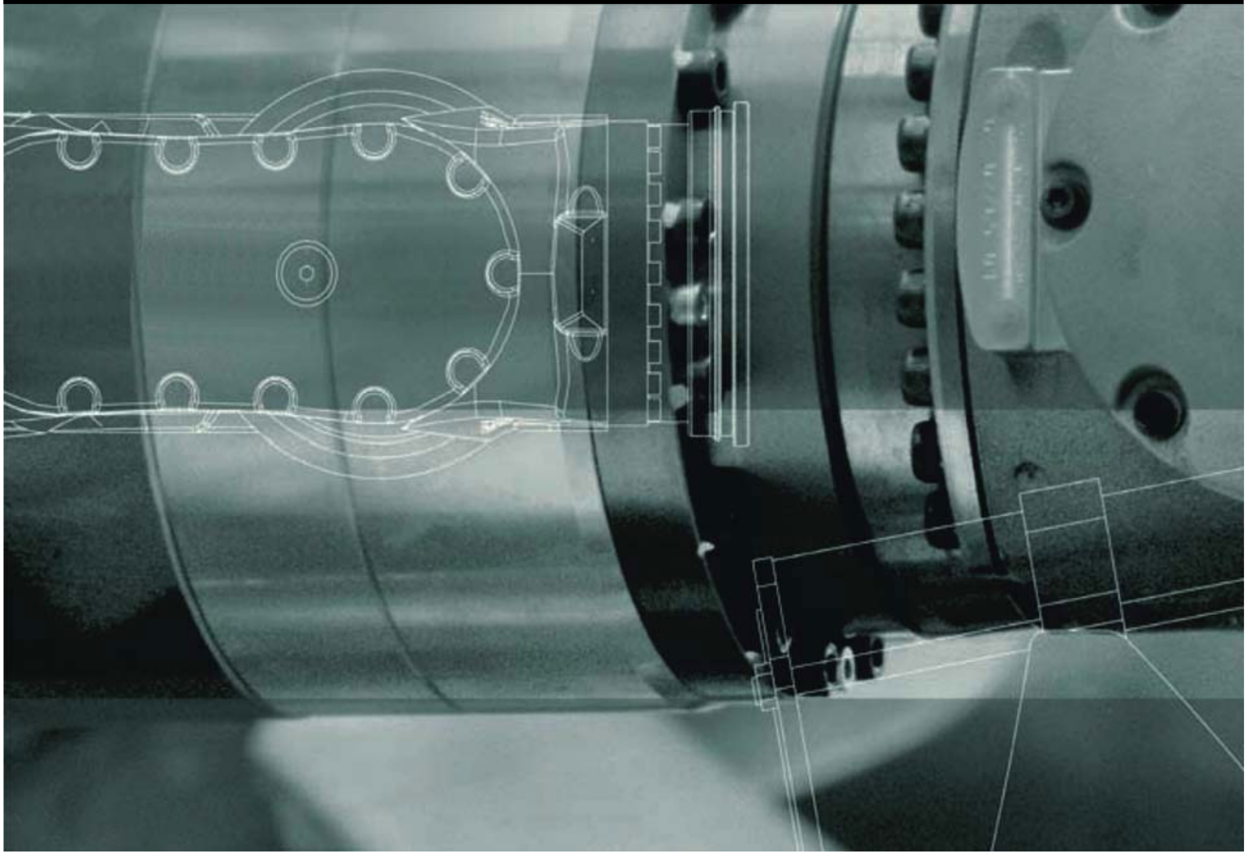
KUKA

KUKA System Software

KUKA Laboratories GmbH

KUKA System Software 5.6 lr

Operating and Programming Instructions for System Integrators



Issued: 03.01.2012

Version: KSS 5.6 lr SI V5 en

KUKA Laboratories GmbH
Zugspitzstraße 140
D-86165 Augsburg
Germany

Other functions not described in this documentation may be operable in the controller. The user has no claims to these functions, however, in the case of a replacement or service work.

We have checked the content of this documentation for conformity with the hardware and software described. Nevertheless, discrepancies cannot be precluded, for which reason we are not able to guarantee total conformity. The information in this documentation is checked on a regular basis, however, and necessary corrections will be incorporated in the subsequent edition.

Subject to technical alterations without an effect on the function.

Translation of the original documentation

KIM-PS5-DOC

Publication: Pub KSS 5.6 lr SI (PDF) en
Bookstructure: KSS 5.6 lr SI V3.1
Version: KSS 5.6 lr SI V5 en

Contents

1 Introduction

1.1 Target group

This documentation is aimed at users with the following knowledge and skills:

- Advanced knowledge of the robot controller system
- Advanced KRL programming skills

> ℹ For optimal use of our products, we recommend that our customers take part in a course of training at KUKA College. Information about the training program can be found at www.kuka.com or can be obtained directly from our subsidiaries.

1.2 Industrial robot documentation

The industrial robot documentation consists of the following parts:

- Documentation for the manipulator
- Documentation for the robot controller
- Operating and programming instructions for the KUKA System Software
- Documentation relating to options and accessories
- Parts catalog on storage medium

Each of these sets of instructions is a separate document.

1.3 Representation of warnings and notes

Safety

These warnings are relevant to safety and **must** be observed.

> ⚠ DANGER These warnings mean that it is certain or highly probable that death or severe physical injury **will** occur, if no precautions are taken.

> ⚠ WARNING These warnings mean that death or severe physical injury **may** occur, if no precautions are taken.

> ⚠ CAUTION These warnings mean that minor physical injuries **may** occur, if no precautions are taken.

> NOTICE These warnings mean that damage to property **may** occur, if no precautions are taken.

> ⚠ These warnings contain references to safety-relevant information or general safety measures. These warnings do not refer to individual hazards or individual precautionary measures.

Notes

These hints serve to make your work easier or contain references to further information.

> ℹ Tip to make your work easier or reference to further information.

1.4 Trademarks

Windows is a trademark of Microsoft Corporation.

WordPad is a trademark of Microsoft Corporation.

2 Product description

2.1 Description of the industrial robot

The industrial robot consists of the following components:

- Manipulator
- Robot controller
- KCP teach pendant
- Connecting cables
- Software
- Options, accessories

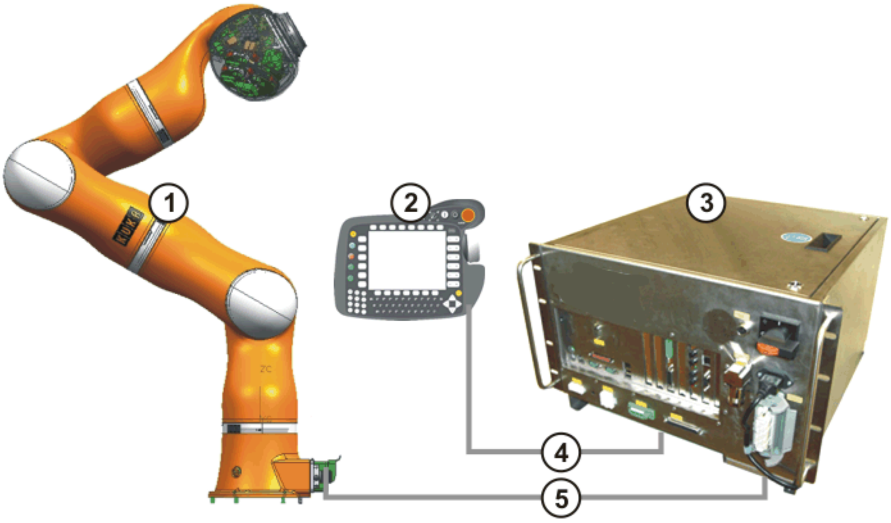


Fig. 2-1: Example of an industrial robot

1	Manipulator	4	Connecting cable/KCP
2	Teach pendant	5	Connecting cable/robot
3	Robot controller		

2.2 Overview of the software components

Overview

The following software components are used:

- KUKA System Software 5.6 lr
- Windows XP embedded 2.0 incl. Service Pack 2
  or Windows XP embedded 2.2 incl. Service Pack 3
- VxWin RT V5.0

2.3 Overview of KUKA System Software (KSS)

Description

The KUKA System Software (KSS) is responsible for all the basic operator control functions of the industrial robot.

- Path planning
- I/O management
- Data and file management
- etc.

Additional technology packages, containing application-specific instructions and configurations, can be installed.

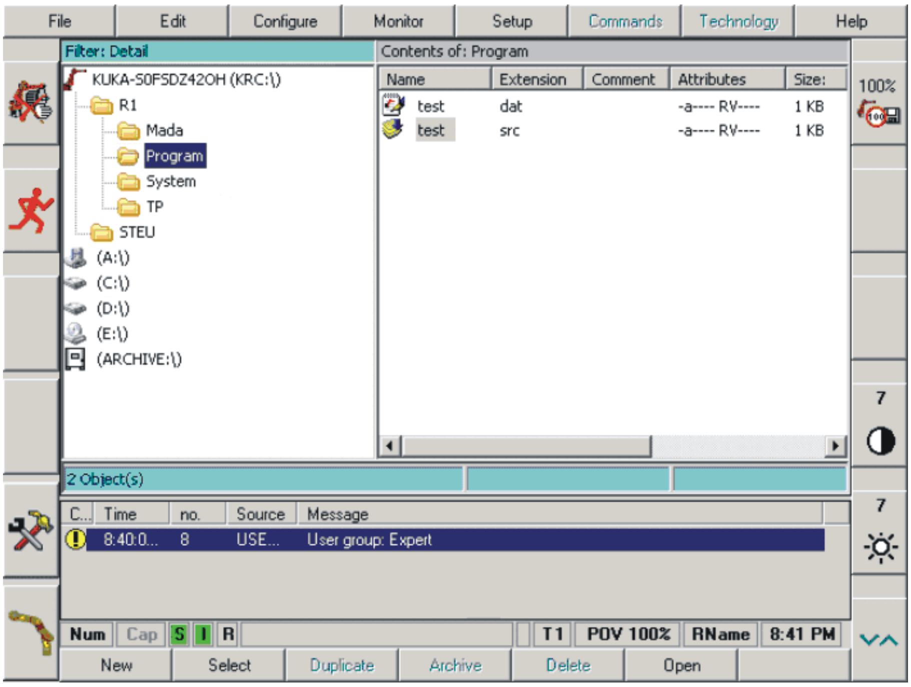
KUKA.HMI

The user interface of the KUKA System Software is called KUKA.HMI (KUKA Human-Machine Interface).

Features:

- User management
- Program editor
- KRL (KUKA Robot Language)
- Inline forms for programming
- Message display
- Configuration window
- Online help
- etc.

> Depending on customer-specific settings, the user interface may vary from the standard interface.



Contents of: Program

Name	Extension	Comment	Attributes	Size:
test	dat		-a---- RV----	1 KB
test	src		-a---- RV----	1 KB

2 Object(s)

C...	Time	no.	Source	Message
	8:40:0...	8	USE...	User group: Expert

Num Cap S I R T1 POV 100% RName 8:41 PM

Fig. 2-2: KUKA.HMI user interface

3 Safety

3.1 General

3.1.1 Liability

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

Components of the industrial robot:

- Manipulator
- Robot controller
- Teach pendant
- Connecting cables
- External axes (optional)

  e.g. linear unit, turn-tilt table, positioner
- Software
- Options, accessories

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

The industrial robot may only be used in perfect technical condition in accordance with its intended use and only by safety-conscious persons who are fully aware of the risks involved in its operation. Use of the industrial robot is subject to compliance with this document and with the declaration of incorporation supplied together with the industrial robot. Any functional disorders affecting the safety of the industrial robot must be rectified immediately.

Safety information

Safety information cannot be held against KUKA Roboter GmbH. Even if all safety instructions are followed, this is not a guarantee that the industrial robot will not cause personal injuries or material damage.

No modifications may be carried out to the industrial robot without the authorization of KUKA Roboter GmbH. Additional components (tools, software, etc.), not supplied by KUKA Roboter GmbH, may be integrated into the industrial robot. The user is liable for any damage these components may cause to the industrial robot or to other material property.

In addition to the Safety chapter, this document contains further safety instructions. These must also be observed.

3.1.2 Intended use of the industrial robot

The industrial robot is intended exclusively for the use designated in the “Purpose” chapter of the operating instructions or assembly instructions.

Further information is contained in the “Purpose” chapter of the operating instructions or assembly instructions of the industrial robot.

Using the industrial robot for any other or additional purpose is considered impermissible misuse. The manufacturer cannot be held liable for any damage resulting from such use. The risk lies entirely with the user.

Operating the industrial robot and its options within the limits of its intended use also involves observance of the operating and assembly instructions for the individual components, with particular reference to the maintenance specifications.

Misuse

Any use or application deviating from the intended use is deemed to be impermissible misuse. This includes e.g.:

- Transportation of persons and animals
- Use as a climbing aid
- Operation outside the permissible operating parameters
- Use in potentially explosive environments
- Operation without additional safeguards
- Outdoor operation

3.1.3 EC declaration of conformity and declaration of incorporation

This industrial robot constitutes partly completed machinery as defined by the EC Machinery Directive. The industrial robot may only be put into operation if the following preconditions are met:

- The industrial robot is integrated into a complete system.

  Or: The industrial robot, together with other machinery, constitutes a complete system.

  Or: All safety functions and safeguards required for operation in the complete machine as defined by the EC Machinery Directive have been added to the industrial robot.
- The complete system complies with the EC Machinery Directive. This has been confirmed by means of an assessment of conformity.

Declaration of conformity

The system integrator must issue a declaration of conformity for the complete system in accordance with the Machinery Directive. The declaration of conformity forms the basis for the CE mark for the system. The industrial robot must be operated in accordance with the applicable national laws, regulations and standards.

The robot controller is CE certified under the EMC Directive and the Low Voltage Directive.

Declaration of incorporation

The industrial robot as partly completed machinery is supplied with a declaration of incorporation in accordance with Annex II B of the EC Machinery Directive 2006/42/EC. The assembly instructions and a list of essential requirements complied with in accordance with Annex I are integral parts of this declaration of incorporation.

The declaration of incorporation declares that the start-up of the partly completed machinery remains impermissible until the partly completed machinery has been incorporated into machinery, or has been assembled with other parts to form machinery, and this machinery complies with the terms of the EC Machinery Directive, and the EC declaration of conformity is present in accordance with Annex II A.

The declaration of incorporation, together with its annexes, remains with the system integrator as an integral part of the technical documentation of the complete machinery.

3.1.4 Terms used

Term	Description
Axis range	Range of each axis, in degrees or millimeters, within which it may move. The axis range must be defined for each axis.
Stopping distance	Stopping distance = reaction distance + braking distance The stopping distance is part of the danger zone.
Workspace	The manipulator is allowed to move within its workspace. The workspace is derived from the individual axis ranges.
Operator (User)	The user of the industrial robot can be the management, employer or delegated person responsible for use of the industrial robot.
Danger zone	The danger zone consists of the workspace and the stopping distances.
Service life	The service life of a safety-relevant component begins at the time of delivery of the component to the customer. The service life is not affected by whether the component is used in a robot controller or elsewhere or not, as safety-relevant components are also subject to ageing during storage.
KCP	The KCP (KUKA Control Panel) teach pendant has all the operator control and display functions required for operating and programming the industrial robot.
Manipulator	The robot arm and the associated electrical installations
Safety zone	The safety zone is situated outside the danger zone.
Stop category 0	The drives are deactivated immediately and the brakes are applied. The manipulator and any external axes (optional) perform path-oriented braking. **Note:** This stop category is called STOP 0 in this document.
Stop category 1	The manipulator and any external axes (optional) perform path-maintaining braking. The drives are deactivated after 1 s and the brakes are applied. **Note:** This stop category is called STOP 1 in this document.
Stop category 2	The drives are not deactivated and the brakes are not applied. The manipulator and any external axes (optional) are braked with a normal braking ramp. **Note:** This stop category is called STOP 2 in this document.
System integrator (plant integrator)	System integrators are people who safely integrate the industrial robot into a complete system and commission it.
T1	Test mode, Manual Reduced Velocity (<= 250 mm/s)
T2	Test mode, Manual High Velocity (> 250 mm/s permissible)
External axis	Motion axis which is not part of the manipulator but which is controlled using the robot controller, e.g. KUKA linear unit, turn-tilt table, Posiflex.

3.2 Personnel

The following persons or groups of persons are defined for the industrial robot:

- User
- Personnel

⚠ All persons working with the industrial robot must have read and understood the industrial robot documentation, including the safety chapter.

User

The user must observe the labor laws and regulations. This includes e.g.:

- The user must comply with his monitoring obligations.
- The user must carry out instructions at defined intervals.

Personnel

Personnel must be instructed, before any work is commenced, in the type of work involved and what exactly it entails as well as any hazards which may exist. Instruction must be carried out regularly. Instruction is also required after particular incidents or technical modifications.

Personnel includes:

- System integrator
- Operators, subdivided into:
  - Start-up, maintenance and service personnel
  - Operating personnel
  - Cleaning personnel

⚠ Installation, exchange, adjustment, operation, maintenance and repair must be performed only as specified in the operating or assembly instructions for the relevant component of the industrial robot and only by personnel specially trained for this purpose.

System integrator

The industrial robot is safely integrated into a complete system by the system integrator.

The system integrator is responsible for the following tasks:

- Installing the industrial robot
- Connecting the industrial robot
- Performing risk assessment
- Implementing the required safety functions and safeguards
- Issuing the declaration of conformity
- Attaching the CE mark
- Creating the operating instructions for the complete system

Operator

The operator must meet the following preconditions:

- The operator must be trained for the work to be carried out.
- Work on the industrial robot must only be carried out by qualified personnel. These are people who, due to their specialist training, knowledge and experience, and their familiarization with the relevant standards, are able to assess the work to be carried out and detect any potential hazards.

Example

The tasks can be distributed as shown in the following table.

Tasks	Operator	Programmer	System integrator
Switch robot controller on/off	x	x	x
Start program	x	x	x
Select program	x	x	x
Select operating mode	x	x	x
Calibration (tool, base)		x	x
Master the manipulator		x	x
Configuration		x	x
Programming		x	x
Start-up			x

Tasks	Operator	Programmer	System integrator
Maintenance			x
Repair			x
Decommissioning			x
Transportation			x

⚠ Work on the electrical and mechanical equipment of the industrial robot may only be carried out by specially trained personnel.

3.3 Workspace, safety zone and danger zone

Workspaces are to be restricted to the necessary minimum size. A workspace must be safeguarded using appropriate safeguards.

The safeguards (e.g. safety gate) must be situated inside the safety zone. In the case of a stop, the manipulator and external axes (optional) are braked and come to a stop within the danger zone.

The danger zone consists of the workspace and the stopping distances of the manipulator and external axes (optional). It must be safeguarded by means of physical safeguards to prevent danger to persons or the risk of material damage.

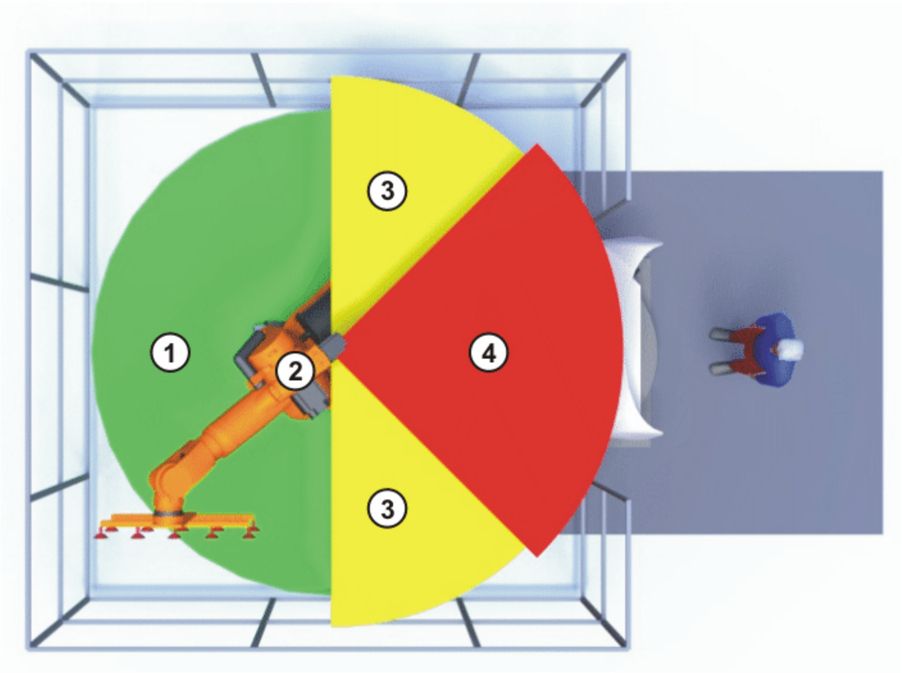


Fig. 3-1: Example of axis range A1

1	Workspace	3	Stopping distance
2	Manipulator	4	Safety zone

3.4 Triggers for stop reactions

Triggers for stop reactions

Stop reactions of the industrial robot are triggered in response to operator actions or as a reaction to monitoring functions and error messages. The following table shows the different stop reactions according to the operating mode that has been set.

STOP 0, STOP 1 and STOP 2 are the stop definitions according to DIN EN 60204-1:2006.

Trigger	T1, T2	AUT, AUT EXT
Safety gate opened	-	STOP 1
EMERGENCY STOP pressed	STOP 0	STOP 1
Enabling withdrawn	STOP 0	-
Start key released	STOP 2	-
"Drives OFF" key pressed	STOP 0	
STOP key pressed	STOP 2	
Operating mode changed	STOP 0	
Encoder error (DSE-RDC connection broken)	STOP 0	
Motion enable canceled	STOP 2	
Robot controller switched off Power failure	STOP 0	

3.5 Safety functions

3.5.1 Overview of safety functions

Safety functions:

- Mode selection
- Operator safety (= connection for the guard interlock)
- Local EMERGENCY STOP device (= EMERGENCY STOP button on the KCP)
- External EMERGENCY STOP device
- Enabling device

These circuits conform to the requirements of Performance Level d and category 3 according to EN ISO 13849-1. This only applies under the following conditions, however:

- The EMERGENCY STOP is not triggered more than once a day on average.
- The operating mode is not changed more than 10 times a day on average.
- Number of switching cycles of the main contactors:
  - Max. 100 per day
  - At least once every 6 months

> ⚠ If these conditions are not met, KUKA Roboter GmbH must be contacted.

> ⚠ **DANGER** In the absence of operational safety functions and safeguards, the industrial robot can cause personal injury or material damage. If safety functions or safeguards are dismantled or deactivated, the industrial robot may not be operated.

3.5.2 ESC safety logic

The function and triggering of the electronic safety functions are monitored by the ESC safety logic.

