModes_Itfs
30235	0x761b	ReadConfigDataFailed	InterpolationModes_Itfs
30236	0x761c	InvalidLicence	InterpolationModes_Itfs
30464	0x7700	InterfaceNotLinked	AntiSloshInterpolation
30465	0x7701	WrongInputParameter	AntiSloshInterpolation
30466	0x7702	ReadConfigDataFailed	AntiSloshInterpolation

Dec	Hex.	Error	Library SEW_MOS_...
30467	0x7703	InvalidConfigParameter	AntiSloshInterpolation
30495	0x771f	UndefinedError	AntiSloshInterpolation
30721	0x7801	OutOfLagErrorWindow	Controller_Itfs
30722	0x7802	EC_EncoderIsNotConnected	Controller_Itfs
30723	0x7803	PositionNotValid	Controller_Itfs
30724	0x7804	InterfaceError	Controller_Itfs
30725	0x7805	QueryFailed	Controller_Itfs
30726	0x7806	GearRatioIsZero	Controller_Itfs
30727	0x7807	IndexOutOfBounds	Controller_Itfs
30728	0x7808	InvalidValueForControlloop	Controller_Itfs
30729	0x7809	InvalidValueForActValueEvaluation	Controller_Itfs
30730	0x780a	OutOfSkewErrorWindow	Controller_Itfs
30731	0x780b	WrongReferencedBitOnStatusWord	Controller_Itfs
30732	0x780c	WrongActualPositionSource	Controller_Itfs
30733	0x780d	ExternalEncoderActivatedOnMDD	Controller_Itfs
30734	0x780e	NoExternalEncoderSelected	Controller_Itfs
30735	0x780f	NoCombinedEncoderEvaluationSelected	Controller_Itfs
30736	0x7810	TorqueLevelingPGainMaxIsZero	Controller_Itfs
30737	0x7811	TooManyAssociatedAGMembers	Controller_Itfs
30738	0x7812	ConfirmTokenFailed	Controller_Itfs
30739	0x7813	InsufficientExternalEncoder	Controller_Itfs
30849	0x7881	InterfaceError	MultiAxisController_Itfs
30850	0x7882	NoAccessToAxes	MultiAxisController_Itfs
30851	0x7883	NoAccessReturnPossible	MultiAxisController_Itfs
30852	0x7884	IndexOutOfBounds	MultiAxisController_Itfs
30853	0x7885	WrongTravelTypeForReadjustment	MultiAxisController_Itfs
30854	0x7886	MoreThanTwoAxesForReadjustment	MultiAxisController_Itfs
30855	0x7887	WrongLSOperationForReadjustment	MultiAxisController_Itfs
30856	0x7888	QueryFailed	MultiAxisController_Itfs
30857	0x7889	ConfirmTokenFailed	MultiAxisController_Itfs
30858	0x788a	MissingCascadingLicense	MultiAxisController_Itfs
30859	0x788b	MissingFourAxesLicense	MultiAxisController_Itfs
30860	0x788c	MissingTorqueSkewingLicense	MultiAxisController_Itfs
30861	0x788d	WrongPriorityLicenseActivated	MultiAxisController_Itfs
30862	0x788e	ParameterChannelTimeOut	MultiAxisController_Itfs
30863	0x788f	NotAllAxesAreActivated	MultiAxisController_Itfs
30865	0x7891	HomingLSReversed	MultiAxisController_Itfs