The ESC (Electronic Safety Circuit) safety logic is a dual-channel computer-aided safety system. It permanently monitors all connected safety-relevant components. In the event of a fault or interruption in the safety circuit, the power supply to the drives is shut off, thus bringing the industrial robot to a standstill.

The ESC safety logic triggers different stop reactions, depending on the operating mode of the industrial robot.

The ESC safety logic monitors the following inputs:

- Operator safety
- Local EMERGENCY STOP (= EMERGENCY STOP button on the KCP)
- External EMERGENCY STOP
- Enabling device
- Drives OFF
- Drives ON
- Operating modes
- Qualifying inputs

The ESC safety logic monitors the following outputs:

- Operating mode
- Local E-STOP

3.5.3 Mode selector switch

The industrial robot can be operated in the following modes:

- Manual Reduced Velocity (T1)
- Manual High Velocity (T2)
- Automatic (AUT)
- Automatic External (AUT EXT)

The operating mode is selected using the mode selector switch on the KCP. The switch is activated by means of a key which can be removed. If the key is removed, the switch is locked and the operating mode can no longer be changed.

If the operating mode is changed during operation, the drives are immediately switched off. The manipulator and any external axes (optional) are stopped with a STOP 0.

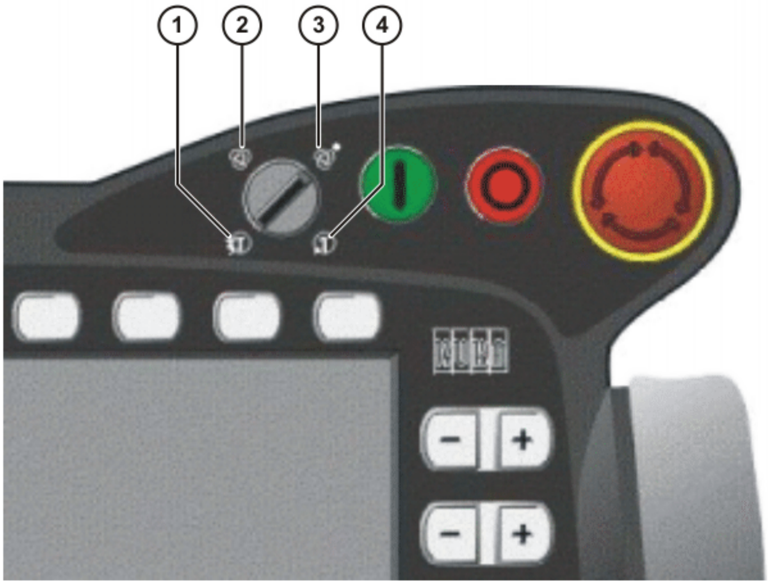


Fig. 3-2: Mode selector switch

1	T2 (Manual High Velocity)
2	AUT (Automatic)
3	AUT EXT (Automatic External)
4	T1 (Manual Reduced Velocity)

Operating mode	Use	Velocities
T1	For test operation, programming and teaching	■ Program verification: Programmed velocity, maximum 250 mm/s ■ Jog mode: Jog velocity, maximum 250 mm/s
T2	For test operation	■ Program verification: Programmed velocity
AUT	For industrial robots without higher-level controllers Only possible with a connected safety circuit	■ Program mode: Programmed velocity ■ Jog mode: Not possible
AUT EXT	For industrial robots with higher-level controllers, e.g. PLC Only possible with a connected safety circuit	■ Program mode: Programmed velocity ■ Jog mode: Not possible

3.5.4 Operator safety

The operator safety input is used for interlocking physical safeguards. Safety equipment, such as safety gates, can be connected to the dual-channel input. If nothing is connected to this input, operation in Automatic mode is not possible. Operator safety is not active in the test modes T1 (Manual Reduced Velocity) and T2 (Manual High Velocity).

In the event of a loss of signal during Automatic operation (e.g. safety gate is opened), the manipulator and the external axes (optional) stop with a STOP 1. Once the signal is active at the input again, automatic operation can be resumed.

Operator safety can be connected via the peripheral interface on the robot controller.

> ⚠ **WARNING** It must be ensured that the operator safety signal is not automatically reset when the safeguard (e.g. safety gate) is closed, but only after an additional manual acknowledgement signal has been given. Only in this way can it be ensured that automatic operation is not resumed inadvertently while there are still persons in the danger zone, e.g. due to the safety gate closing accidentally.
> Failure to observe this precaution may result in death, severe physical injuries or considerable damage to property.

3.5.5 EMERGENCY STOP device

The EMERGENCY STOP device for the industrial robot is the EMERGENCY STOP button on the KCP. The button must be pressed in the event of a hazardous situation or emergency.

Reactions of the industrial robot if the EMERGENCY STOP button is pressed:

- Manual Reduced Velocity (T1) and Manual High Velocity (T2) modes:

  The drives are switched off immediately. The manipulator and any external axes (optional) are stopped with a STOP 0.
- Automatic modes (AUT and AUT EXT):

  The drives are switched off after 1 second. The manipulator and any external axes (optional) are stopped with a STOP 1.

Before operation can be resumed, the EMERGENCY STOP button must be turned to release it and the stop message must be acknowledged.

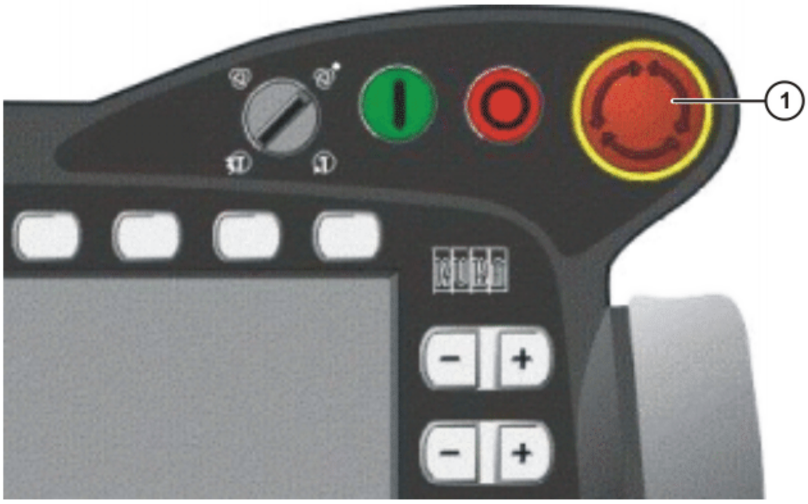


Fig. 3-3: EMERGENCY STOP button on the KCP

1 EMERGENCY STOP button

⚠ WARNING Tools and other equipment connected to the manipulator must be integrated into the EMERGENCY STOP circuit on the system side if they could constitute a potential hazard.
Failure to observe this precaution may result in death, severe physical injuries or considerable damage to property.

3.5.6 External EMERGENCY STOP device

There must be EMERGENCY STOP devices available at every operator station that can initiate a robot motion or other potentially hazardous situation. The system integrator is responsible for ensuring this.

There must always be at least one external EMERGENCY STOP device installed. This ensures that an EMERGENCY STOP device is available even when the KCP is disconnected.

External EMERGENCY STOP devices are connected via the customer interface. External EMERGENCY STOP devices are not included in the scope of supply of the industrial robot.

3.5.7 Enabling device

The enabling devices of the industrial robot are the enabling switches on the KCP.

There are 3 enabling switches installed on the KCP. The enabling switches have 3 positions:

- Not pressed
- Center position
- Panic position

In the test modes, the manipulator can only be moved if one of the enabling switches is held in the central position. If the enabling switch is released or pressed fully down (panic position), the drives are deactivated immediately and the manipulator stops with a STOP 0.

⚠ WARNING The enabling switches must not be held down by adhesive tape or other means or manipulated in any other way.
Death, serious physical injuries or major damage to property may result.

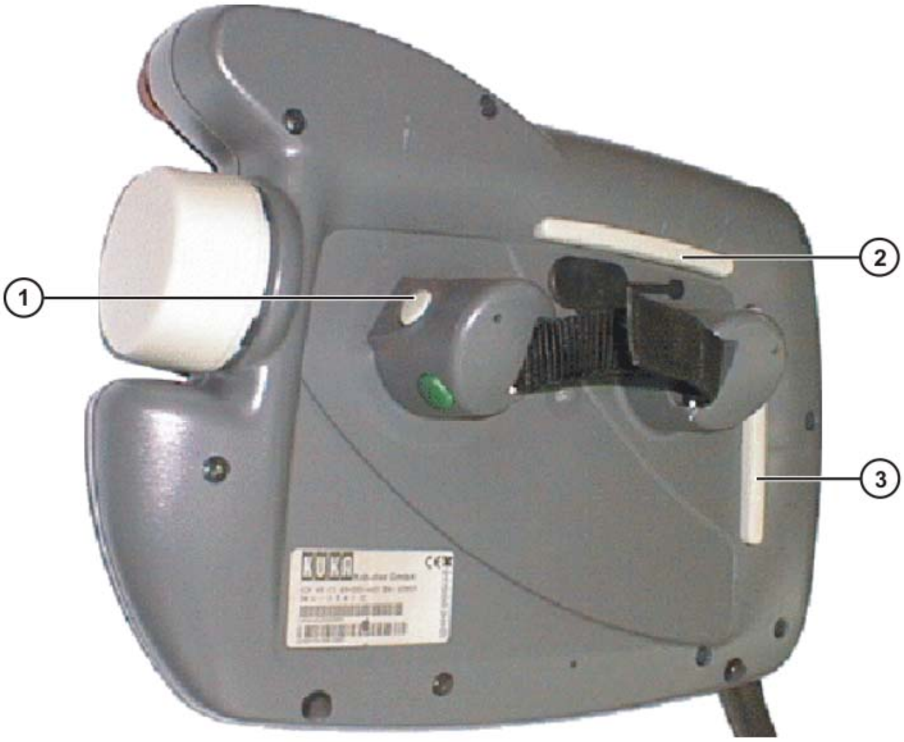


Fig. 3-4: Enabling switches on the KCP

1 - 3	Enabling switches

3.6 Additional protective equipment

3.6.1 Jog mode

In the operating modes T1 (Manual Reduced Velocity) and T2 (Manual High Velocity), the robot controller can only execute programs in jog mode. This means that it is necessary to hold down an enabling switch and the Start key in order to execute a program.

If the enabling switch is released or pressed fully down (panic position), the drives are deactivated immediately and the manipulator and any external axes (optional) stop with a STOP 0.

Releasing only the Start key causes the industrial robot to be stopped with a STOP 2.

3.6.2 Software limit switches

The axis ranges of all manipulator and positioner axes are limited by means of adjustable software limit switches. These software limit switches only serve as machine protection and must be adjusted in such a way that the manipulator/ positioner cannot hit the mechanical end stops.

The software limit switches are set during commissioning of an industrial robot.

Further information is contained in the operating and programming instructions.

3.6.3 Labeling on the industrial robot

All plates, labels, symbols and marks constitute safety-relevant parts of the industrial robot. They must not be modified or removed.

Labeling on the industrial robot consists of:

- Identification plates
- Warning labels
- Safety symbols
- Designation labels
- Cable markings
- Rating plates

> Further information is contained in the technical data of the operating instructions or assembly instructions of the components of the industrial robot.

3.6.4 External safeguards

Safeguards

The access of persons to the danger zone of the manipulator must be prevented by means of safeguards.

Physical safeguards must meet the following requirements:

- They meet the requirements of EN 953.
- They prevent access of persons to the danger zone and cannot be easily circumvented.
- They are sufficiently fastened and can withstand all forces that are likely to occur in the course of operation, whether from inside or outside the enclosure.
- They do not, themselves, represent a hazard or potential hazard.
- The prescribed minimum clearance from the danger zone is maintained.

Safety gates (maintenance gates) must meet the following requirements:

- They are reduced to an absolute minimum.
- The interlocks (e.g. safety gate switches) are linked to the operator safety input of the robot controller via safety gate switching devices or safety PLC.
- Switching devices, switches and the type of switching conform to the requirements of Performance Level d and category 3 according to EN ISO 13849-1.
- Depending on the risk situation: the safety gate is additionally safeguarded by means of a locking mechanism that only allows the gate to be opened if the manipulator is safely at a standstill.
- The button for acknowledging the safety gate is located outside the space limited by the safeguards.

> Further information is contained in the corresponding standards and regulations. These also include EN 953.

Other safety equipment

Other safety equipment must be integrated into the system in accordance with the corresponding standards and regulations.

3.7 Overview of operating modes and safety functions

The following table indicates the operating modes in which the safety functions are active.

Safety functions	T1	T2	AUT	AUT EXT
Operator safety	-	-	active	active
EMERGENCY STOP device	active	active	active	active
Enabling device	active	active	-	-
Reduced velocity during program verification	active	-	-	-
Jog mode	active	active	-	-
Software limit switches	active	active	active	active

3.8 Safety measures

3.8.1 General safety measures

The industrial robot may only be used in perfect technical condition in accordance with its intended use and only by safety-conscious persons. Operator errors can result in personal injury and damage to property.

It is important to be prepared for possible movements of the industrial robot even after the robot controller has been switched off and locked. Incorrect installation (e.g. overload) or mechanical defects (e.g. brake defect) can cause the manipulator or external axes to sag. If work is to be carried out on a switched-off industrial robot, the manipulator and external axes must first be moved into a position in which they are unable to move on their own, whether the payload is mounted or not. If this is not possible, the manipulator and external axes must be secured by appropriate means.

⚠ DANGER In the absence of operational safety functions and safeguards, the industrial robot can cause personal injury or material damage. If safety functions or safeguards are dismantled or deactivated, the industrial robot may not be operated.

⚠ WARNING Standing underneath the robot arm can cause death or serious physical injuries. For this reason, standing underneath the robot arm is prohibited!

⚠ CAUTION The motors reach temperatures during operation which can cause burns to the skin. Contact must be avoided. Appropriate safety precautions must be taken, e.g. protective gloves must be worn.

KCP

The user must ensure that the industrial robot is only operated with the KCP by authorized persons.

If more than one KCP is used in the overall system, it must be ensured that each KCP is unambiguously assigned to the corresponding industrial robot. They must not be interchanged.

⚠ WARNING The operator must ensure that decoupled KCPs are immediately removed from the system and stored out of sight and reach of personnel working on the industrial robot. This serves to prevent operational and non-operational EMERGENCY STOP facilities from becoming interchanged.
Failure to observe this precaution may result in death, severe physical injuries or considerable damage to property.

External keyboard, external mouse

An external keyboard and/or external mouse may only be used if the following conditions are met:

- Start-up or maintenance work is being carried out.
- The drives are switched off.
- There are no persons in the danger zone.

The KCP must not be used as long as an external keyboard and/or external mouse are connected.

The external keyboard and/or external mouse must be removed as soon as the start-up or maintenance work is completed or the KCP is connected.

Faults

The following tasks must be carried out in the case of faults in the industrial robot:

- Switch off the robot controller and secure it (e.g. with a padlock) to prevent unauthorized persons from switching it on again.
- Indicate the fault by means of a label with a corresponding warning (tag-out).
- Keep a record of the faults.
- Eliminate the fault and carry out a function test.

Modifications

After modifications to the industrial robot, checks must be carried out to ensure the required safety level. The valid national or regional work safety regulations must be observed for this check. The correct functioning of all safety circuits must also be tested.

New or modified programs must always be tested first in Manual Reduced Velocity mode (T1).

After modifications to the industrial robot, existing programs must always be tested first in Manual Reduced Velocity mode (T1). This applies to all components of the industrial robot and includes modifications to the software and configuration settings.

3.8.2 Testing safety-related controller components

All safety-related controller components are rated for a service life of 20 years (with the exception of the input/output terminals for safe bus systems). The controller components must nonetheless be tested regularly to ensure that they are still functional.

Check:

The E-STOP pushbutton and the mode selector switch must be actuated at least once every 6 months in order to detect any malfunction.

Additional checks are required during start-up and recommissioning.

> **⚠ WARNING** If input/output terminals are used in the robot controller for safe bus systems, these must be exchanged after 10 years at the latest. If this is not done, the integrity of the safety functions is not assured. This can result in death, physical injuries and damage to property.

3.8.3 Transportation

Manipulator

The prescribed transport position of the manipulator must be observed. Transportation must be carried out in accordance with the operating instructions or assembly instructions of the manipulator.

Robot controller

The robot controller must be transported and installed in an upright position. Avoid vibrations and impacts during transportation in order to prevent damage to the robot controller.

Transportation must be carried out in accordance with the operating instructions or assembly instructions of the robot controller.

External axis (optional)

The prescribed transport position of the external axis (e.g. KUKA linear unit, turn-tilt table, etc.) must be observed. Transportation must be carried out in accordance with the operating instructions or assembly instructions of the external axis.

3.8.4 Start-up and recommissioning

Before starting up systems and devices for the first time, a check must be carried out to ensure that the systems and devices are complete and operational, that they can be operated safely and that any damage is detected.

The valid national or regional work safety regulations must be observed for this check. The correct functioning of all safety circuits must also be tested.

> ⚠ The passwords for logging onto the KUKA System Software as “Expert” and “Administrator” must be changed before start-up and must only be communicated to authorized personnel.

> **⚠ DANGER** The robot controller is preconfigured for the specific industrial robot. If cables are interchanged, the manipulator and the external axes (optional) may receive incorrect data and can thus cause personal injury or material damage. If a system consists of more than one manipulator, always connect the connecting cables to the manipulators and their corresponding robot controllers.

> ⚠ If additional components (e.g. cables), which are not part of the scope of supply of KUKA Roboter GmbH, are integrated into the industrial robot, the user is responsible for ensuring that these components do not adversely affect or disable safety functions.

> **NOTICE** If the internal cabinet temperature of the robot controller differs greatly from the ambient temperature, condensation can form, which may cause damage to the electrical components. Do not put the robot controller into operation until the internal temperature of the cabinet has adjusted to the ambient temperature.

Function test

The following tests must be carried out before start-up and recommissioning:

General test:

It must be ensured that:

- The industrial robot is correctly installed and fastened in accordance with the specifications in the documentation.
- There are no foreign bodies or loose parts on the industrial robot.
- All required safety equipment is correctly installed and operational.
- The power supply ratings of the industrial robot correspond to the local supply voltage and mains type.
- The ground conductor and the equipotential bonding cable are sufficiently rated and correctly connected.
- The connecting cables are correctly connected and the connectors are locked.

Test of safety-oriented circuits:

A function test must be carried out for the following safety-oriented circuits to ensure that they are functioning correctly:

- Local EMERGENCY STOP device (= EMERGENCY STOP button on the KCP)
- External EMERGENCY STOP device (input and output)
- Enabling device (in the test modes)
- Operator safety (in the automatic modes)
- Qualifying inputs (if connected)
- All other safety-relevant inputs and outputs used

Test of reduced velocity control:

This test is to be carried out as follows:

1. Program a straight path with the maximum possible velocity.
2. Calculate the length of the path.
3. Execute the path in T1 mode with the override set to 100% and time the motion with a stopwatch.

> ⚠ **WARNING** It must be ensured that no persons are present within the danger zone during path execution. Death or severe physical injuries may result.

4. Calculate the velocity from the length of the path and the time measured for execution of the motion.

Control of reduced velocity is functioning correctly if the following results are achieved:

- The calculated velocity does not exceed 250 mm/s.
- The robot executes the path as programmed (i.e. in a straight line, without deviations).

Machine data

It must be ensured that the rating plate on the robot controller has the same machine data as those entered in the declaration of incorporation. The machine data on the rating plate of the manipulator and the external axes (optional) must be entered during start-up.

> ⚠ **DANGER** The industrial robot must not be moved if incorrect machine data are loaded. Death, severe physical injuries or considerable damage to property may otherwise result. The correct machine data must be loaded.

3.8.5 Virus protection and network security

The user of the industrial robot is responsible for ensuring that the software is always safeguarded with the latest virus protection. If the robot controller is integrated into a network that is connected to the company network or to the Internet, it is advisable to protect this robot network against external risks by means of a firewall.

> For optimal use of our products, we recommend that our customers carry out a regular virus scan. Information about security updates can be found at www.kuka.com.

3.8.6 Manual mode

Manual mode is the mode for setup work. Setup work is all the tasks that have to be carried out on the industrial robot to enable automatic operation. Setup work includes:

- Jog mode
- Teaching
- Programming
- Program verification

The following must be taken into consideration in manual mode:

- If the drives are not required, they must be switched off to prevent the manipulator or the external axes (optional) from being moved unintentionally.

  New or modified programs must always be tested first in Manual Reduced Velocity mode (T1).
- The manipulator, tooling or external axes (optional) must never touch or project beyond the safety fence.
- Workpieces, tooling and other objects must not become jammed as a result of the industrial robot motion, nor must they lead to short-circuits or be liable to fall off.
- All setup work must be carried out, where possible, from outside the safeguarded area.

If the setup work has to be carried out inside the safeguarded area, the following must be taken into consideration:

In **Manual Reduced Velocity mode (T1)**:

- If it can be avoided, there must be no other persons inside the safeguarded area.

  If it is necessary for there to be several persons inside the safeguarded area, the following must be observed:
  - Each person must have an enabling device.
  - All persons must have an unimpeded view of the industrial robot.
  - Eye-contact between all persons must be possible at all times.
- The operator must be so positioned that he can see into the danger area and get out of harm’s way.

In **Manual High Velocity mode (T2)**:

- This mode may only be used if the application requires a test at a velocity higher than Manual Reduced Velocity.
- Teaching and programming are not permissible in this operating mode.
- Before commencing the test, the operator must ensure that the enabling devices are operational.
- The operator must be positioned outside the danger zone.

- There must be no other persons inside the safeguarded area. It is the responsibility of the operator to ensure this.

3.8.7 Simulation

Simulation programs do not correspond exactly to reality. Robot programs created in simulation programs must be tested in the system in **Manual Reduced Velocity mode (T1)**. It may be necessary to modify the program.

3.8.8 Automatic mode

Automatic mode is only permissible in compliance with the following safety measures:

- All safety equipment and safeguards are present and operational.
- There are no persons in the system.
- The defined working procedures are adhered to.

If the manipulator or an external axis (optional) comes to a standstill for no apparent reason, the danger zone must not be entered until an EMERGENCY STOP has been triggered.

3.8.9 Maintenance and repair

After maintenance and repair work, checks must be carried out to ensure the required safety level. The valid national or regional work safety regulations must be observed for this check. The correct functioning of all safety circuits must also be tested.

The purpose of maintenance and repair work is to ensure that the system is kept operational or, in the event of a fault, to return the system to an operational state. Repair work includes troubleshooting in addition to the actual repair itself.

The following safety measures must be carried out when working on the industrial robot:

- Carry out work outside the danger zone. If work inside the danger zone is necessary, the user must define additional safety measures to ensure the safe protection of personnel.
- Switch off the industrial robot and secure it (e.g. with a padlock) to prevent it from being switched on again. If it is necessary to carry out work with the robot controller switched on, the user must define additional safety measures to ensure the safe protection of personnel.
- If it is necessary to carry out work with the robot controller switched on, this may only be done in operating mode T1.
- Label the system with a sign indicating that work is in progress. This sign must remain in place, even during temporary interruptions to the work.
- The EMERGENCY STOP systems must remain active. If safety functions or safeguards are deactivated during maintenance or repair work, they must be reactivated immediately after the work is completed.

Faulty components must be replaced using new components with the same article numbers or equivalent components approved by KUKA Roboter GmbH for this purpose.

Cleaning and preventive maintenance work is to be carried out in accordance with the operating instructions.

Robot controller

Even when the robot controller is switched off, parts connected to peripheral devices may still carry voltage. The external power sources must therefore be switched off if work is to be carried out on the robot controller.

The ESD regulations must be adhered to when working on components in the robot controller.

Voltages in excess of 50 V (up to 600 V) can be present in various components for several minutes after the robot controller has been switched off! To prevent life-threatening injuries, no work may be carried out on the industrial robot in this time.

Water and dust must be prevented from entering the robot controller.

Hazardous substances

The following safety measures must be carried out when handling hazardous substances:

- Avoid prolonged and repeated intensive contact with the skin.
- Avoid breathing in oil spray or vapors.
- Clean skin and apply skin cream.

⚠ To ensure safe use of our products, we recommend that our customers regularly request up-to-date safety data sheets from the manufacturers of hazardous substances.

3.8.10 Decommissioning, storage and disposal

The industrial robot must be decommissioned, stored and disposed of in accordance with the applicable national laws, regulations and standards.

3.8.11 Safety measures for “single point of control”

Overview

If certain components in the industrial robot are operated, safety measures must be taken to ensure complete implementation of the principle of “single point of control”.

Components:

- Submit interpreter
- PLC
- OPC Server
- Remote control tools
- External keyboard/mouse

ℹ The implementation of additional safety measures may be required. This must be clarified for each specific application; this is the responsibility of the system integrator, programmer or user of the system.

Since only the system integrator knows the safe states of actuators in the periphery of the robot controller, it is his task to set these actuators to a safe state, e.g. in the event of an EMERGENCY STOP.

Submit interpreter, PLC

If motions, (e.g. drives or grippers) are controlled with the Submit interpreter or the PLC via the I/O system, and if they are not safeguarded by other means, then this control will take effect even in T1 and T2 modes or while an EMERGENCY STOP is active.

If variables that affect the robot motion (e.g. override) are modified with the Submit interpreter or the PLC, this takes effect even in T1 and T2 modes or while an EMERGENCY STOP is active.

Safety measures:

- Do not modify safety-relevant signals and variables (e.g. operating mode, EMERGENCY STOP, safety gate contact) via the Submit interpreter or PLC.
- If modifications are nonetheless required, all safety-relevant signals and variables must be linked in such a way that they cannot be set to a dangerous state by the Submit interpreter or PLC.

OPC server, remote control tools

These components can be used with write access to modify programs, outputs or other parameters of the robot controller, without this being noticed by any persons located inside the system.

Safety measures:

- KUKA stipulates that these components are to be used exclusively for diagnosis and visualization.

  Programs, outputs or other parameters of the robot controller must not be modified using these components.

External keyboard/mouse

These components can be used to modify programs, outputs or other parameters of the robot controller, without this being noticed by any persons located inside the system.

Safety measures:

- Only use one operator console at each robot controller.
- If the KCP is being used for work inside the system, remove any keyboard and mouse from the robot controller beforehand.

3.9 Applied norms and regulations

Name	Definition	Edition
2006/42/EC	Machinery Directive: Directive 2006/42/EC of the European Parliament and of the Council of 17 May 2006 on machinery, and amending Directive 95/16/EC (recast)	2006
2004/108/EC	EMC Directive: Directive 2004/108/EC of the European Parliament and of the Council of 15 December 2004 on the approximation of the laws of the Member States relating to electromagnetic compatibility and repealing Directive 89/336/EEC	2004
EN ISO 13850	Safety of machinery: Emergency stop - Principles for design	2008
EN ISO 13849-1	Safety of machinery: Safety-related parts of control systems - Part 1: General principles of design	2008
EN ISO 13849-2	Safety of machinery: Safety-related parts of control systems - Part 2: Validation	2008
EN ISO 12100-1	Safety of machinery: Basic concepts, general principles for design - Part 1: Basic terminology, methodology	2003
EN ISO 12100-2	Safety of machinery: Basic concepts, general principles for design - Part 2: Technical principles	2003

Name	Definition	Edition
EN ISO 10218-1	Industrial robots: Safety	2008
EN 614-1	Safety of machinery: Ergonomic design principles - Part 1: Terms and general principles	2006
EN 61000-6-2	Electromagnetic compatibility (EMC): Part 6-2: Generic standards; Immunity for industrial environments	2005
EN 61000-6-4	Electromagnetic compatibility (EMC): Part 6-4: Generic standards; Emission standard for industrial environments	2007
EN 60204-1	Safety of machinery: Electrical equipment of machines - Part 1: General requirements	2006

4 Operation

4.1 KCP teach pendant

4.1.1 Front view

Function

The KCP (KUKA Control Panel) is the teach pendant for the industrial robot. The KCP has all the control and display functions required for operating and programming the industrial robot.

Overview

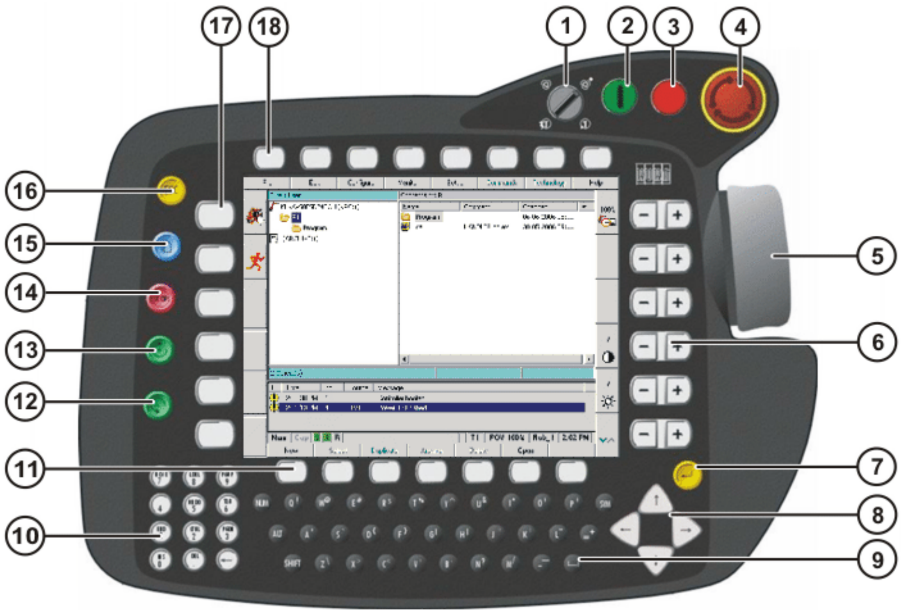


Fig. 4-1: Front view of KCP

1	Mode selector switch	10	Numeric keypad
2	Drives ON	11	Softkeys
3	Drives OFF / SSB GUI	12	Start backwards key
4	EMERGENCY STOP button	13	Start key
5	Space Mouse	14	STOP key
6	Right-hand status keys	15	Window selection key
7	Enter key	16	ESC key
8	Arrow keys	17	Left-hand status keys
9	Keypad	18	Menu keys

Description

Element	Description
Mode selector switch	(>>> 4.12 "Mode selector switch" Page 52)
Drives ON	Switches the robot drives on.
Drives OFF **SSB GUI**	Switches the robot drives off. Only with Shared Pendant (KCP for KUKA.RoboTeam): SSB GUI calls the user interface of the Safety Selection Board
E-STOP push-button	Stops the robot in hazardous situations. The EMERGENCY STOP button locks itself in place when it is pressed.
Space Mouse	Jogs the robot.

Element	Description
Right-hand status keys	(>>> 4.2.1 "Status keys, menu keys, softkeys" Page 43)
Enter key	The Enter key is used to close an active window or inline form. Changes are saved.
Arrow keys	The arrow keys are used to jump from element to element in the user interface. **Note:** If an element cannot be accessed using the arrow keys, use the TAB key instead.
Keypad	(>>> 4.1.2 "Keypad" Page 40)
Numeric keypad	(>>> 4.1.3 "Numeric keypad" Page 41)
Softkeys	(>>> 4.2.1 "Status keys, menu keys, softkeys" Page 43)
Start backwards key	The Start backwards key is used to start a program backwards. The program is executed step by step.
Start key	The Start key is used to start a program.
STOP key	The STOP key is used to stop a program that is running.
Window selection key	The window selection key is used to toggle between the main, option and message windows. The selected window is indicated by a blue background.
ESC key	The ESC key is used to abort an action on the user interface.
Left-hand status keys	(>>> 4.2.1 "Status keys, menu keys, softkeys" Page 43)
Menu keys	(>>> 4.2.1 "Status keys, menu keys, softkeys" Page 43)

4.1.2 Keypad

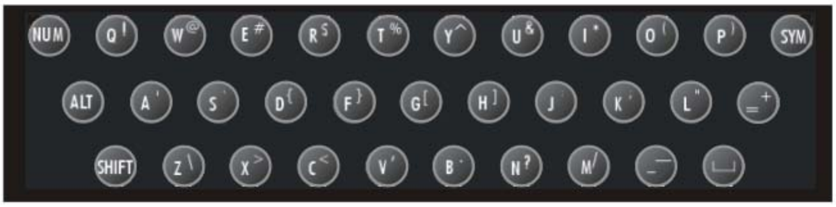


Fig. 4-2: Keypad

Key	Description
NUM	NUM is used to toggle between the numeric function and the control function of the numeric keypad. The status bar indicates which of the functions is active (>>> 4.2.4 "Status bar" Page 46).
ALT	ALT is used in keyboard shortcuts. The key remains activated for one keystroke. In other words, it does not need to be held down.

Key	Description
SHIFT	SHIFT is used to switch between upper-case and lower-case letters. The key remains activated for one keystroke. In other words, it does not need to be held down to type one upper-case letter. To type several upper-case characters, the SHIFT key must be held down. SYM+SHIFT switches to permanent upper-case typing. The status bar indicates whether upper-case or lower-case typing is active (>>> 4.2.4 "Status bar" Page 46).
SYM	SYM must be pressed to enter the secondary characters assigned to the letter keys, e.g. the "#" character on the "E" key. The key remains activated for one keystroke. In other words, it does not need to be held down.

4.1.3 Numeric keypad

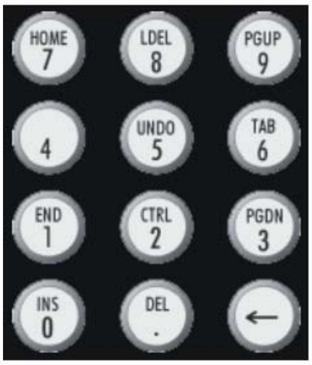


Fig. 4-3: Numeric keypad

The NUM key in the keypad is used to toggle between the numeric function and the control function of the numeric keypad. The status bar indicates which of the functions is active. (>>> 4.2.4 "Status bar" Page 46)

Key	Control function
INS (0)	Switches between insert and overwrite mode.
DEL (.)	Deletes the character to the right of the cursor.
<-	Deletes the character to the left of the cursor.
END (1)	Positions the cursor to the end of the line in which it is currently situated.
CTRL (2)	Used in keyboard shortcuts.
PG DN (3)	Scrolls one screen towards the end of the file.
(4)	----
UNDO (5)	Undoes the last input. (This function is not currently supported.)
TAB (6)	Positions the focus or the cursor on the next user interface element. **Note:** If an element cannot be accessed using the TAB key, use the arrow keys instead.
HOME (7)	Positions the cursor to the start of the line in which it is currently situated.
LDEL (8)	Deletes the line in which the cursor is positioned.
PG UP (9)	Scrolls one screen towards the start of the file.

4.1.4 Rear view

Overview

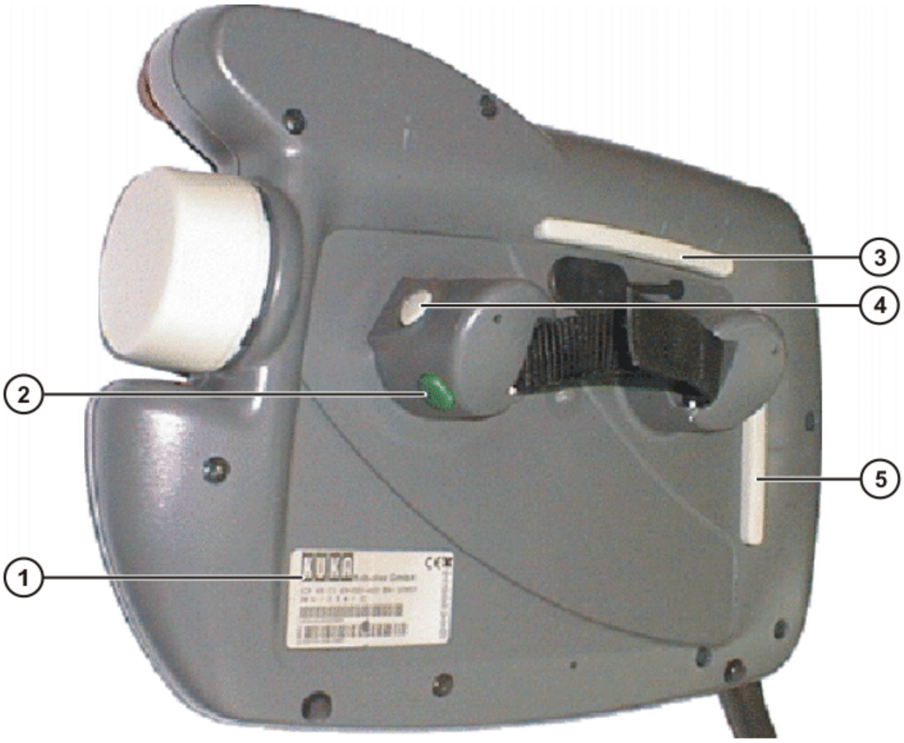


Fig. 4-4: Rear view of KCP

1	Identification plate	4	Enabling switch
2	Start key	5	Enabling switch
3	Enabling switch		

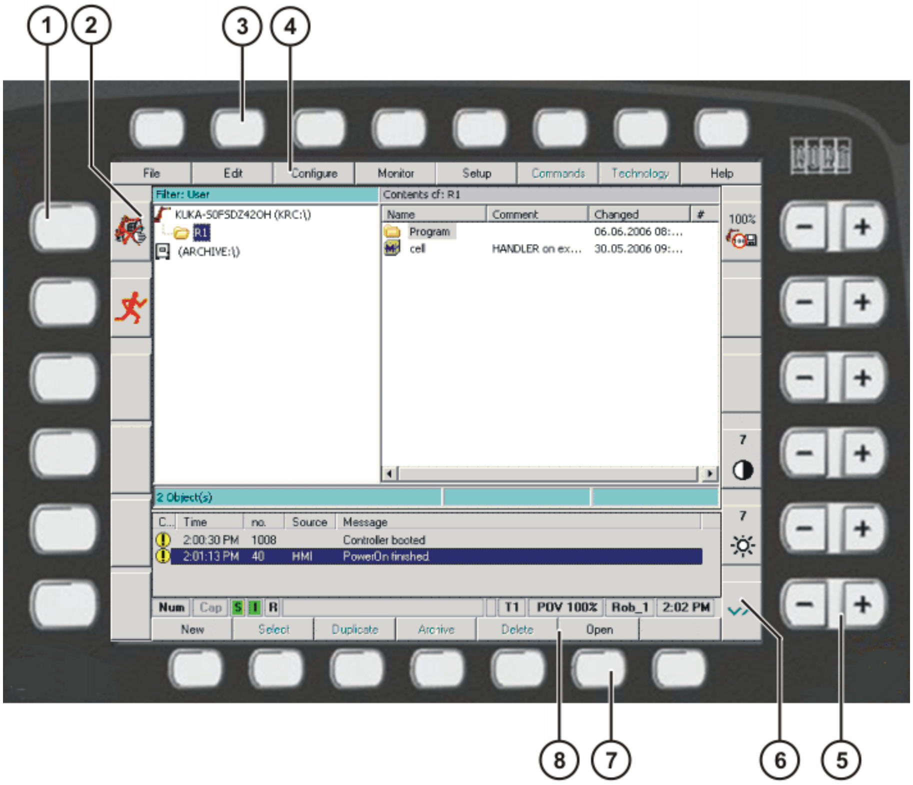
Description

Element	Description
Identification plate	KCP rating plate
Start key	The Start key is used to start a program.
Enabling switch	The enabling switch has 3 positions: ■ Not pressed ■ Center position ■ Panic position The enabling switch must be held in the **center position** in operating modes T1 and T2 in order to be able to jog the robot. In the operating modes Automatic and Automatic External, the enabling switch has no function.

4.2 KUKA.HMI user interface

4.2.1 Status keys, menu keys, softkeys

Overview



Name	Comment	Changed	#
Program		06.06.2006 08:...	
cell	HANDLER on ex...	30.05.2006 09:...	

C...	Time	no.	Source	Message
!	2:00:30 PM	1008		Controller booted
!	2:01:13 PM	40	HMI	PowerOn finished

Fig. 4-5: Status keys, menu keys and softkeys in the user interface

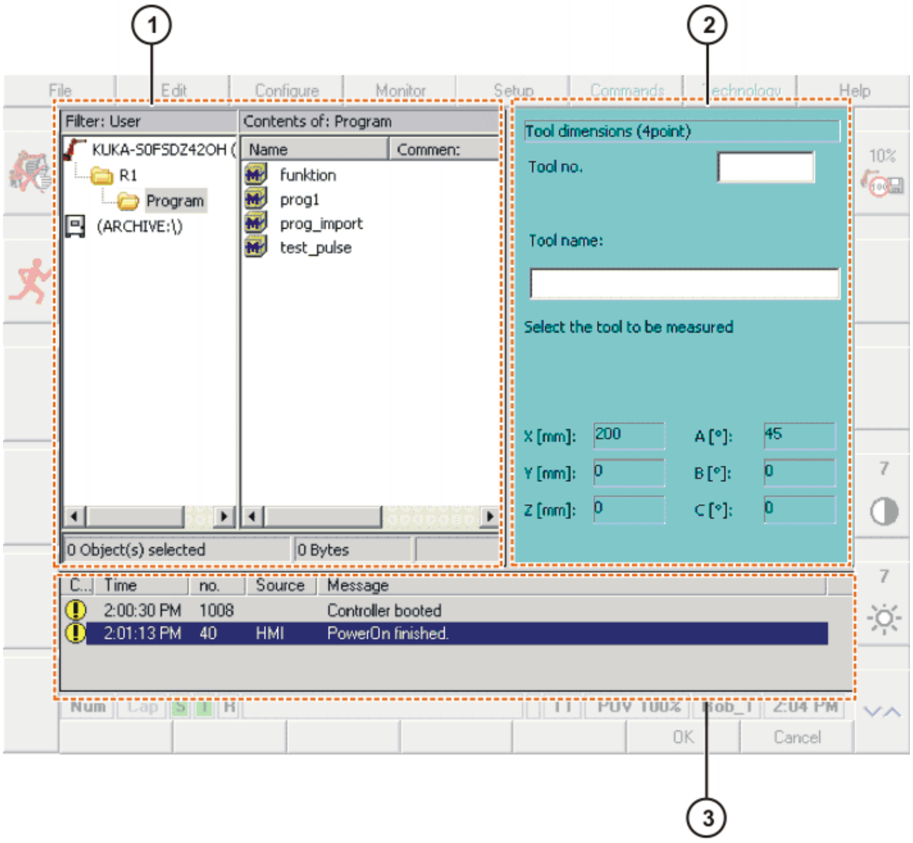
1	Left-hand status keys	5	Right-hand status keys
2	Left-hand status keys (icons)	6	Right-hand status keys (icons)
3	Menu keys	7	Softkeys
4	Menu keys (icons)	8	Softkeys (icons)

Description

Element	Description
Status keys	The status keys are used primarily for controlling the robot and setting values. Example: selecting the robot jog mode. The icons change dynamically.
Menu keys	The menu keys are used to open the menus.
Softkeys	The icons change dynamically and always refer to the active window.

4.2.2 Windows in the user interface

Overview



Filter: User

KUKA-S0FSDZ42OH (
R1
Program
(ARCHIVE:\)

Contents of: Program

Name	Commen:
funktion	
prog1	
prog_import	
test_pulse	

0 Object(s) selected | 0 Bytes

Tool dimensions (4point)

Tool no.

Tool name:

Select the tool to be measured

X [mm]: 200 A [°]: 45
Y [mm]: 0 B [°]: 0
Z [mm]: 0 C [°]: 0

C...	Time	no.	Source	Message
!	2:00:30 PM	1008		Controller booted
!	2:01:13 PM	40	HMI	PowerOn finished.

Fig. 4-6: Windows in the user interface

1	Main window	3	Message window
2	Option window		

Description

A maximum of 3 windows can be displayed at a time. The window selection key is used to toggle between the windows. The selected window is indicated by a blue background.

Window	Description
Main window	The main window displays either the Navigator or the selected or opened program.
Option window	Option windows are associated with individual functions or work sequences. They are not permanently visible in the user interface. It is not possible to have more than one option window open at any one time.
Message window	The message window displays error messages, system messages and dialog messages. The message window is not shown if there are no messages present, e.g. if all messages have been acknowledged.

4.2.3 Elements in the user interface

Input box

A value or a text can be entered.

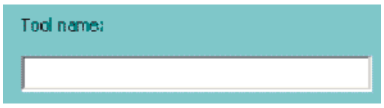
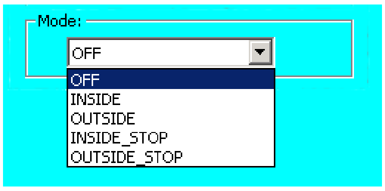

Tool name:

Fig. 4-7: Example of an input box

List box

A parameter can be selected from a list.

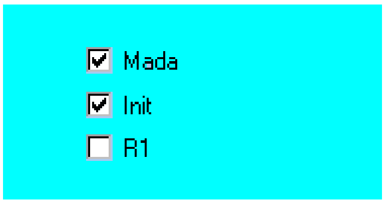

Mode:

OFF

Fig. 4-8: Example of an opened list box

Check box

One or more options can be selected.