Dec	Hex.	Error	Library SEW_MOS_...
30866	0x7892	HomingSafetyTimeExpired	MultiAxisController_Itfs
30867	0x7893	HomingSafetyDistancePassed	MultiAxisController_Itfs
30868	0x7894	HomingStartPositionNotValid	MultiAxisController_Itfs
30869	0x7895	TravelTypeDeactivated	MultiAxisController_Itfs
31232	0x7a00	InvalidInterface	AxisGroupBasic_Itfs
31233	0x7a01	NoAccessToAxes	AxisGroupBasic_Itfs
31234	0x7a02	NoAccessReturnPossible	AxisGroupBasic_Itfs
31235	0x7a03	UI_Init_Not_IBasic	AxisGroupBasic_Itfs
31236	0x7a04	NoAxesLinked	AxisGroupBasic_Itfs
31237	0x7a05	ArrayIndexOutOfBounds	AxisGroupBasic_Itfs
31238	0x7a06	MaxNumberOfAxesExceeded	AxisGroupBasic_Itfs
31264	0x7a20	InterfaceNotValid	AxisGroupPositioningInterpol-ated_Itfs
31265	0x7a21	OutOfBound_LinkingOfAxes	AxisGroupPositioningInterpol-ated_Itfs
31266	0x7a22	TouchprobeCounterOverflow	AxisGroupPositioningInterpol-ated_Itfs
31267	0x7a23	TouchprobeCounterTimeout	AxisGroupPositioningInterpol-ated_Itfs
31296	0x7a40	LSPositiveInverterConfigured	LimitSwitchEvaluation_Itfs
31297	0x7a41	LSNegativeInverterConfigured	LimitSwitchEvaluation_Itfs
31298	0x7a42	PositiveLSHit	LimitSwitchEvaluation_Itfs
31299	0x7a43	NegativeLSHit	LimitSwitchEvaluation_Itfs
31300	0x7a44	InterfaceError	LimitSwitchEvaluation_Itfs
31301	0x7a45	QueryFailed	LimitSwitchEvaluation_Itfs
31302	0x7a46	IndexOutOfBounds	LimitSwitchEvaluation_Itfs
31303	0x7a47	NoAxesConnected	LimitSwitchEvaluation_Itfs
31304	0x7a48	LimitSwitchReversed	LimitSwitchEvaluation_Itfs
31305	0x7a49	BothLSHit	LimitSwitchEvaluation_Itfs
36960	0x9060	ModeNotValid	ModeAdministrator

29180252/EN – 03/2020

14.2.4 Subordinate software module messages

Dec	Hex.	Message	Library SEW_MOS_...
91200	0x16440	eSEW_LicMgr_RepMisSiLic	IECLicenseManager
91201	0x16441	eSEW_LicMgr_RepMisPerLic	IECLicenseManager
91202	0x16442	eSEW_LicMgr_RepMisRunLic	IECLicenseManager
91203	0x16443	eSEW_LicMgr_TrialLicenseActive	IECLicenseManager
91204	0x16444	eSEW_LicMgr_TrialLicenseExpired	IECLicenseManager
91205	0x16445	eSEW_LicMgr_LicenseActive	IECLicenseManager
91206	0x16446	eSEW_LicMgr_DualUseLicenseActive	IECLicenseManager
91207	0x16447	eSEW_LicMgr_NotTestableLicenseActive	IECLicenseManager
91208	0x16448	eSEW_LicMgr_RuntimeTrialLicenseActive	IECLicenseManager
91209	0x16449	eSEW_LicMgr_RuntimeTrialLicenseExpired	IECLicenseManager
91210	0x1644a	eSEW_LicMgr_TrialLicenseActivated	IECLicenseManager
91232	0x16460	InterfaceNotValid	ErrorHandling_Itfs
91233	0x16461	ErrorHandling_NotYetInitialized	ErrorHandling_Itfs
91264	0x16480	LoggingNotSuccessful	LoggingAdapter_Itfs
91265	0x16481	LogbookOpeningFailed	LoggingAdapter_Itfs
93248	0x16c40	eSEW_ExSourc_OffOnLimit	SyncExtSource_Itfs
94208	0x17000	eSEW_FH_BufferTooShort	FileHandler
94209	0x17001	eSEW_FH_CancleJobNotAllowed	FileHandler
96800	0x17a20	TouchprobeInterfaceNotValid	AxisGroupPositioningInterpolated_Itfs
103920	0x195f0	DynamicValueTooLarge	DeviceAdapter_Itfs
103921	0x195f1	DynamicValueTooSmall	DeviceAdapter_Itfs
103922	0x195f2	InverterWarning	DeviceAdapter_Itfs

Index

A

Access rights 132

B

Blending 19

C

Communication
- Fieldbus 23
- System bus 23

Concurrent access 132
Configuration 32
Constellation 17
Copyright notice 9

D

Decimal separator 9

E

Embedded safety notes 9

F

Fault management 163
Fieldbus 23

H

Hazard symbols
- Meaning 9

K

Kinematic models 16
- CARTESIAN GANTRY 16
- DELTA 16
- MIXED 17
- ROLLER GANTRY 16
- SCARA 16
- TRIPOD 16