☑ Mada

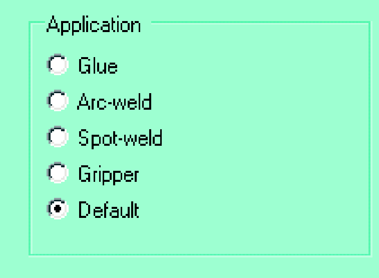
☑ Init

☐ R1

Fig. 4-9: Example of a check box

Option box

One option can be selected.


Application

○ Glue

○ Arc-weld

○ Spot-weld

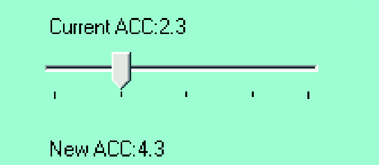
○ Gripper

● Default

Fig. 4-10: Example of an option box

Slider

A value on a scale can be set.

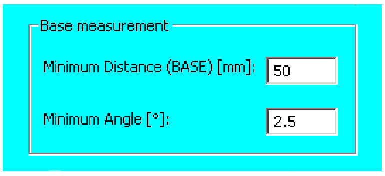

Current ACC:2.3

New ACC:4.3

Fig. 4-11: Example of a slider

Group

Boxes can be arranged in groups. A group is indicated by a frame. The name of the group is generally indicated in the top left-hand corner of the frame.



Base measurement

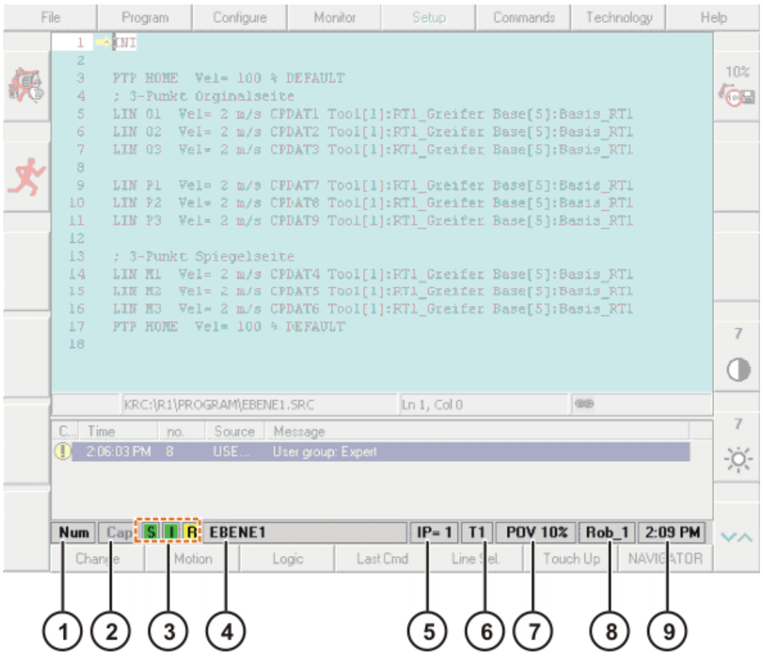
Minimum Distance (BASE) [mm]: 50

Minimum Angle [°]: 2.5

Fig. 4-12: Example of a group

4.2.4 Status bar

Overview



```
INI

PTP HOME  Vel= 100 % DEFAULT
; 3-Punkt Orginalseite
LIN O1  Vel= 2 m/s CPDAT1 Tool[1]:RT1_Greifer Base[5]:Basis_RT1
LIN O2  Vel= 2 m/s CPDAT2 Tool[1]:RT1_Greifer Base[5]:Basis_RT1
LIN O3  Vel= 2 m/s CPDAT3 Tool[1]:RT1_Greifer Base[5]:Basis_RT1

LIN P1  Vel= 2 m/s CPDAT7 Tool[1]:RT1_Greifer Base[5]:Basis_RT1
LIN P2  Vel= 2 m/s CPDAT8 Tool[1]:RT1_Greifer Base[5]:Basis_RT1
LIN P3  Vel= 2 m/s CPDAT9 Tool[1]:RT1_Greifer Base[5]:Basis_RT1

; 3-Punkt Spiegelseite
LIN M1  Vel= 2 m/s CPDAT4 Tool[1]:RT1_Greifer Base[5]:Basis_RT1
LIN M2  Vel= 2 m/s CPDAT5 Tool[1]:RT1_Greifer Base[5]:Basis_RT1
LIN M3  Vel= 2 m/s CPDAT6 Tool[1]:RT1_Greifer Base[5]:Basis_RT1
PTP HOME  Vel= 100 % DEFAULT

```

KRC:\R1\PROGRAM\EBENE1.SRC Ln 1, Col 0

C...	Time	no.	Source	Message
!	2:06:03 PM	8	USE...	User group: Expert

Fig. 4-13: Status bar in the user interface

1 Status of the numeric keypad

2 Upper-case/lower-case status

3 **S**: Status of the Submit interpreter

I/O: Status of the drives

R: Status of the program

4 Name of the selected program

5 Number of the current block

6 Current operating mode

7 Current override setting

8 Robot name

9 System time

Description

Icon	Description
Num	The numeric function of the numeric keypad is active.
Num	The control function of the numeric keypad is active.
Cap	Upper-case characters are active.
Cap	Lower-case characters are active.

Icon	Color	Description
S	Gray	Submit interpreter is deselected.
S	Red	Submit interpreter has been stopped.
S	Green	Submit interpreter is running.
I	Green	Drives ready.
I	Red	Drives not ready.
R	Gray	No program is selected.
R	Yellow	The block pointer is situated on the first line of the selected program.
R	Green	The program is selected and is being executed.
R	Red	The selected and started program has been stopped.
R	Black	The block pointer is situated on the last line of the selected program.

4.2.5 Calling online help

Description

Help texts are available for the following user interface elements:

- Messages
- Inline forms
- Error display
- Logbook entries

Procedure

1. Select, or position the cursor in, the element for which a help text is to be displayed.
2. Select the menu sequence **Help** > **Online help**.

   The help text for the element is displayed.

Alternative procedure

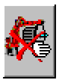
- Select the menu sequence **Help** > **Online help - Contents/Index**.

  You can search for a help text in the **Contents** and **Index** tabs.

4.2.6 Setting the brightness and contrast of the user interface

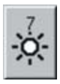
Precondition

- The following status key must be displayed for the jog mode:

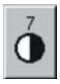


Procedure

- Set the brightness using the following status key:



- Set the contrast using the following status key:



4.3 Switching on the robot controller and starting the KSS

Procedure

- Turn the main switch on the robot controller to ON.

  The operating system and the KSS start automatically.

If the KSS does not start automatically, e.g. because the Startup function has been disabled, execute the file StartKRC.exe in the directory C:\KRC\BIN.

If the robot controller is logged onto the network, the start may take longer.

4.4 Restarting the KSS

Precondition

- Operating mode T1 or T2.

Procedure

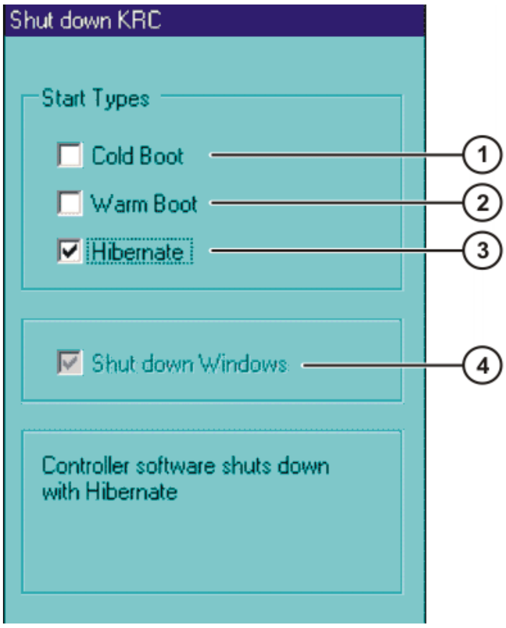
1. Select the menu sequence **File** > **Shut down KRC**.
2. Select the start type.
3. Press the **Shut Down** softkey. Confirm the request for confirmation with **Yes**.

   The KSS is shut down and then automatically restarts immediately with the selected start type.

> ⚠ **CAUTION** If the KSS has been restarted with the command **Shut down KRC**, the main switch on the robot controller must not be actuated during the rebooting sequence. System files may otherwise be destroyed.

> **i** If the robot controller detects a system error or modified data, the KSS always starts with a cold start – irrespective of the selected start type.

Description



Shut down KRC

Start Types

□ Cold Boot

□ Warm Boot

☑ Hibernate

☑ Shut down Windows

Controller software shuts down with Hibernate

Fig. 4-14: Option window “Shut down KRC”

Item	Description
1	The next start is a cold start.
2	The next start is a warm start.

Item	Description
3	The next start is a start after “Hibernate”.
4	Windows is rebooted. This option is only available in the user group “Expert”. **Shut down Windows** is automatically active if **Hibernate** has been selected. In this case, Windows is also restarted with Hibernate.

The following softkeys are available:

Softkey	Description
Shut Down	KSS is rebooted. If the option **Shut down Windows** is active, then Windows is also restarted.
Cancel	Closes the window. The settings are not saved.

4.5 Defining the start type for KSS

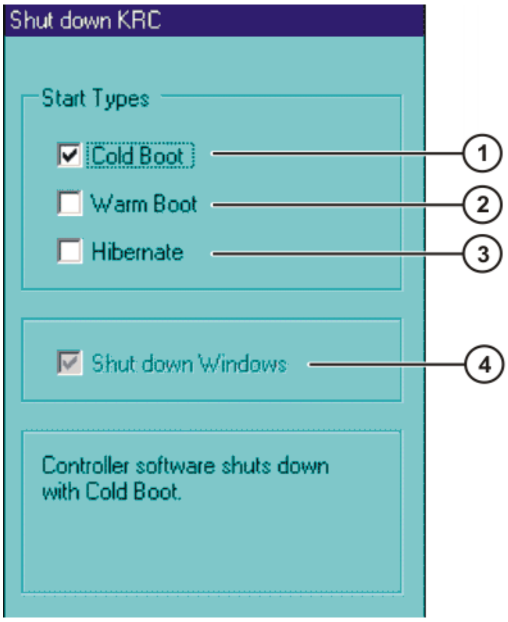
This function defines how the KSS starts after a power failure. A power failure and start are generally triggered by switching the main switch on the robot controller off and on.

If the robot controller detects a system error or modified data, the KSS always starts with a cold start – irrespective of the selected start type.

Procedure

1. Select the menu sequence **Configure** > **On/Off Options** > **Start Types**.
2. Select the start type.
3. Press the **OK** softkey.

Description



Shut down KRC

Start Types

☑ Cold Boot

□ Warm Boot

□ Hibernate

☑ Shut down Windows

Controller software shuts down with Cold Boot.

Fig. 4-15: Option window “Start types”

Item	Description
1	Cold start
2	Warm start

Item	Description
3	Hibernate
4	Windows is rebooted. This option cannot be modified in the option window **Start Types**. If **Hibernate** has been selected, Windows is also restarted with Hibernate.

The following softkeys are available:

Softkey	Description
OK	Saves the settings and closes the window.
Cancel	Closes the window. The settings are not saved.

4.6 Start types

Start type	Description
Cold start	After a cold start the robot controller displays the Navigator. No program is selected. The controller is completely reinitialized, e.g. all user outputs are set to FALSE.
Warm start	After a warm start, the previously selected robot program can be resumed. The state of the kernel system: programs, block pointer, variable contents and outputs, is completely restored.
Hibernate	The response is like that for the warm start. Additionally, after Hibernate, all programs that were open parallel to the robot controller are reopened and have the same state that they had before the system was shut down. The last state of Windows is also restored.

4.7 Switching the robot controller off

Procedure

- Turn the main switch on the robot controller to OFF.

  The robot controller automatically backs up data.

⚠ **CAUTION** If the KSS has been restarted with the command **Shut down KRC**, the main switch on the robot controller must not be actuated during the rebooting sequence. System files may otherwise be destroyed.

4.8 Setting the user interface language

Procedure

1. Select the menu sequence **Configure** > **Tools** > **Language**.
2. Select the desired language. Confirm with **OK**.

Description

The following languages are available:

Chinese (simplified)	Dutch
Chinese (traditional)	Polish
Danish	Portuguese
German	Romanian
English	Russian
Finnish	Swedish
French	Slovakian

Italian	Spanish
Japanese	Czech
Korean	Turkish

4.9 Changing user group

Procedure

1. Select the menu sequence **Configure** > **User group**. The current user group is displayed.
2. Press the **Default** softkey to switch to the default user group. (**Default** is not available if the default user group is already selected.)

   Press the **Log On...** softkey to switch to a different user group. Select the desired user group and confirm with the **Log On...** softkey.

   If prompted: Enter password and confirm with the **Log On** softkey.

Description

Different functions are available in the KSS, depending on the user group. The following user groups are available as standard:

- **Operator**

  User group for the operator. This is the default user group.
- **User**

  User group for the operator
- **Expert**

  User group for the programmer. In this user group it is possible to switch to the Windows interface.
- **Administrator**

  The range of functions is the same as that for the user group “Expert”. It is additionally possible, in this user group, to integrate plug-ins into the robot controller.

Depending on what technology packages or other software options are installed, additional user groups may also be available.

Expert and **Administrator** are password-protected. The default password is “kuka”. The password can be changed.

(>>> 6.11 "Changing the password" Page 147)

By default, **Operator** and **User** are defined for the same target group. The range of functions of all user groups can be altered. Furthermore, additional user groups can be created.

(>>> 6.21 "Setting up a new user group and password" Page 170)

When the system is booted, the default user group is selected. If the mode is switched to AUT or AUT EXT, the robot controller switches to the default user group for safety reasons. If a different user group is desired, this must be selected subsequently.

If no actions are carried out in the user interface within a certain period of time, the robot controller switches to the default user group for safety reasons. The default setting is 300 s.

> The default settings and other log-on properties can be configured.
> (>>> 6.20 "Configuring the log-in" Page 166)

4.10 Disabling the robot controller

Description

The robot controller can be disabled. It is then disabled for all actions except logging back on.

The robot controller cannot be disabled in the default user group.

Precondition

- The default user group is not selected.

Procedure

1. Select the menu sequence **Configure** > **User group**.
2. Press **Lock**. The robot controller is then disabled for all actions except logging on. The current user group is displayed.
3. Log back on:
   - Log on as the default user: Press **Default**.
   - Log on as a different user: Press **Log On...**. Select the desired user group and confirm with the **Log On...** softkey.

     If prompted: Enter password and confirm with **Log On**.

If you log onto the same user group as before, all the windows and programs of the previous user remain open. No data are lost.
If you log onto a different user group, the windows and programs of the previous user may be closed. Data can be lost!

4.11 Switching to the operating system interface

Precondition

- User group “Expert”
- The NUM function of the numeric keypad is deactivated.

Procedure

Switch to a different application

1. Press the ALT key and hold it down.
2. Press the TAB key. A window opens, displaying all active applications.
3. Press TAB repeatedly until the desired application is selected. Release both keys. The application is displayed.
4. Pressing ALT + ESC returns to the previous application.

Open the Start menu of the operating system.

1. CTRL + ESC. The Start menu is opened.
2. Using the arrow keys, select the desired menu item and press the Enter key.

4.12 Mode selector switch

The industrial robot can be operated in the following modes:

- Manual Reduced Velocity (T1)
- Manual High Velocity (T2)
- Automatic (AUT)
- Automatic External (AUT EXT)

The operating mode is selected using the mode selector switch on the KCP. The switch is activated by means of a key which can be removed. If the key is removed, the switch is locked and the operating mode can no longer be changed.

If the operating mode is changed during operation, the drives are immediately switched off. The manipulator and any external axes (optional) are stopped with a STOP 0.

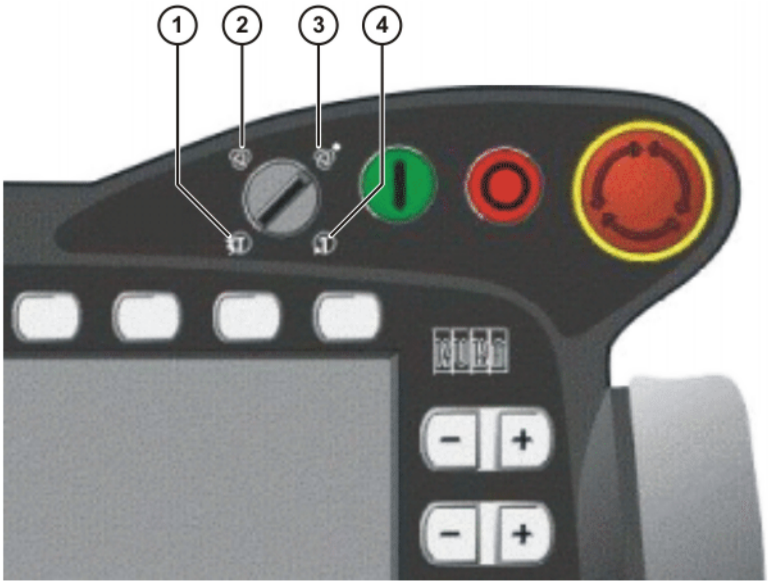


Fig. 4-16: Mode selector switch

1	T2 (Manual High Velocity)
2	AUT (Automatic)
3	AUT EXT (Automatic External)
4	T1 (Manual Reduced Velocity)

Operating mode	Use	Velocities
T1	For test operation, programming and teaching	■ Program verification: Programmed velocity, maximum 250 mm/s ■ Jog mode: Jog velocity, maximum 250 mm/s
T2	For test operation	■ Program verification: Programmed velocity
AUT	For industrial robots without higher-level controllers Only possible with a connected safety circuit	■ Program mode: Programmed velocity ■ Jog mode: Not possible
AUT EXT	For industrial robots with higher-level controllers, e.g. PLC Only possible with a connected safety circuit	■ Program mode: Programmed velocity ■ Jog mode: Not possible

4.13 Coordinate systems

Overview The following Cartesian coordinate systems are defined in the robot controller:

- WORLD
- ROBROOT
- BASE

- TOOL

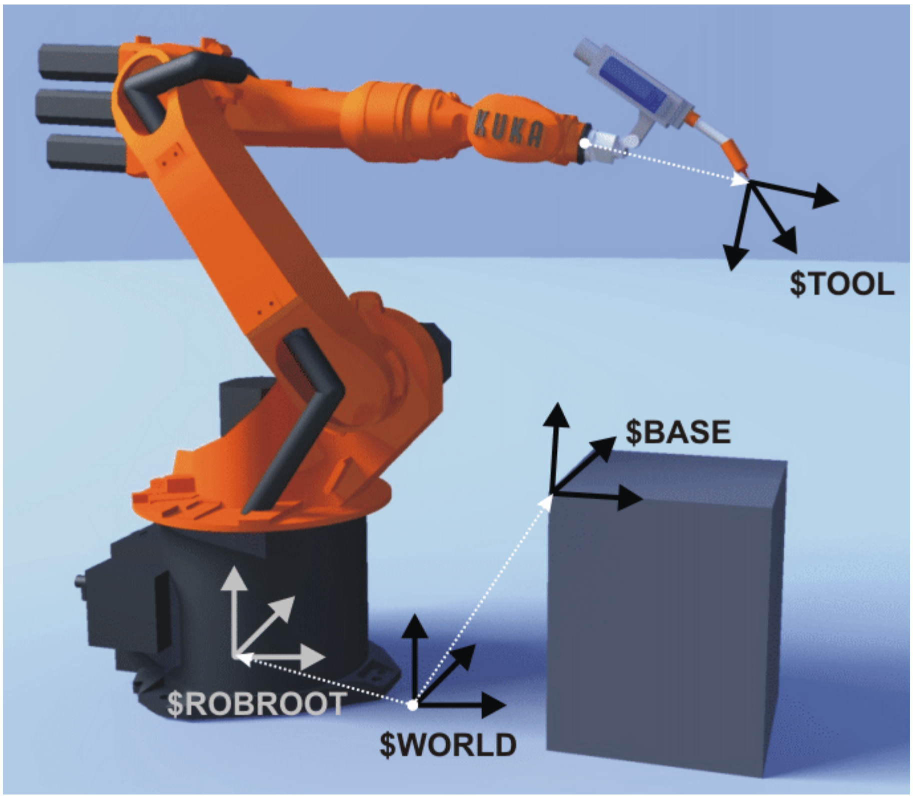


Fig. 4-17: Overview of coordinate systems

Description

WORLD

The WORLD coordinate system is a permanently defined Cartesian coordinate system. It is the root coordinate system for the ROBROOT and BASE coordinate systems.

By default, the WORLD coordinate system is located at the robot base.

ROBROOT

The ROBROOT coordinate system is a Cartesian coordinate system, which is always located at the robot base. It defines the position of the robot relative to the WORLD coordinate system.

By default, the ROBROOT coordinate system is identical to the WORLD coordinate system. $ROBROOT allows the definition of an offset of the robot relative to the WORLD coordinate system.

BASE

The BASE coordinate system is a Cartesian coordinate system that defines the position of the workpiece. It is relative to the WORLD coordinate system.

By default, the BASE coordinate system is identical to the WORLD coordinate system. It is offset to the workpiece by the user.

(>>> 5.5.4 "Base calibration" Page 104)

TOOL

The TOOL coordinate system is a Cartesian coordinate system which is located at the tool center point.

By default, the origin of the TOOL coordinate system is located at the flange center point. (In this case it is called the FLANGE coordinate system.) The TOOL coordinate system is offset to the tool center point by the user.

(>>> 5.5.2 "Tool calibration" Page 92)

Angles of rotation of the robot coordinate systems

Angle	Rotation about axis
Angle A	Rotation about the Z axis
Angle B	Rotation about the Y axis
Angle C	Rotation about the X axis

4.14 Jogging the lightweight robot

Description

There are 2 ways of jogging the robot:

- Cartesian jogging

  The TCP is jogged in the positive or negative direction along the axes of a coordinate system.
- Axis-specific jogging

  Each axis can be moved individually in a positive and negative direction.

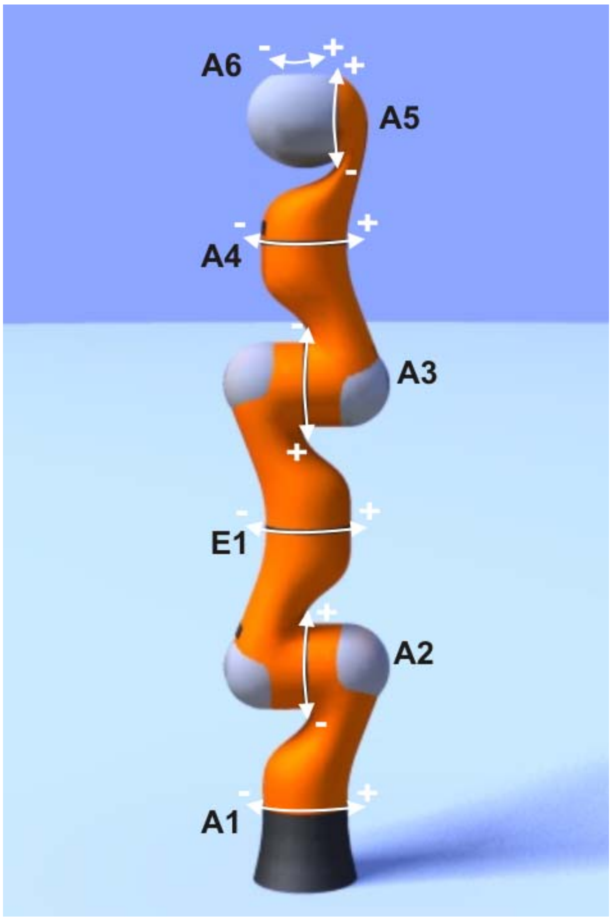


Fig. 4-18: Robot axes – home position for floor-mounted robot

Axes of the lightweight robot:

Software and KCP	Mechanical	Notation
A1	J1	A1J1
A2	J2	A2J2
E1	J3	E1J3
A3	J4	A3J4
A4	J5	A4J5
A5	J6	A5J6
A6	J7	A6J7

There are 2 operator control elements that can be used for jogging the robot:

- Space Mouse
- Jog keys

There are 6 jog keys available for 7 axes. Axis E1J3 is controlled and jogged as an external axis. At the same time, axis E1J3 is the zero space parameter.

Zero space motion

In the case of Cartesian jogging of the external axis, the robot executes a zero space motion, i.e. the other axes are so aligned (green arrow) that the position of the TCP (red arrow) remains unchanged.

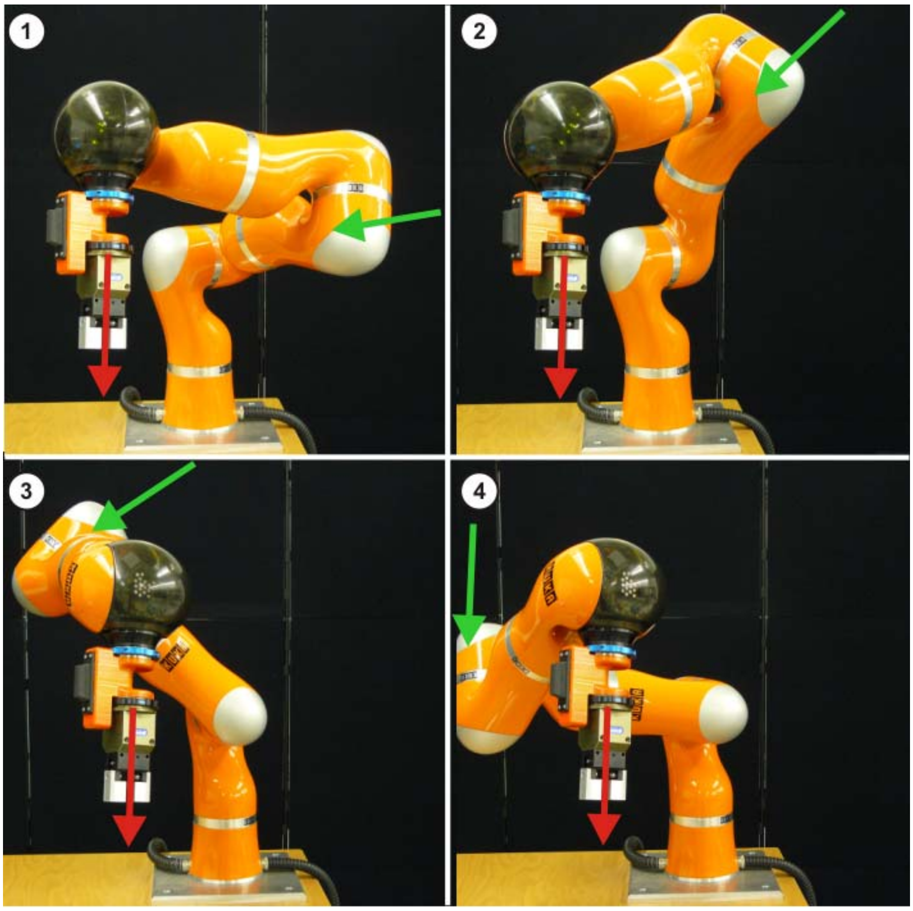


Fig. 4-19: Zero space motion

Status keys

The following axis combinations are available in the right-hand status key bar for jogging:

Status key	Description
	The robot can be moved with axis-specific or Cartesian jogging. External axis E1 is not moved.
	Axis-specific jogging: external axis E1 can be jogged in a positive and negative direction. Cartesian jogging: the robot executes a zero space motion.
	Axis-specific jogging: axes A1 to A3 and external axis E1 can be jogged individually in a positive and negative direction. Cartesian jogging: the TCP can be jogged in a positive and negative direction along the X, Y and Z axes of a coordinate system and the robot can execute a zero space motion.

Overview

	Cartesian jogging	Axis-specific jogging
Jog keys	(>>> 4.14.4 "Cartesian jogging of the robot with the jog keys" Page 59) (>>> 4.14.6 "Cartesian jogging of the external axis with the jog keys" Page 61)	(>>> 4.14.3 "Axis-specific jogging of the robot with the jog keys" Page 57) (>>> 4.14.5 "Axis-specific jogging of the external axis with the jog keys" Page 60)
Space Mouse	(>>> 4.14.9 "Cartesian jogging with the Space Mouse" Page 64)	Axis-specific jogging with the Space Mouse is possible, but is not described here.

4.14.1 Setting the jog override (HOV)

Description

Jog override is the velocity of the robot during jogging. It is specified as a percentage and refers to the maximum possible jog velocity. This is 250 mm/s.

Preparation

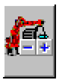
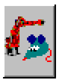
- Define the jog override intervals:

  Select the menu sequence **Configure** > **Jogging** > **Jog OV Steps**.

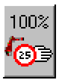
Active	Meaning
No	The override can be adjusted in 1% steps.
Yes	Intervals: 100%, 75%, 50%, 30%, 10%, 3%, 1%

Procedure

1. Select the jog mode "Jog keys" or "Space Mouse" in the left-hand status key bar:

    or 

2. Increase or reduce the override in the right-hand status key bar. The status key always indicates the current override as a percentage.

4.14.2 Selecting the tool and base

Description

A maximum of 16 TOOL and 32 BASE coordinate systems can be saved in the robot controller. One tool (TOOL coordinate system) and one base (BASE coordinate system) must be selected for Cartesian jogging.

Procedure

1. Select the menu sequence **Configure** > **Set tool/base**.
2. In the softkey bar, select whether a fixed tool is to be used:
   - **ext. Tool**: the tool is a fixed tool.
   - **Tool**: The tool is mounted on the mounting flange.
3. Enter the number of the desired tool in the box **Tool no.**.
4. Enter the number of the desired base in the box **Base No.**.
5. Press **OK**.

4.14.3 Axis-specific jogging of the robot with the jog keys

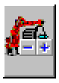
Precondition

- Operating mode T1 or T2.

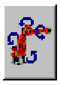
If the robot is unable to keep to the configured stiffness values, e.g. if an axis has a stiffness value of zero, jogging with stiffness control is deactivated for this axis and an error message is generated.

Procedure

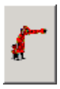
1. Select the jog mode “Jog keys” in the left-hand status key bar:



2. Select axis-specific jogging in the right-hand status key bar:



3. Select the axis combination “Robot axes” in the right-hand status key bar:



4. Set jog override.
5. Hold down the enabling switch.
6. Axes A1 to A6 are displayed in the right-hand status key bar.

   Press the Plus or Minus status key to move an axis in the positive or negative direction.

Right hand rule

The right hand rule provides a simple means of determining the direction of rotation of the axes. With a lightly clenched right fist, point with the thumb from the robot base along the axes towards the TCP. The fingers point in the positive direction of rotation.

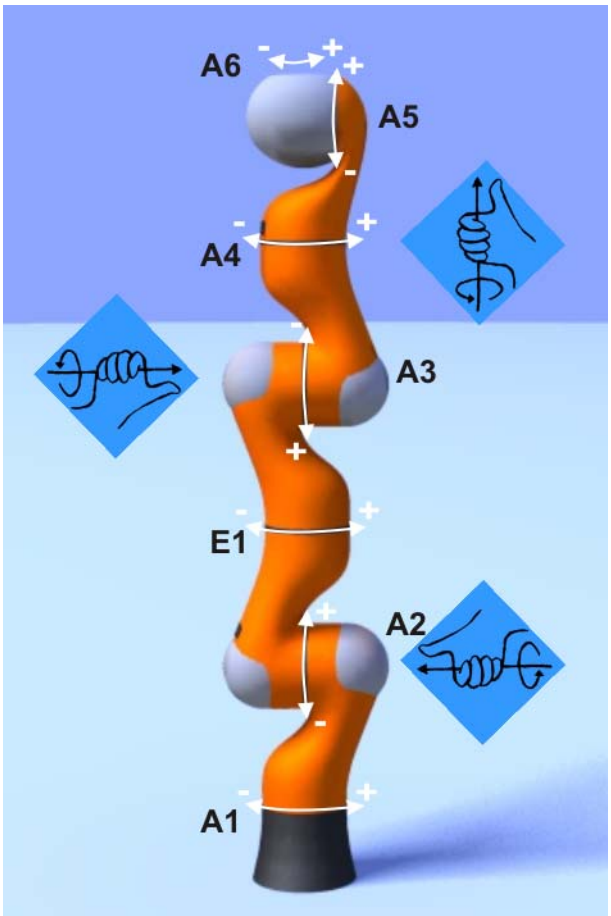


Fig. 4-20: Direction of rotation of the axes – right hand rule

4.14.4 Cartesian jogging of the robot with the jog keys

Precondition

- Tool and base have been selected.

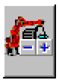
  (>>> 4.14.2 "Selecting the tool and base" Page 57)

- Operating mode T1 or T2

> If the robot is unable to keep to the configured stiffness values, jogging with stiffness control is deactivated and an error message is generated, e.g. the selected coordinate system must correspond to the reference coordinate system $STIFFNESS.FRAMETYPE if the Cartesian spring stiffness $STIFFNESS.CPSTIFFNESS is very low.
> (>>> 6.2 "Configuring servo controllers via $STIFFNESS" Page 113)

Procedure

1. Select the jog mode “Jog keys” in the left-hand status key bar:

2. Select the coordinate system in the right-hand status key bar.
3. Select the axis combination “Robot axes” in the right-hand status key bar:

4. Set jog override.
5. Hold down the enabling switch.
6. The following status keys are displayed in the right-hand status key bar:

   X, **Y**, **Z**: for the linear motions along the axes of the selected coordinate system

   A, **B**, **C**: for the rotational motions about the axes of the selected coordinate system

   Press the Plus or Minus status key to move the robot in the positive or negative direction. External axis E1 remains stiff and does not take part in the Cartesian motion.

> The position of the robot during jogging can be displayed: select the menu sequence **Monitor** > **Rob. Position**.

4.14.5 Axis-specific jogging of the external axis with the jog keys

Precondition

- Operating mode T1 or T2.

> If the robot is unable to keep to the configured stiffness values, e.g. if an axis has a stiffness value of zero, jogging with stiffness control is deactivated for this axis and an error message is generated.

Procedure

1. Select the jog mode “Jog keys” in the left-hand status key bar:
2. Select axis-specific jogging in the right-hand status key bar:
3. Select the axis combination “External axis” or “Robot axes with external axis” in the right-hand status key bar:

   or
4. Set jog override.
5. Hold down the enabling switch.
6. External axis E1 or axes A1 to A3 plus external axis E1 are displayed in the right-hand status key bar.

   Press the Plus or Minus status key to move an axis in the positive or negative direction.

4.14.6 Cartesian jogging of the external axis with the jog keys

Precondition

- Operating mode T1 or T2

> If the robot is unable to keep to the configured stiffness values, jogging with stiffness control is deactivated and an error message is generated, e.g. the selected coordinate system must correspond to the reference coordinate system $STIFFNESS.FRAMETYPE if the Cartesian spring stiffness $STIFFNESS.CPSTIFFNESS is very low.
> (>>> 6.2 "Configuring servo controllers via $STIFFNESS" Page 113)

Procedure

1. Select the jog mode “Jog keys” in the left-hand status key bar:

2. Select the coordinate system in the right-hand status key bar.
3. Select the axis combination “External axis” or “Robot axes with external axis” in the right-hand status key bar:

or

4. Set jog override.
5. Hold down the enabling switch.
6. The following status keys are displayed in the right-hand status key bar:

   E1: for the zero space motion

   or

   X, **Y**, **Z**: for the linear motions along the axes of the selected coordinate system; external axis E1 remains stiff and does not take part in the Cartesian motion.

   E1: for the zero space motion

   Press the Plus or Minus status key to move the robot in the positive or negative direction.

4.14.7 Configuring the Space Mouse

Procedure

1. Select the menu sequence **Configure** > **Jogging** > **Mouse configuration**.
2. **Axis selection**: Select whether the TCP is to be moved using translational motions, rotational motions, or both. The following softkeys are available:

   6D; **XYZ**; **ABC**
3. **Dominant mode**: Activate or deactivate. The following softkeys are available:

   Dominant; **Not dom.**
4. The softkey **Close** saves the current settings and closes the window.

Axis selection description

Softkey	Description
XYZ	The robot can only be moved by pulling or pushing the Space Mouse. The following motions are possible with Cartesian jogging: - Translational motions in the X, Y and Z directions
ABC	The robot can only be moved by rotating or tilting the Space Mouse. The following motions are possible with Cartesian jogging: - Rotational motions about the X, Y and Z axes
6D	The robot can be moved by pulling, pushing, rotating or tilting the Space Mouse. The following motions are possible with Cartesian jogging: - Translational motions in the X, Y and Z directions - Rotational motions about the X, Y and Z axes

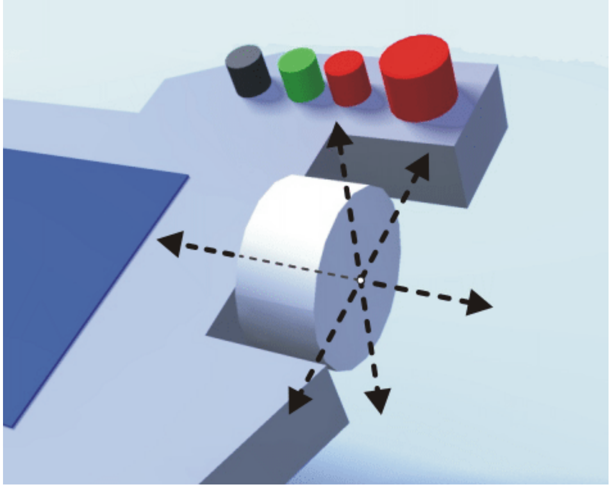


Fig. 4-21: Pushing and pulling the Space Mouse

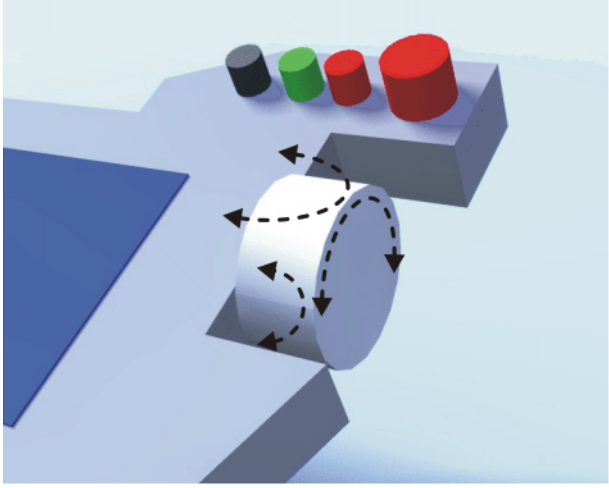


Fig. 4-22: Rotating and tilting the Space Mouse

Description of dominant mode

Depending on the dominant mode, the Space Mouse can be used to move just one axis or several axes simultaneously.

Softkey	Description
Dominant	Activates the dominant mode. Only the coordinate axis with the greatest deflection of the Space Mouse is moved.
Not dom.	Deactivates the dominant mode. Depending on the axis selection, either 3 or 6 axes can be moved simultaneously.

4.14.8 Defining the alignment of the Space Mouse

Description

The functioning of the Space Mouse can be adapted to the location of the user so that the motion direction of the TCP corresponds to the deflection of the Space Mouse.

The location of the user is specified in degrees. The reference point for the specification in degrees is the junction box on the base frame. The position of the robot arm or axes is irrelevant.

Default setting: 0°. This corresponds to a user standing opposite the junction box.

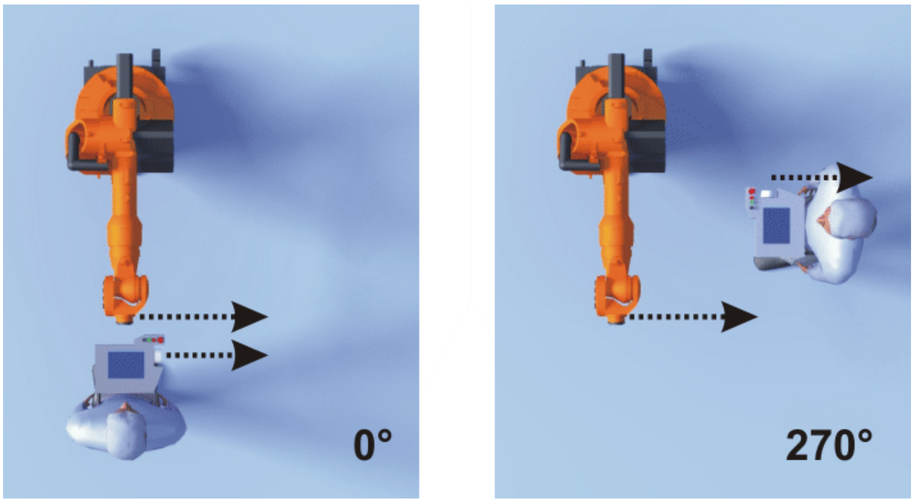


Fig. 4-23: Space Mouse: 0° and 270°

Precondition

- Operating mode T1 or T2.

Procedure

1. Select the menu sequence **Configure** > **Jogging** > **Mouse position**.
2. The alignment of the Space Mouse can be modified using the “**+**” or “**-**” softkey.

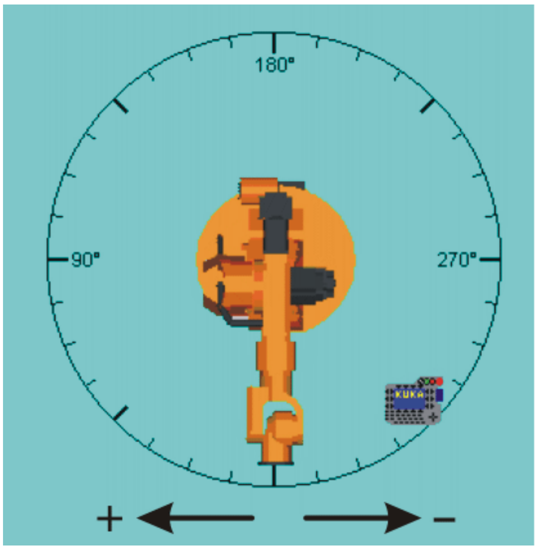


Fig. 4-24: Option window for aligning the Space Mouse

3. The softkey **Close** saves the current settings and closes the window.

> Switching to Automatic or Automatic External mode automatically resets the alignment of the Space Mouse to 0°.

4.14.9 Cartesian jogging with the Space Mouse

Precondition

- Tool and base have been selected.

  (>>> 4.14.2 "Selecting the tool and base" Page 57)

- The Space Mouse is configured.

  (>>> 4.14.7 "Configuring the Space Mouse" Page 61)

- The alignment of the Space Mouse has been defined.

  (>>> 4.14.8 "Defining the alignment of the Space Mouse" Page 63)

- Operating mode T1 or T2

> If the robot is unable to keep to the configured stiffness values, jogging with stiffness control is deactivated and an error message is generated, e.g. the selected coordinate system must correspond to the reference coordinate system $STIFFNESS.FRAMETYPE if the Cartesian spring stiffness $STIFFNESS.CPSTIFFNESS is very low.
> (>>> 6.2 "Configuring servo controllers via $STIFFNESS" Page 113)

Procedure

1. Select the following jog mode in the left-hand status key bar:

2. Select the coordinate system in the right-hand status key bar.
3. Set jog override.
4. Hold down the enabling switch.

5. Move the robot in the desired direction using the Space Mouse. External axis E1 remains stiff and does not take part in the Cartesian motion.

> The position of the robot during jogging can be displayed: select the menu sequence **Monitor** > **Rob. Position**.

4.14.10 Incremental jogging

Description

Incremental jogging makes it possible to move the robot a defined distance, e.g. 10 mm or 3°. The robot then stops by itself.

Incremental jogging can be activated for jogging with the jog keys. Incremental jogging is not possible in the case of jogging with the Space Mouse.

Areas of application:

- Positioning of equidistant points
- Moving a defined distance away from a position, e.g. in the event of a fault
- Mastering with the dial gauge

Status keys

Incremental jogging is set using the following status keys in the right-hand status key bar:

State	Description
→	Incremental jogging switched off
10° 100mm	Increment = 100 mm or 10°
3° 10mm	Increment = 10 mm or 3°
1° 1mm	Increment = 1 mm or 1°
0.005° 0.1mm	Increment = 0.1 mm or 0.005°

Increments in mm:

- Valid for Cartesian jogging in the X, Y or Z direction.

Increments in degrees:

- Valid for Cartesian jogging in the A, B or C direction.
- Valid for axis-specific jogging.

Procedure

1. Select the jog mode “Jog keys” in the left-hand status key bar:
2. Set the size of the increment in the right-hand status key bar.
3. Jog the robot using the jog keys. Jogging can be Cartesian or axis-specific.

   Once the set increment has been reached, the robot stops.

   (>>> 4.14.3 "Axis-specific jogging of the robot with the jog keys" Page 57)

   (>>> 4.14.4 "Cartesian jogging of the robot with the jog keys" Page 59)

> If the robot motion is interrupted, e.g. by releasing the enabling switch, the interrupted increment is not resumed with the next motion; a new increment is started instead.

4.15 Displaying status keys for the lightweight robot

Procedure

- Select the menu sequence **Configure** > **Status keys** > **DLR**.