M

Monitor access 132
Motion path 18
Motion profile 20

N

Notes
- Designation in the documentation 8
- Meaning of the hazard symbols 9

O

Override 21
Overview
- Kinematic models 16

P

Path segment 18
PD monitor 156
- Insert 44

Product names 9
Project planning 24

R

Rapid stop 22
Rights to claim under limited warranty 9

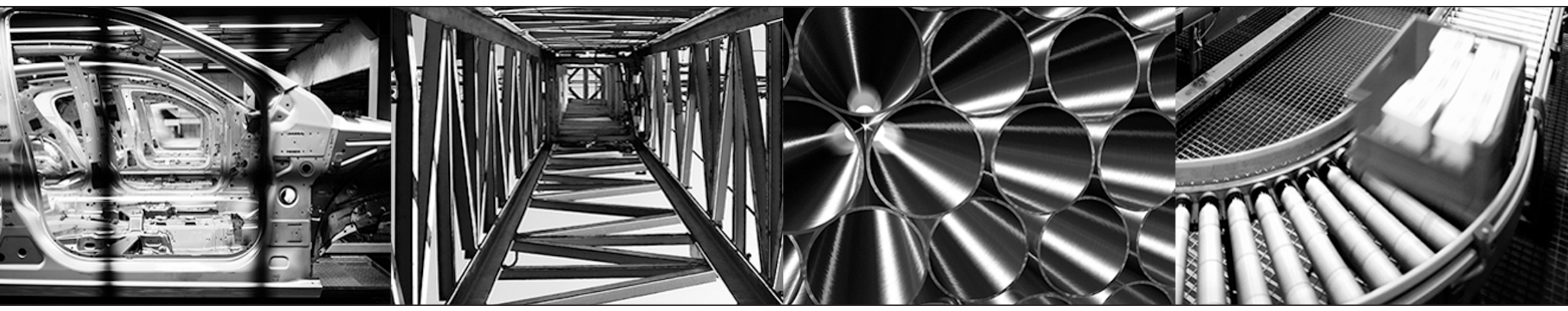
S

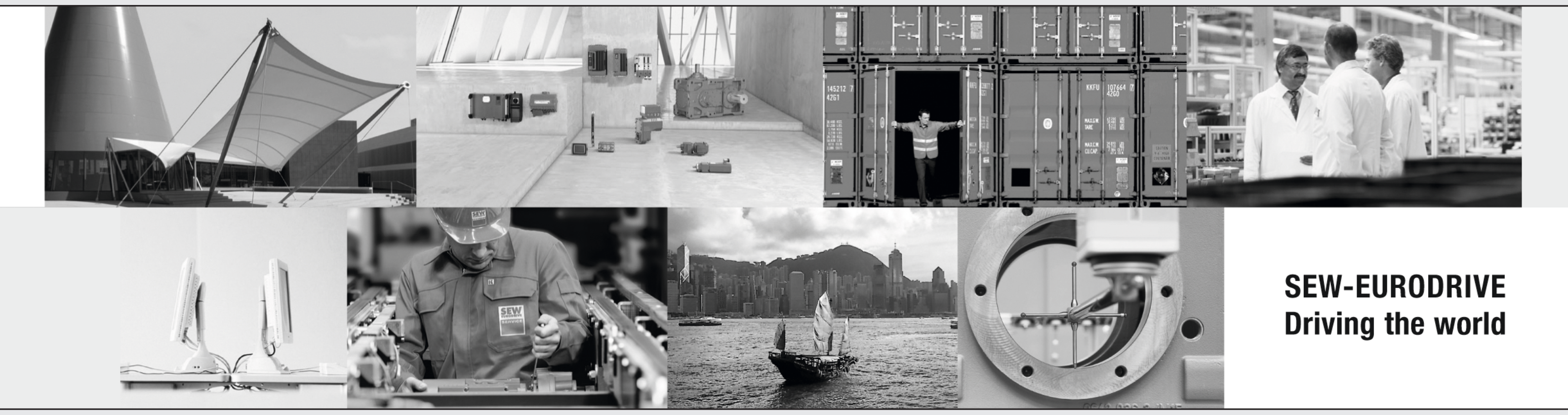
Safety notes
- Bus systems 11
- Designation in the documentation 8
- Meaning of the hazard symbols 9
- Preliminary information 11
- Structure of embedded 9
- Structure of section-related 8

Section-related safety notes 8
Segment parameters 21
Short designation 10
Signal words in safety notes 8
System bus 23

T

Target group 11
Trademarks 9
Types of interpolation
- Axis interpolation 18
- Path interpolation 18





SEW-EURODRIVE GmbH & Co KG
Ernst-Blickle-Str. 42
76646 BRUCHSAL
GERMANY
Tel. +49 7251 75-0
Fax +49 7251 75-1970
sew@sew-eurodrive.com
➞ **www.sew-eurodrive.com**