Description

The following status keys are available in the left-hand status key bar:

Status key	Description
	Set-up (>>> 5.4.1 "Displaying softkeys for mastering" Page 87) This status key is only available in the user group “Expert” or higher.
	Controller (>>> 4.16 "Manually selecting servo controllers" Page 66)

4.16 Manually selecting servo controllers

Procedure

1. Display status keys for the lightweight robot.
   (>>> 4.15 "Displaying status keys for the lightweight robot" Page 66)
2. Press the **Controller** status key in the left-hand status key bar.
   The softkeys for controller selection are displayed.
3. Select the desired controller by pressing the corresponding softkey.
4. Jog the robot with the jog keys to activate the controller.
   Exception **GravComp**: the controller is activated when the enabling switch is pressed.

> ⚠ **CAUTION** In the case of gravitation compensation, the velocity in T1 mode is not restricted to a maximum of 250 mm/s. Higher velocities can be reached.

> ⚠ **CAUTION** If the robot is moved with gravitation compensation activated, the safety precautions for manual guidance must be observed. Failure to observe these precautions may result in physical injuries and damage to property.
> (>>> 4.17 "Manual guidance of the lightweight robot" Page 68)

The following softkeys are available:

Softkey	Description
GravComp	**Gravity compensation** The payload is gravity-compensated. The robot can only be moved by means of manual guidance. For this, T1 mode must be set and the enabling switch must be pressed. The safety precautions for manual guidance must be observed. (>>> 4.17 "Manual guidance of the lightweight robot" Page 68) **Note:** Automatic mode, moving the robot in program mode and manual jogging with the jog keys and Space Mouse are not possible. This controller is suitable for teaching points. (>>> 6.1.4 "Gravitation compensation" Page 112)
Position	**Position controller** The robot does not exhibit compliance. The robot moves to the programmed position. This controller is the default controller. It is suitable for accurate positioning. (>>> 6.1.1 "Position controller" Page 111)
Stiff	**Cartesian stiffness controller** The robot is in compliance control, i.e. it can be moved by external force. In the absence of external force, the robot returns to its original position; compliance similar to that of a spring. This controller is suitable for establishing and maintaining contact with the surroundings. ■ Insert pin. ■ Turn crank. ■ Trace profile of a surface with a probe pin. (>>> 6.1.2 "Cartesian stiffness controller" Page 111)

Softkey	Description
Txyz free	**Cartesian stiffness controller** The payload is gravity-compensated. The payload can only be moved by means of a translational motion. **Note:** The robot cannot be moved in the free directions by means of Cartesian jogging with the jog keys. Cartesian jogging in the non-free directions and axis-specific jogging with the jog keys is possible, but is not recommended. The robot motions do not correspond to the jog keys and are not predictable. (>>> 6.1.2 "Cartesian stiffness controller" Page 111)
TxyzRz free	**Cartesian stiffness controller** The payload is gravity-compensated. The payload can be moved by means of translational motions and rotational motions about the Z axis (angle A). **Note:** The robot cannot be moved in the free directions by means of Cartesian jogging with the jog keys. Cartesian jogging in the non-free directions and axis-specific jogging with the jog keys is possible, but is not recommended. The robot motions do not correspond to the jog keys and are not predictable. This controller is suitable, after teaching a point, for teaching additional points with the same orientation. (>>> 6.1.2 "Cartesian stiffness controller" Page 111)

4.17 Manual guidance of the lightweight robot

Manual guidance of the robot is only permissible in T1 mode and only with reduced power (limited to 80 watts).

> Whether or not the power has been limited to 80 watts can be polled using the system variable $POWERLIMIT_AUX.
> $POWERLIMIT_AUX=TRUE: limitation is active.

Manual guidance must be safeguarded by means of an additional safe (PL d, Cat. 3) three-position enabling switch. This enabling switch is not included in the scope of supply and must be mounted on the robot flange by the customer.

> ⚠ **CAUTION** The enabling switch must be mounted in such a way that the guiding hand cannot be injured in the event of unexpected robot motions, e.g. the sudden closure of a gripper mounted on the flange.

In gravitation compensation, the velocity is limited by default to 600 mm/s. This value can be changed within a range from 0 to 1000 mm/s by changing the variable $SPEED_LIMIT_GRAVCOMP in the machine data. After a change, the robot controller must be restarted. If this velocity is exceeded, a STOP 0 is triggered. The choice of the limit velocity depends on the application and is determined for this application by the user's risk assessment.

> **NOTICE** The velocity monitoring during manual guidance does not conform to the requirements of EN ISO 10218-1.

If the actual load does not correspond to the load entered in the load data, the following steps must be performed in order to move the robot:

1. Press the enabling switch.

2. Confirm the dialog message on the KCP with **OK**.

The velocity of the robot is reduced to 250 mm/s in this case.

⚠ Unexpected, fast motion is possible during gravitation compensation!

⚠ **WARNING** During the direct human-robot interaction in manual guidance mode, the operator must do all he can to ensure his safety:

- The KCP must be secured against unauthorized operation.
- There must be no hazardous objects in the vicinity of the operator (e.g. scissors in shirt pocket).
- It must be ensured that hair, clothing, glasses, jewelry, etc., cannot be caught up in the robot.

4.18 Guiding an axis out of hardware limitation

Overview

The motion range of the axes is limited by means of software limit switches. If, during manual guidance, for example, an axis is pushed beyond this limitation, the axis brake is immediately applied. The robot can no longer be moved and a message such as *"SEN: LR2: Hardware limit stop"* is displayed (*LR2* = hardware limit stop for axis A2/J2).

In order to be able to move the robot again, the brake must be released using the LWR brake release device. The axis can then be guided away from the hardware limitation.

(>>> 4.18.1 "Enabling axes A1 to A4, E1" Page 69)

(>>> 4.18.2 "Enabling wrist axes A5 and A6" Page 70)

4.18.1 Enabling axes A1 to A4, E1

Precondition

- The brake release device is present.
- The robot is switched on and secured.

⚠ **WARNING** Unintentional robot motions can cause injuries and damage to property. If work is carried out on an operational robot, the robot must be secured by activating the EMERGENCY STOP button.
Warn all persons concerned before starting to put it back into operation.

ℹ If the brakes are opened with the robot switched off or during a power failure, the robot or the affected axis must be remastered.
(>>> 5.4.3 "Carrying out position mastering" Page 90)
(>>> 5.4.4 "Carrying out torque mastering" Page 90)

Procedure

1. Manually remove the cover over the axis drive unit.

ℹ To enable axis A1, the cover over the drive unit of axis A2 must be removed.

2. Using tweezers, pull out the cable with the brake connector. The cable is stowed loosely in the drive unit. (>>> Fig. 4-25)
3. Connect brake release device to brake connector.
4. Connect the brake release device to slot X63 on the robot controller.
5. Press the button on the brake release device to release the brake.

⚠ CAUTION The axis yields immediately. To prevent material damage and physical injuries, the robot must be secured before the brakes are released.

6. Only in the case of rotational axes: manually rotate the axis in the direction indicated by the arrow. (>>> Fig. 4-26)

NOTICE Rotational axes are only limited by means of the software limit switches and not by mechanical end stops. If rotational axes are rotated in the wrong direction, this can result in damage to the robot.

7. Only in the case of tilting axes: manually guide the axis away from the software limitation.
8. Disconnect the brake connector from the brake release device. The brake is applied and the robot remains in the current position.
9. Push the cable with the brake connector back into the drive unit.
10. Fasten cover over the drive unit.

Description

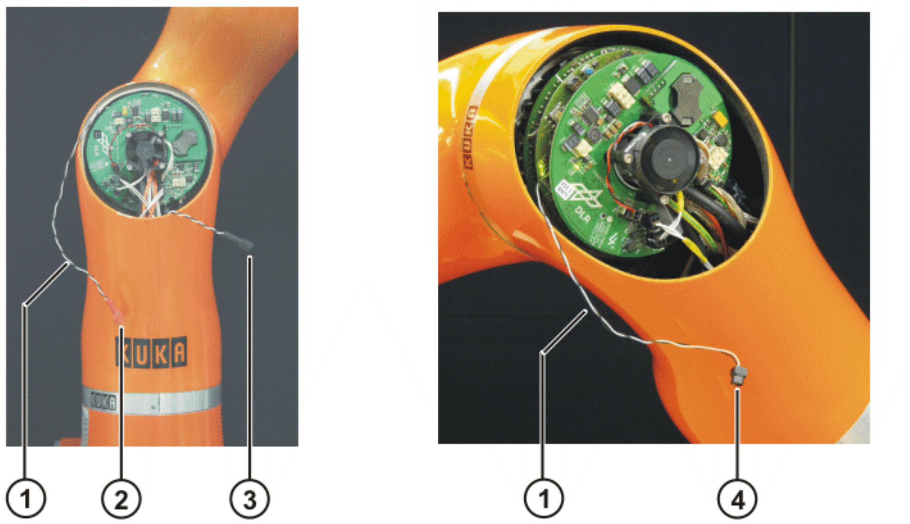


Fig. 4-25: Brake connectors, axes A1, A2, A3

1	Brake release cable	3	Brake connector A1
2	Brake connector A2	4	Brake connector A3

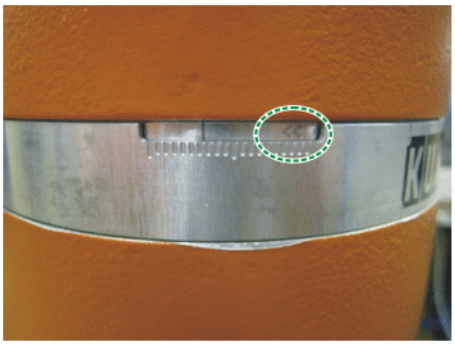


Fig. 4-26: Rotational axes – arrow indicating direction of rotation

4.18.2 Enabling wrist axes A5 and A6

Precondition

- The brake release device is present.

- The robot is switched on and secured.

⚠ **WARNING** Unintentional robot motions can cause injuries and damage to property. If work is carried out on an operational robot, the robot must be secured by activating the EMERGENCY STOP button.
Warn all persons concerned before starting to put it back into operation.

i If the brakes are opened with the robot switched off or during a power failure, the robot or the affected axis must be remastered.
(>>> 5.4.3 "Carrying out position mastering" Page 90)
(>>> 5.4.4 "Carrying out torque mastering" Page 90)

Procedure

1. Remove 1 Allen screw M3x8-8.8 and 2 internal TORX screws M2x16 and take off the covers over the drive unit of the wrist axes. (>>> Fig. 4-27)
2. Pull out brake connector. (>>> Fig. 4-28)
3. Connect brake release device to brake connector.
4. Connect the brake release device to slot X63 on the robot controller.
5. Press the button on the brake release device to release the brake.

⚠ **CAUTION** The axis yields immediately. To prevent material damage and physical injuries, the robot must be secured before the brakes are released.

6. Only for axis A6: manually rotate the axis in the direction indicated by the arrow beneath the mastering mark. (>>> Fig. 4-29)

NOTICE Rotational axes are only limited by means of the software limit switches and not by mechanical end stops. If rotational axes are rotated in the wrong direction, this can result in damage to the robot.

7. Only in the case of tilting axes: manually guide the axis away from the software limitation.
8. Disconnect the brake connector from the brake release device. The brake is applied and the robot remains in the current position.
9. Push the brake connector back in again.
10. Fasten covers over the drive unit.
11. Insert and tighten 1 Allen screw M3x8-8.8 and 2 internal TORX screws M2x16 by hand.

Description

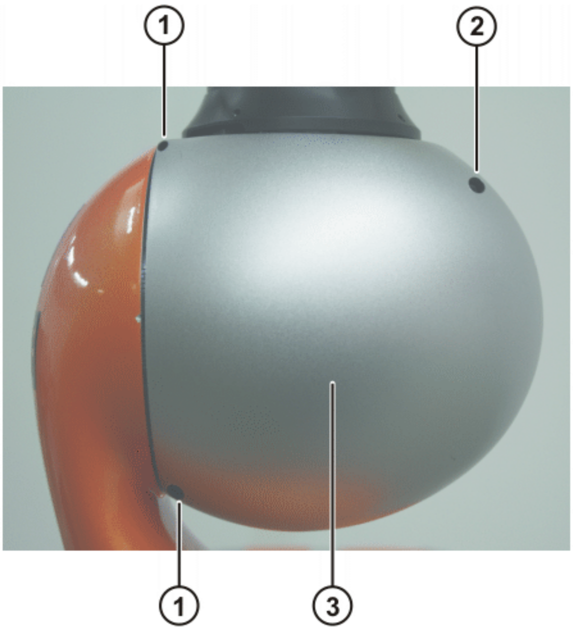


Fig. 4-27: Wrist axis drive unit cover

1 Internal TORX screw M2x16

2 Allen screw M3x8-8.8

3 Cover

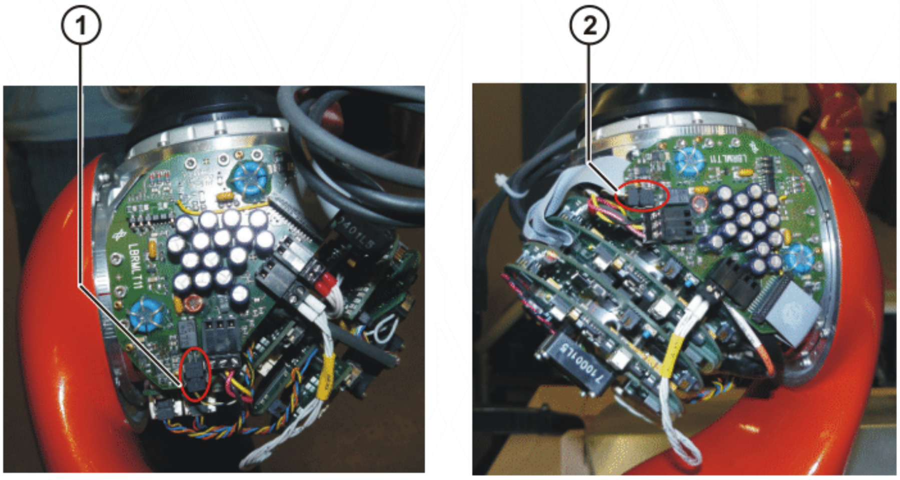


Fig. 4-28: Brake connectors, axes A5, A6

1 Brake connector A5

2 Brake connector A6

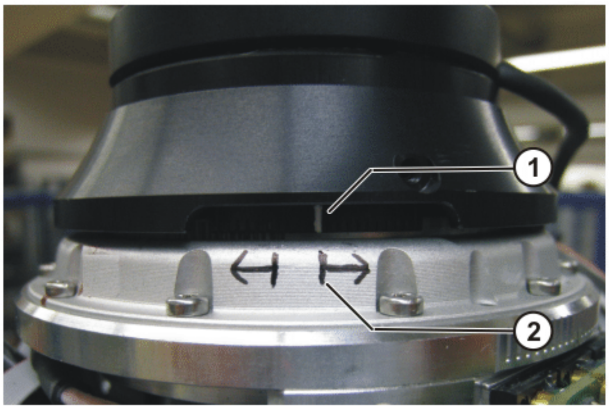


Fig. 4-29: Axis A6 – arrows indicating direction of rotation

1	Mastering mark	2	Direction of rotation

4.19 Bypassing workspace monitoring

Description

Workspaces can be configured for a robot. Workspaces serve to protect the system.

There are 2 types of workspace:

- The workspace is an exclusion zone.
  The robot may only move outside the workspace.
- Only the workspace is a permitted zone.
  The robot may not move outside the workspace.

Exactly what reactions occur when the robot violates a workspace depends on the configuration. (>>> 6.13 "Configuring workspaces" Page 150)

One possible reaction, for example, is that the robot stops and an error message is generated. The workspace monitoring must be bypassed in such a case. The robot can then move back out of the prohibited workspace.

Precondition

- User group “Expert”
- Operating mode T1

Procedure

1. Select the menu sequence **Configure** > **Miscellaneous** > **WorkSpace-Monitor** > **Override**.
2. Move the robot manually out of the prohibited workspace.
   Once the robot has left the prohibited workspace, the workspace monitoring is automatically active again.

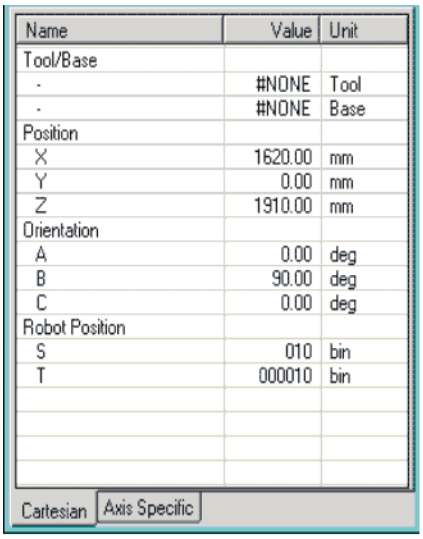
4.20 Monitor functions

4.20.1 Displaying the actual position

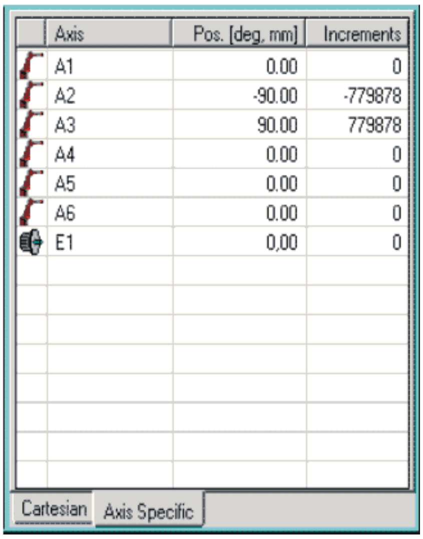
Procedure

- Select the menu sequence **Monitor** > **Rob. Position** > **Cartesian** or **Axis specific**.

Description



Name	Value	Unit
Tool/Base		
-	#NONE	Tool
-	#NONE	Base
Position		
X	1620.00	mm
Y	0.00	mm
Z	1910.00	mm
Orientation		
A	0.00	deg
B	90.00	deg
C	0.00	deg
Robot Position		
S	010	bin
T	000010	bin



Axis	Pos. [deg, mm]	Increments
A1	0.00	0
A2	-90.00	-779878
A3	90.00	779878
A4	0.00	0
A5	0.00	0
A6	0.00	0
E1	0,00	0

Fig. 4-30: Actual position: Cartesian and axis-specific

Cartesian

The current position (X, Y, Z) and orientation (A, B, C) of the TCP are displayed. In addition to this, the current TOOL and BASE coordinate systems and the Status and Turn are displayed.

Axis Specific

The current position of axes A1 to A6 are indicated in degrees and increments. If external axes are being used, the position of the external axes is also displayed.

The actual position can also be displayed while the robot is moving.

4.20.2 Displaying digital inputs/outputs

Precondition

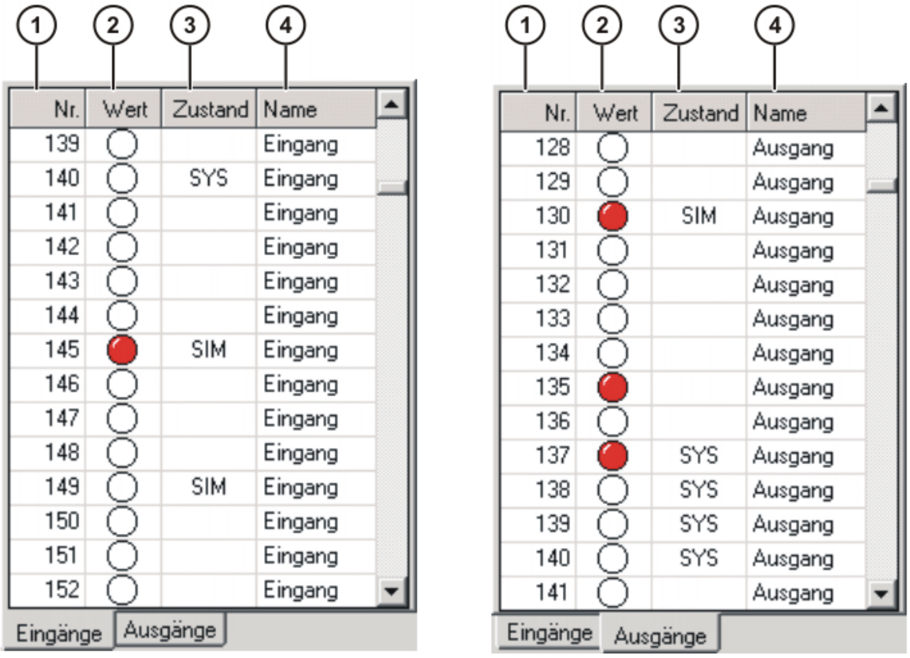
- The “NUM” function is active in the status bar.

Procedure

1. Select the menu sequence **Monitor** > **I/O** > **Digital Outputs** or **Digital Inputs**.
2. To display a specific input/output:
   - Select any cell in the **No.** column.
   - Enter the number using the numeric keypad.
   - Press the Enter key.

   The display jumps to the input/output with this number.

Description



Nr.	Wert	Zustand	Name
139	○		Eingang
140	○	SYS	Eingang
141	○		Eingang
142	○		Eingang
143	○		Eingang
144	○		Eingang
145	●	SIM	Eingang
146	○		Eingang
147	○		Eingang
148	○		Eingang
149	○	SIM	Eingang
150	○		Eingang
151	○		Eingang
152	○		Eingang

Eingänge Ausgänge

Nr.	Wert	Zustand	Name
128	○		Ausgang
129	○		Ausgang
130	●	SIM	Ausgang
131	○		Ausgang
132	○		Ausgang
133	○		Ausgang
134	○		Ausgang
135	●		Ausgang
136	○		Ausgang
137	●	SYS	Ausgang
138	○	SYS	Ausgang
139	○	SYS	Ausgang
140	○	SYS	Ausgang
141	○		Ausgang

Eingänge Ausgänge

Fig. 4-31: Digital inputs/outputs

Item	Description
1	Input/output number
2	Value of the input/output. The icon is red if the input or output is TRUE.
3	SIM entry: The input/output is simulated. SYS entry: The value of the input/output is saved in a system variable. This input/output is write-protected.
4	Name of the input/output

The following softkeys are available:

Softkey	Description
Tab+	Toggles between the **Inputs** and **Outputs** tabs.
Value	Toggles the selected output between TRUE and FALSE. Precondition: The enabling switch is pressed. This softkey is not available in AUT and AUT EXT modes or for inputs.
Name	The name of the selected input or output can be changed.

4.20.3 Displaying analog inputs/outputs

Precondition

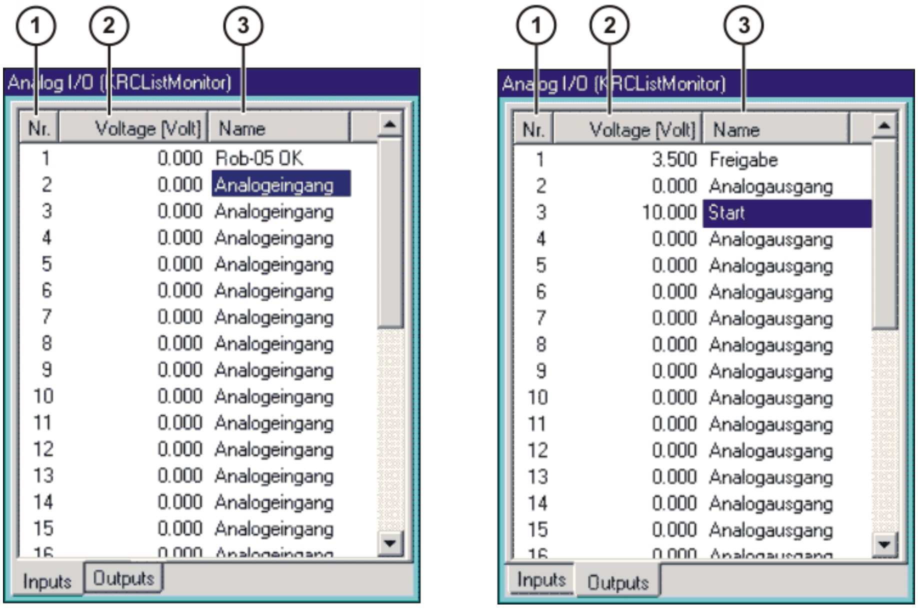
- The “NUM” function is active in the status bar.

Procedure

1. Select the menu sequence **Monitor** > **I/O** > **Analog I/O**.
2. To display a specific input/output:
   - Select any cell in the **No.** column.
   - Enter the number using the numeric keypad.

   The display jumps to the input/output with this number.

Description



Analog I/O (KRCListMonitor)

Nr.	Voltage [Volt]	Name
1	0.000	Rob-05 OK
2	0.000	Analogeingang
3	0.000	Analogeingang
4	0.000	Analogeingang
5	0.000	Analogeingang
6	0.000	Analogeingang
7	0.000	Analogeingang
8	0.000	Analogeingang
9	0.000	Analogeingang
10	0.000	Analogeingang
11	0.000	Analogeingang
12	0.000	Analogeingang
13	0.000	Analogeingang
14	0.000	Analogeingang
15	0.000	Analogeingang
16	0.000	Analogeingang

Inputs | Outputs

Analog I/O (KRCListMonitor)

Nr.	Voltage [Volt]	Name
1	3.500	Freigabe
2	0.000	Analogausgang
3	10.000	Start
4	0.000	Analogausgang
5	0.000	Analogausgang
6	0.000	Analogausgang
7	0.000	Analogausgang
8	0.000	Analogausgang
9	0.000	Analogausgang
10	0.000	Analogausgang
11	0.000	Analogausgang
12	0.000	Analogausgang
13	0.000	Analogausgang
14	0.000	Analogausgang
15	0.000	Analogausgang
16	0.000	Analogausgang

Inputs | Outputs

Fig. 4-32: Analog inputs/outputs

Item	Description
1	Input/output number
2	Input/output voltage - **-10 … 10 V**
3	Name of the input/output

The following softkeys are available:

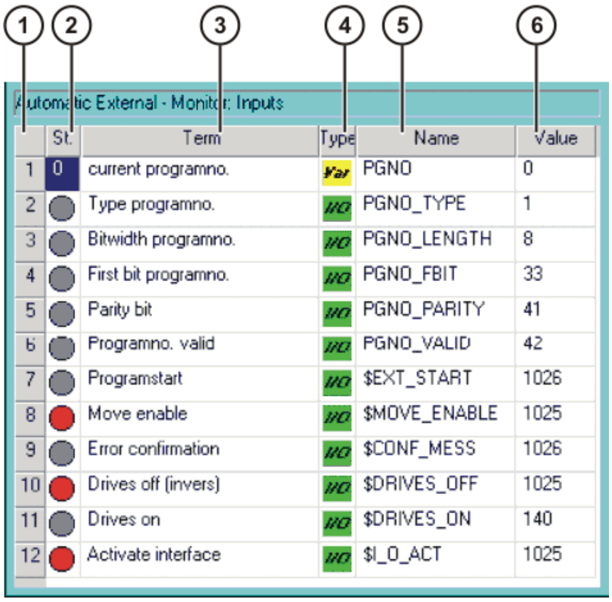
Softkey	Description
Tab +	Toggles between the **Inputs** and **Outputs** tabs.
Voltage	A voltage can be entered for the selected output. - **-10 … 10 V** This softkey is not available for inputs.
Name	The name of the selected input or output can be changed.

4.20.4 Displaying inputs/outputs for Automatic External

Procedure

- Select the menu sequence **Monitor** > **I/O** > **Automatic External**.

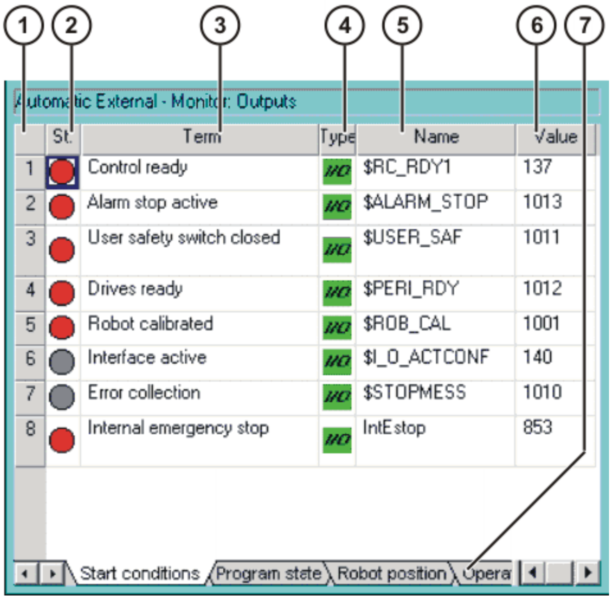
Description



Automatic External - Monitor: Inputs

	St.	Term	Type	Name	Value
1	0	current programno.	Var	PGNO	0
2	●	Type programno.	I/O	PGNO_TYPE	1
3	●	Bitwidth programno.	I/O	PGNO_LENGTH	8
4	●	First bit programno.	I/O	PGNO_FBIT	33
5	●	Parity bit	I/O	PGNO_PARITY	41
6	●	Programno. valid	I/O	PGNO_VALID	42
7	●	Programstart	I/O	$EXT_START	1026
8	🔴	Move enable	I/O	$MOVE_ENABLE	1025
9	●	Error confirmation	I/O	$CONF_MESS	1026
10	🔴	Drives off (invers)	I/O	$DRIVES_OFF	1025
11	●	Drives on	I/O	$DRIVES_ON	140
12	🔴	Activate interface	I/O	$I_O_ACT	1025

Fig. 4-33: Automatic External inputs (detail view)



Automatic External - Monitor: Outputs

	St.	Term	Type	Name	Value
1	🔴	Control ready	I/O	$RC_RDY1	137
2	🔴	Alarm stop active	I/O	$ALARM_STOP	1013
3	🔴	User safety switch closed	I/O	$USER_SAF	1011
4	🔴	Drives ready	I/O	$PERI_RDY	1012
5	🔴	Robot calibrated	I/O	$ROB_CAL	1001
6	●	Interface active	I/O	$I_O_ACTCONF	140
7	●	Error collection	I/O	$STOPMESS	1010
8	🔴	Internal emergency stop	I/O	IntEstop	853

Fig. 4-34: Automatic External outputs (detail view)

Item	Description
1	Number
2	State - **Gray:** inactive (FALSE) - **Red:** active (TRUE)
3	Long text name of the input/output
4	Type - **Green:** input/output - **Yellow:** variable or system variable ($...)
5	Name of the signal or variable

Item	Description
6	Input/output number or channel number
7	The outputs are thematically assigned to the following tabs: ■ Start conditions ■ Program status ■ Robot position ■ Operating mode

Columns 4, 5 and 6 are only displayed if the softkey **Details** has been pressed.

The following softkeys are available:

Softkey	Description
Config.	Switches to the configuration of the Automatic External interface. (>>> 6.23.2 "Configuring Automatic External inputs/outputs" Page 175)
Inputs/Outputs	Toggles between the windows for inputs and outputs.
Details/Normal	Toggles between the **Details** and **Normal** views.
Tab -/Tab +	Toggles between the tabs. This softkey is only available for outputs.

4.20.5 Displaying and modifying the value of a variable

This function is also called “variable correction”.

Procedure

1. Select the menu sequence **Monitor** > **Variable** > **Single**.

   The **Variable Overview - Single** window is opened.
2. Enter the name of the variable in the **Name** box.
3. If a program has been selected, it is automatically entered in the **Module** box.

   If a variable from a different program is to be displayed, enter the program as follows:

   /R1/*Program name*

   Do not specify a folder between /R1/ and the program name. Do not add a file extension to the file name.

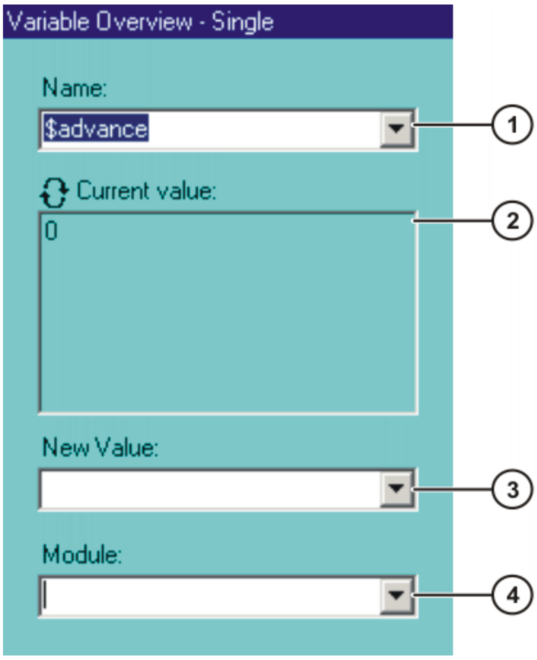
In the case of system variables, no program needs to be specified in the **Module** box.

4. Press the Enter key.

   The current value of the variable is displayed in the **Current value** box. If nothing is displayed, no value has yet been assigned to the variable.
5. Enter the desired value in the **New value** box.
6. Press the Enter key.

   The new value is displayed in the **Current value** box.

Description



Variable Overview - Single

Name:

$advance

Current value:

0

New Value:

Module:

Fig. 4-35: Variable Overview - Single window

Item	Description
1	Name of the variable to be modified.
2	This box has two states: ■ ⊘: The displayed value is refreshed when the Enter key is pressed. ■ ↻: The displayed value is refreshed automatically. **Note:** The automatic refresh function only works for data list variables, and not for runtime variables. Switching between the states: 1. Position the cursor in the **Name** or **Module** box. 2. SHIFT + Enter key
3	New value to be assigned to the variable.
4	Program in which the search for the variable is to be carried out. In the case of system variables, the **Module** box is irrelevant.

4.20.6 Displaying the state of a variable

Description

Variables can have the following states:

- UNKNOWN: The variable is unknown.
- DECLARED: The variable is declared.
- INITIALIZED: The variable is initialized.

Procedure

1. Select the menu sequence **Monitor** > **Variable** > **Single**.

   The **Variable display - Single** window is opened.
2. In the **Name** box, enter: =varstate("*name*")

   name = name of the variable whose state is to be displayed.
3. If a program has been selected, it is automatically entered in the **Module** box.

If a variable from a different program is to be displayed, enter the program as follows:

/R1/*Program name*

Do not specify a folder between /R1/ and the program name. Do not add a file extension to the file name.

> In the case of system variables, no program needs to be specified in the **Module** box.

4. Press the Enter key.

   The current state of the variable is displayed in the **Current value** box.

4.20.7 Displaying the variable overview and modifying variables

In the variable overview, variables are displayed in groups. The variables can be modified.

The number of groups and which variables they contain are defined in the configuration.

(>>> 6.9 "Configuring the variable overview" Page 145)

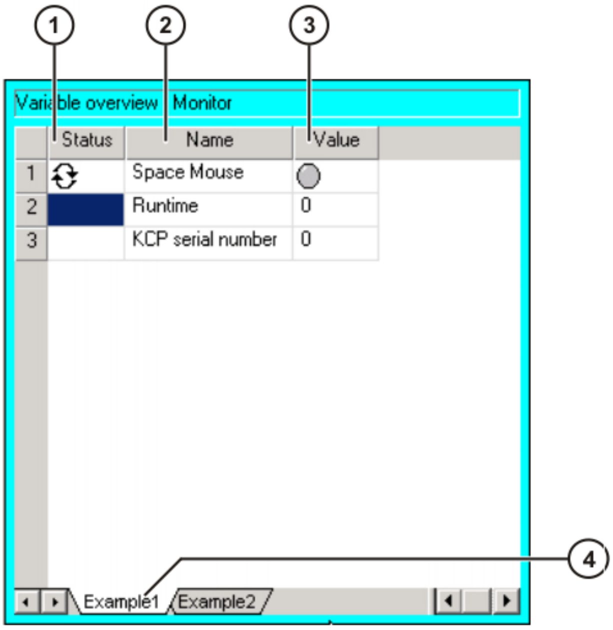
> Variables can only be displayed and modified in the user group “User” if these functions have been enabled in the configuration.

Procedure

1. Select the menu sequence **Monitor** > **Variable** > **Overview** > **Display**.

   The **Variable overview - Monitor** window is opened.
2. Select the desired group by pressing **Tab +**.
3. Select the cell to be modified. Carry out modification using the softkeys.
4. Press **OK** to save the change and close the window.

Description



	Status	Name	Value
1	↻	Space Mouse	○
2		Runtime	0
3		KCP serial number	0

Fig. 4-36: Variable overview - Monitor window

Item	Description
1	Arrow symbol ⟳: If the value of the variable changes, the display is automatically refreshed. No arrow symbol: The display is not automatically refreshed.
2	Descriptive name
3	Value of the variable. In the case of inputs/outputs, the state is indicated: ■ **Gray:** inactive (FALSE) ■ **Red:** active (TRUE)
4	There is one tab per group.

The following softkeys are available:

Softkey	Description
Config.	Switches to the configuration of the variable overview. (>>> 6.9 "Configuring the variable overview" Page 145) This softkey is not available in the user group “User”.
Tab+	Switches to the next group.
Refresh all	Refreshes the display.
Cancel Info	Deactivates the automatic refreshing function.
Start info	Activates the automatic refreshing function. A maximum of 12 variables per group can be refreshed automatically.
Edit	Switches the current cell to edit mode so that the name or value can be modified. In the **Value** column, this softkey changes the state of inputs/outputs (TRUE/FALSE). This softkey is only available in the user group “User” if it has been enabled in the configuration. **Note:** The values of write-protected variables cannot be changed.

4.20.8 Displaying calibration data

Procedure

1. Select the menu sequence **Setup** > **Measure** > **Measurement Points** and the desired menu item:
   - **Tool type**
   - **Base type**
   - **External axis**
2. Enter the number of the tool, base or external kinematic system.

   The calibration method and the calibration data are displayed.

4.20.9 Displaying information about the robot and robot controller

Procedure

- Select the menu sequence **Help** > **Info**.

Description

The information is required, for example, when requesting help from KUKA Customer Support.

The tabs contain the following information:

Tab	Description
Info	- Robot controller type - Robot controller version - User interface version - Kernel system version
Robot	- Robot name - Robot type and configuration - Service life The operating hours meter is running as long as the drives are switched on. Alternatively, the operating hours can also be displayed via the variable $ROB-RUNTIME. - Number of axes - List of external axes - Machine data version
System	- Control PC name - Operating system versions and BIOS version - Storage capacities
Options	Additionally installed options and technology packages
Comments	Additional comments
Modules	Names and versions of important system files The **Save** softkey exports the contents of the **Modules** tab to the file C:\KRC\ROBOTER\LOG\OCXVER.TXT.
Virus scanner	Names and versions of installed virus scanner files The **Export** softkey exports the contents of the **Virus scanner** tab to the file C:\KRC\ROBOTER\LOG\VIRUS-INFO.XML.

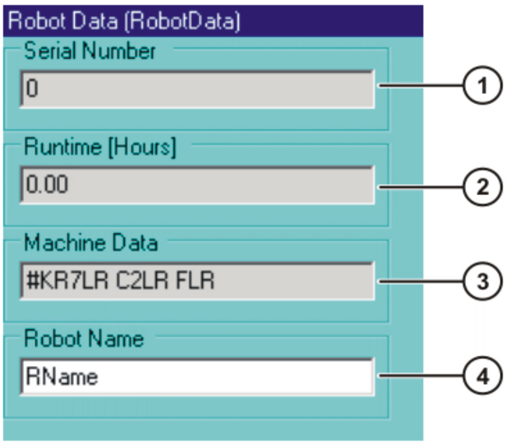
4.20.10 Displaying robot data

Procedure

- Select the menu sequence **Setup** > **Robot data**.

Description

Robot serial number on hard drive correct


Robot Data (RobotData)

Serial Number: 0

Runtime [Hours]: 0.00

Machine Data: #KR7LR C2LR FLR

Robot Name: RName

Fig. 4-37: Robot data – serial number on hard drive correct

Item	Description
1	Serial number. The serial number can be modified in the user group “Expert”.
2	Operating hours. The operating hours meter is running as long as the drives are switched on. Alternatively, the operating hours can also be displayed via the variable $ROBRUNTIME.
3	Machine data name
4	Robot name. The robot name can be changed.

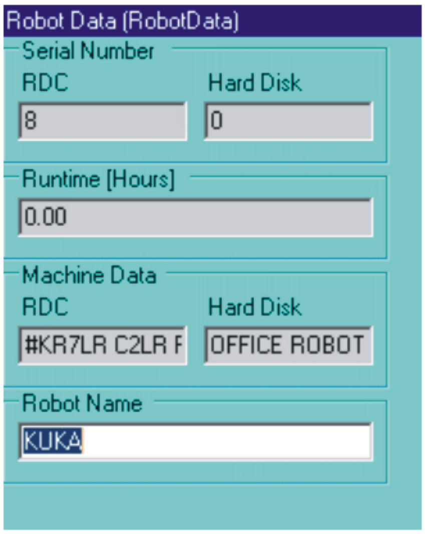
The following softkey is available in the user group Expert:

Softkey	Description
Import PID	No function

Robot serial number on hard drive not correct

During initial start-up and if the hard drive is exchanged, the robot serial number displayed in the **RDC** box and the serial number on the hard drive are no longer identical. (This is also the case, for example, if not just the hard drive, but the entire robot controller has been exchanged.)

When the KSS is started, this is indicated by a message: The **Robot data** window indicates the different serial numbers.



Robot Data (RobotData)

Serial Number

RDC: 8

Hard Disk: 0

Runtime [Hours]: 0.00

Machine Data

RDC: #KR7LR C2LR F

Hard Disk: OFFICE ROBOT

Robot Name: KUKA

Fig. 4-38: Robot data – serial number on hard drive not correct

The following softkeys are available in the user group Expert:

Softkey	Description
Hard disk	No function
RDC	Transfers the following data to the hard drive: ■ Serial number ■ Operating hours The name of the machine data is not transferred, as it otherwise no longer matches the machine data on the hard drive ($MACHINE.DAT, $ROBCOR.DAT).

4.20.11 Displaying hardware information

Procedure

1. Select the menu sequence **Monitor** > **Hardware Info**.
2. If required, open up the tree structure in the left-hand section of the window and select the desired hardware component.

   Information about the selected component is displayed in the right-hand section of the window.

Description

The following softkeys are available:

Softkey	Description
Load config	Loads the last saved configuration.
Refresh	Refreshes the display.
Export	Exports the hardware information as an XML file.

5 Start-up and recommissioning

5.1 Start-up overview

The lightweight robot cannot be started up until the industrial robot has been correctly installed and connected.
Further information about this can be found in the operating or assembly instructions of the robot and in the operating or assembly instructions of the robot controller.

Step	Description
1	Configure the inputs/outputs between the robot controller and the periphery in the file IOSYS.INI. Detailed information can be found in the field bus documentation.
2	Check the machine data. (>>> 5.2 "Checking the machine data" Page 85)
3	Transfer serial number to hard drive. (>>> 5.3 "Transferring the serial number to the hard drive" Page 86)
4	Master the robot. (>>> 5.4 "Mastering" Page 87)
5	Tool calibration completed. In the case of a fixed tool: calibrate external TCP. (>>> 5.5.2 "Tool calibration" Page 92) (>>> 5.5.3 "Fixed tool calibration" Page 99)
6	Enter load data. (>>> 5.6 "Load data" Page 107)
7	Calibrate the base. In the case of a fixed tool: calibrate workpiece. (>>> 5.5.4 "Base calibration" Page 104) (>>> 5.5.3 "Fixed tool calibration" Page 99)
8	If the robot is to be controlled from a higher-level controller, e.g. a PLC: configure Automatic External interface. (>>> 6.23 "Configuring Automatic External" Page 173)

Long text names of inputs/outputs, flags, etc., can be saved in a text file and imported after a reinstallation. In this way, the long texts do not need to be re-entered manually for each manipulator. Furthermore, the long text names can be updated in application programs.
(>>> 5.7 "Transferring long text names" Page 108)

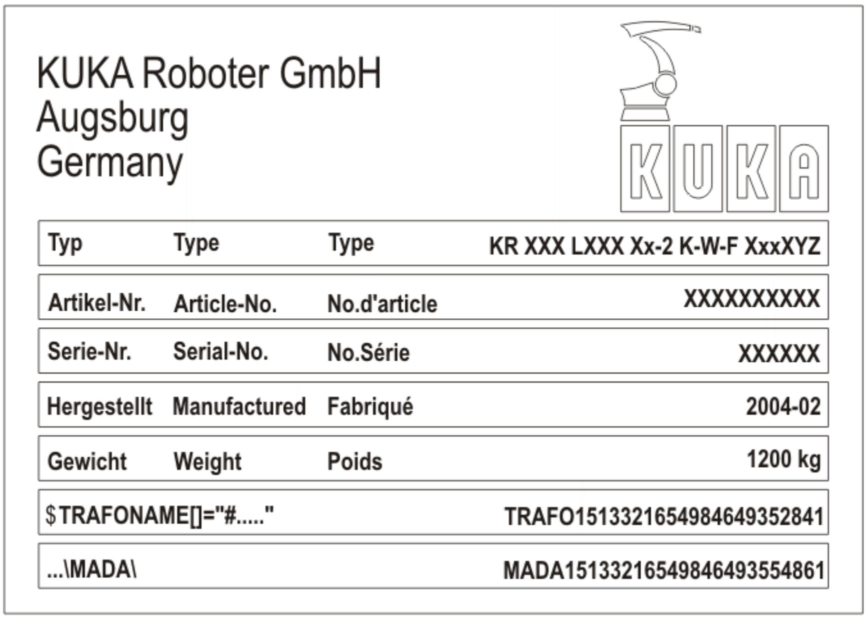
5.2 Checking the machine data

Description

The correct machine data must be loaded. This must be checked by comparing the loaded machine data with the machine data on the rating plate.

If machine data are reloaded, the version of the machine data must correspond exactly to the KSS version. This is ensured if the machine data supplied together with the KSS release are used.

DANGER The industrial robot must not be moved if incorrect machine data are loaded. Death, severe physical injuries or considerable damage to property may otherwise result. The correct machine data must be loaded.



KUKA Roboter GmbH
Augsburg
Germany

Typ	Type	Type	KR XXX LXXX Xx-2 K-W-F XxxXYZ
Artikel-Nr.	Article-No.	No.d'article	XXXXXXXXXX
Serie-Nr.	Serial-No.	No.Série	XXXXXX
Hergestellt	Manufactured	Fabriqué	2004-02
Gewicht	Weight	Poids	1200 kg
$TRAFONAME[]="#....."			TRAFO1513321654984649352841
...\MADA\			MADA15133216549846493554861

Fig. 5-1: Rating plate

Procedure

1. Select the menu sequence **Setup** > **Robot data**.

   The **Robot Data** window is opened.

2. Compare the following entries:
   - In the **Robot data** window: the entry in the **Machine data** box
   - On the rating plate on the base of the robot: the entry in the line **$TRAFONAME()="# ..... "**

The file path of the machine data on the CD is specified on the rating plate in the line **...\MADA**.

5.3 Transferring the serial number to the hard drive

Description

During initial start-up and if the hard drive is exchanged, the robot serial number displayed in the **RDC** box and the serial number on the hard drive are no longer identical. (This is also the case, for example, if not just the hard drive, but the entire robot controller has been exchanged.)

When the KSS is started, this is indicated by the following message: "*Data of RDC and Hard Disk inconsistent! Check robot data!*". The data must now be transferred to the hard drive.

Precondition

- "Expert" user group

Procedure

1. Select the menu sequence **Setup** > **Robot data**.
2. Press the **RDC** softkey. Confirm the request for confirmation with **OK**.

   The following data are transferred:

   - Serial number
   - Operating hours

The name of the machine data is not transferred, as it otherwise no longer matches the machine data on the hard drive ($MACHINE.DAT, $ROBCOR.DAT).

5.4 Mastering

Overview

Step	Description
1	Move axes to the mastering position. (>>> 5.4.2 "Moving axes to the mastering position" Page 87)
2	Carry out position mastering. (>>> 5.4.3 "Carrying out position mastering" Page 90)
3	Carry out torque mastering. (>>> 5.4.4 "Carrying out torque mastering" Page 90)
4	Enter gravitation vector. (>>> 5.4.5 "Gravitation vector" Page 91)

5.4.1 Displaying softkeys for mastering

Precondition

- "Expert" user group

Procedure

1. Display status keys for the lightweight robot.
   (>>> 4.15 "Displaying status keys for the lightweight robot" Page 66)
2. Press the **Set-up** status key in the left-hand status key bar.
   The softkeys for mastering are displayed.

Description

The following softkeys are available:

Softkey	Description
AllTorqZero	Resets the torque sensors in all axes Precondition: All axes are in the torque mastering position.
AllPosZero	Position mastering of all axes Precondition: All axes are in the position mastering position.
Temperature	Displays the temperature data for all axes
TorqZero	Resets the torque sensors of individual axes Precondition: Axis is in the torque mastering position.
PosZero	Position mastering of individual axes Precondition: Axis is in the position mastering position.

5.4.2 Moving axes to the mastering position

Precondition

- Operating mode T1

Procedure

- Using the jog keys, jog the axes in axis-specific mode and move them to the mastering position.

Description

The mastering position varies according to the installation type and according to whether position mastering or torque mastering is carried out.

- For position mastering, the robot must always be moved to the extended position.

  Axes A1 to A4, A6 and E1 are aligned exactly in the zero position by means of the vernier.

  Axis A5 is moved so that the mastering marks line up.

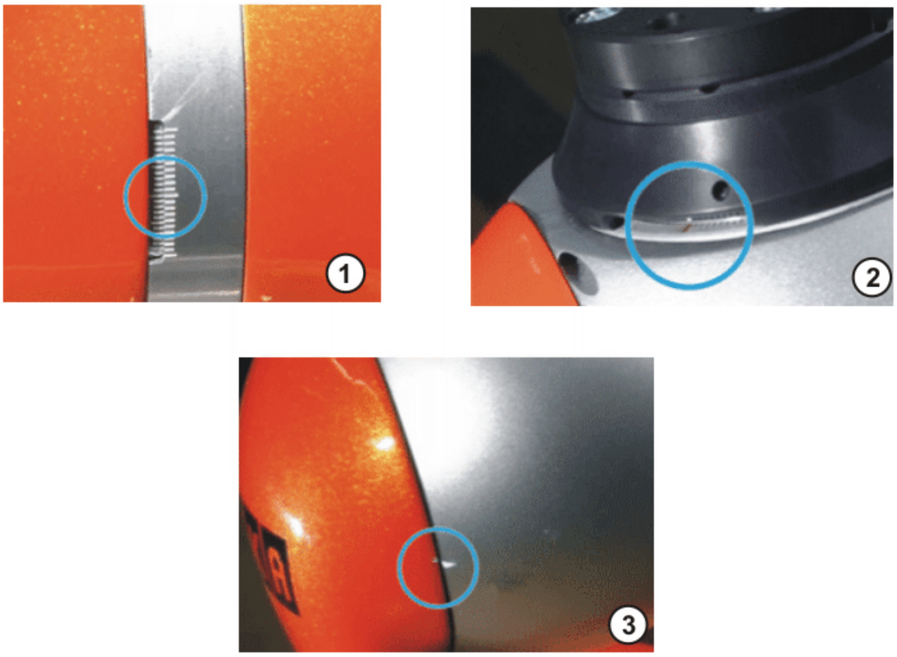


Fig. 5-2: Vernier scales and mastering marks for position mastering

1	Vernier of axes A1 to A4, E1	3	Mastering mark on axis A5
2	Vernier of axis A6		

There are 2 vernier scales opposite one another on axis A6. The vernier that is to be aligned with the zero position is the one near joint A5.

- For torque mastering, the robot must be moved to the position in which the axis-specific torques are virtually zero. This is the case when the axes are aligned in the direction of gravitation or against gravitation.

Floor/ceiling-mounted

In the case of floor and ceiling-mounted robots, the position for position mastering is identical to the position for torque mastering.

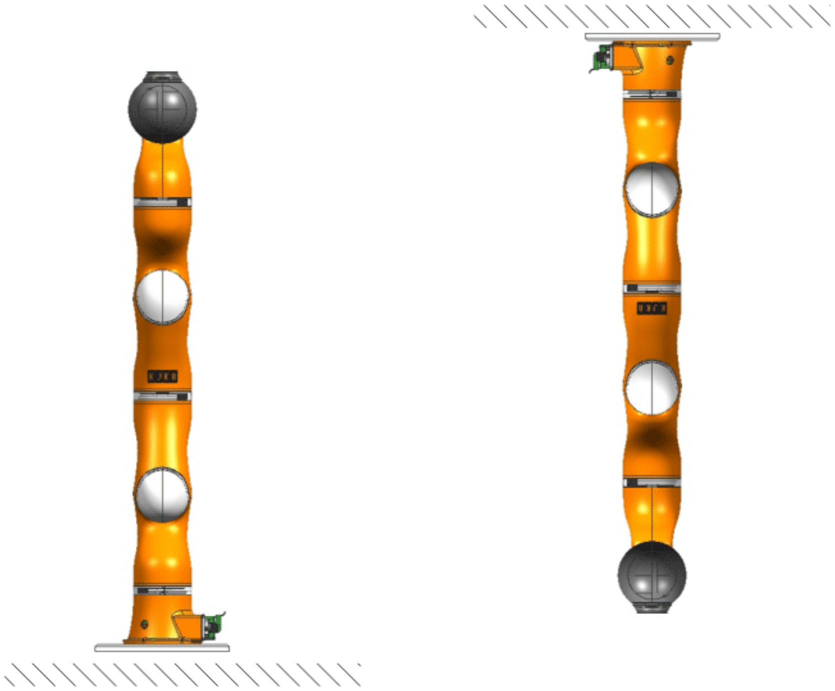


Fig. 5-3: Mastering position for floor- and ceiling-mounted robots

Wall-mounted

In the case of wall-mounted robots, the positions for position mastering and torque mastering differ.

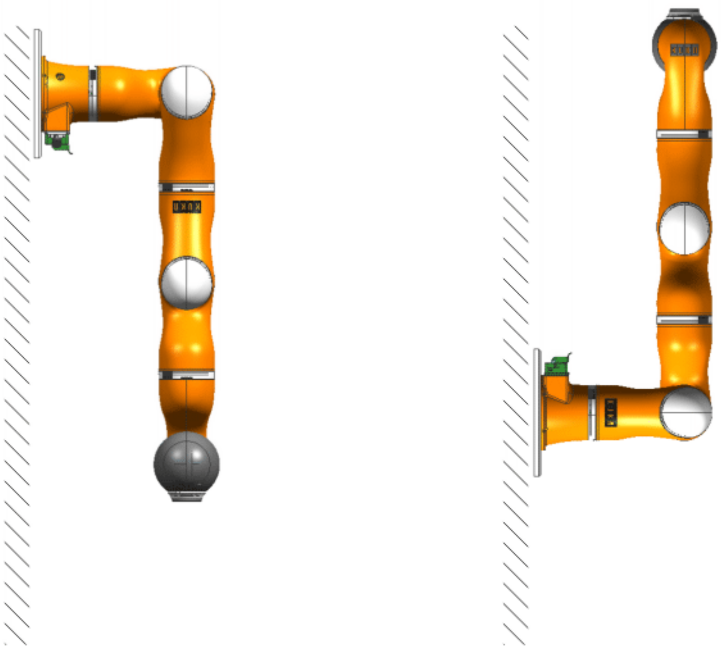


Fig. 5-4: Torque mastering position for wall-mounted robots

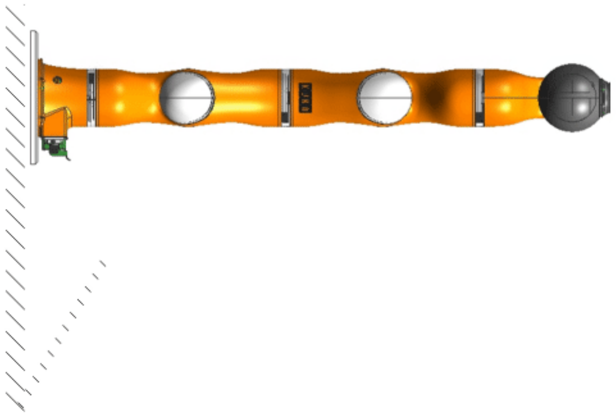


Fig. 5-5: Position mastering position for wall-mounted robots

5.4.3 Carrying out position mastering

Precondition

- User group “Expert”
- Robot is in the position mastering position.

Procedure

Mastering all axes simultaneously:

1. Display softkeys for mastering.
   (>>> 5.4.1 "Displaying softkeys for mastering" Page 87)
2. Press the **AllPosZero** softkey.
3. Press the softkey **SetZeroPosAll**.

Mastering axes individually:

> This method is used if not all axes can be moved simultaneously to the mastering position, e.g. due to lack of space.

1. Display softkeys for mastering.
   (>>> 5.4.1 "Displaying softkeys for mastering" Page 87)
2. Press the **PosZero** softkey.
3. Select the axis by means of the softkey, e.g. **A1J1**.

> The axes can be mastered in any order.

4. Press the **SetZeroPos** softkey.

5.4.4 Carrying out torque mastering

Precondition

- User group “Expert”
- Robot is in the torque mastering position.

Procedure

Mastering all axes simultaneously:

1. Display softkeys for mastering.
   (>>> 5.4.1 "Displaying softkeys for mastering" Page 87)
2. Press the **AllTorqZero** softkey.
3. Press the **SetZeroTorqAll** softkey.

4. Display the axis-specific torques by means of the system variable $TORQUE_AXIS_ACT.

   (>>> 4.20.5 "Displaying and modifying the value of a variable" Page 78)
5. The axis-specific torques must lie within a tolerance range of ±0.1.

   If the mastering result is not within this tolerance, repeat steps 1 to 3.

Mastering axes individually:

> This method is used if not all axes can be moved simultaneously to the mastering position, e.g. due to lack of space.

1. Display softkeys for mastering.

   (>>> 5.4.1 "Displaying softkeys for mastering" Page 87)
2. Press the **TorqZero** softkey.
3. Select the axis by means of the softkey, e.g. **A1J1**.

> The axes can be mastered in any order.

4. Press the **SetZeroTorq** softkey.
5. Display the axis-specific torques by means of the system variable $TORQUE_AXIS_ACT.

   (>>> 4.20.5 "Displaying and modifying the value of a variable" Page 78)
6. The axis-specific torques must lie within a tolerance range of ±0.1.

   If the mastering result is not within this tolerance, repeat steps 1 to 4.

5.4.5 Gravitation vector

Description

The gravitation vector is relative to the WORLD coordinate system and compensates for gravitational acceleration. The gravitation vector is defined by means of the system variable $GRAVITATION[] and stored in $CUSTOM.DAT.

Default setting for floor-mounted robots:

```
REAL $Gravitation[3]
$Gravitation[1]=0.0 ;Gravitation in X direction
$Gravitation[2]=0.0 ;Gravitation in Y direction
$Gravitation[3]=9.81000042 ;Gravitation in Z direction
```

In the case of other configurations, e.g. wall- or ceiling-mounted, the system variable $ROBROOT must be modified in the machine data during initial start-up so that the gravitation vector refers once again to the WORLD coordinate system.

> Further information about the variable $ROBROOT can be found in the machine data documentation.

In the case of subsequent modifications, the gravitation vector can be adapted by means of the system variable $GRAVITATION[], using the variable correction function.

Procedure

1. Select the menu sequence **Monitor** > **Variable** > **Single**.
2. Set the following values:
   - $GRAVITATION[1]: Gravitation in the X direction
   - $GRAVITATION[2]: Gravitation in the Y direction
   - $GRAVITATION[3]: Gravitation in the Z direction

Example Gravitation vector for ceiling-mounted robots:

```
REAL $Gravitation[3]
$Gravitation[1]=0.0 ;;Gravitation in the X direction
$Gravitation[2]=0.0 ;Gravitation in the Y direction
$Gravitation[3]=-9.81000042 ;Gravitation in Z direction
```

5.5 Calibration

5.5.1 Defining the tool direction

Description By default, the X axis is defined in the system as the tool direction. The tool direction can be changed using the system variable $TOOL_DIRECTION.

- The change relates only to spline motions. For LIN and CIRC motions, the tool direction is the X axis and cannot be changed.
- The change applies to all tools. It is not possible to define different tool directions for different tools.

> **⚠ WARNING** The tool direction must be defined before calibration and before program creation. It cannot be modified subsequently. Failure to observe this may result in unexpected changes to the motion characteristics of the robot. Death to persons, severe physical injuries or considerable damage to property may result.

Precondition

- “Expert” user group

Procedure

- Set the system variable $TOOL_DIRECTION to the desired value in the file $CUSTOM.DAT, located in the directory KRC\Steu\MaDa.

  Possible values: #X (default); #Y; #Z

It is not possible to modify $TOOL_DIRECTION by means of the variable correction function or by writing to the variable from the program.

5.5.2 Tool calibration

Description During tool calibration, the user assigns a Cartesian coordinate system (TOOL coordinate system) to the tool mounted on the mounting flange.

The TOOL coordinate system has its origin at a user-defined point. This is called the TCP (Tool Center Point). The TCP is generally situated at the working point of the tool.

> ℹ In the case of a fixed tool, the type of calibration described here must not be used. A separate type of calibration must be used for fixed tools. (>>> 5.5.3 "Fixed tool calibration" Page 99)

Advantages of tool calibration:

- The tool can be moved in a straight line in the tool direction.
- The tool can be rotated about the TCP without changing the position of the TCP.
- In program mode: The programmed velocity is maintained at the TCP along the path.

A maximum of 16 TOOL coordinate systems can be saved. Variable: TOOL_DATA[1…16].

The following data are saved:

- X, Y, Z:
  Origin of the TOOL coordinate system relative to the FLANGE coordinate system
- A, B, C:
  Orientation of the TOOL coordinate system relative to the FLANGE coordinate system

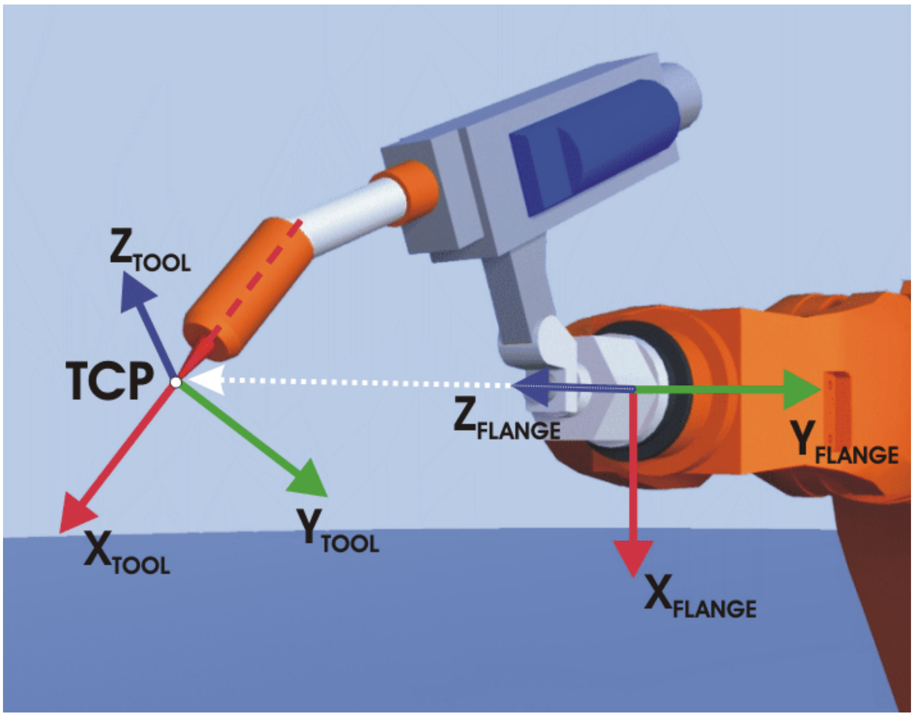


Fig. 5-6: TCP calibration principle

Overview

Tool calibration consists of 2 steps:

Step	Description
1	**Definition of the origin of the TOOL coordinate system** The following methods are available: - XYZ 4-point (>>> 5.5.2.1 "TCP calibration: XYZ 4-point method" Page 94) - XYZ Reference (>>> 5.5.2.2 "TCP calibration: XYZ Reference method" Page 95)
2	**Definition of the orientation of the TOOL coordinate system** The following methods are available: - ABC 2-point (>>> 5.5.2.3 "Defining the orientation: ABC 2-point method" Page 96) - ABC World (>>> 5.5.2.4 "Defining the orientation: ABC World method" Page 98)

If the calibration data are already known, they can be entered directly. (>>> 5.5.2.5 "Entering the tool numerically" Page 98)

5.5.2.1 TCP calibration: XYZ 4-point method

ℹ The XYZ 4-point method cannot be used for palletizing robots.

Description

The TCP of the tool to be calibrated is moved to a reference point from 4 different directions. The reference point can be freely selected. The robot controller calculates the TCP from the different flange positions.

ℹ The 4 flange positions at the reference point must be sufficiently different from one another.

Precondition

- The tool to be calibrated is mounted on the mounting flange.
- Operating mode T1 or T2

Procedure

1. Select the menu **Setup** > **Measure** > **Tool** > **XYZ 4-Point**.
2. Assign a number and a name for the tool to be calibrated. Confirm with **Continue**.
3. Move the TCP to a reference point. Press **Calibrate**. Confirm with **Next**.

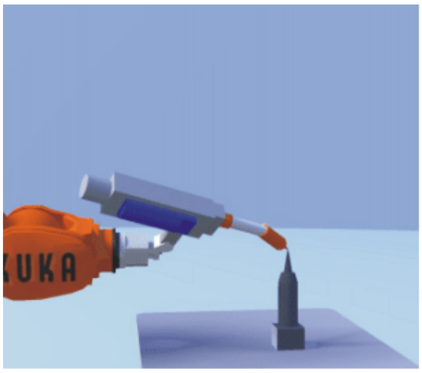


Fig. 5-7: Move to reference point

4. Move the TCP to the reference point from a different direction. Press **Calibrate**. Confirm with **Next**.

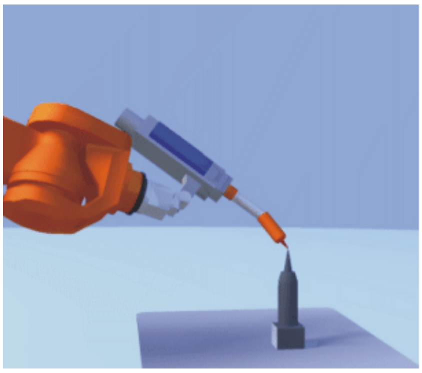


Fig. 5-8: Move to point from different direction

5. Repeat step 4 twice.

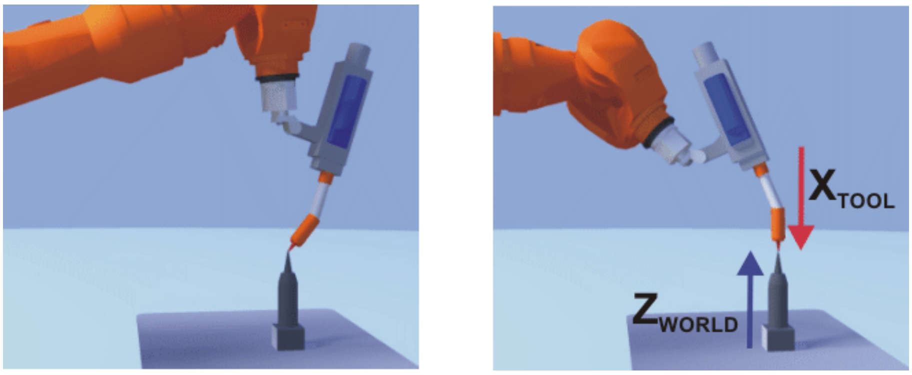


Fig. 5-9: Move to point from 3rd and 4th direction

6. Either press **Save**. The data are saved and the window is closed.

   Or press **Load data**. The data are saved and a window is opened in which the payload data can be entered.

   (>>> 5.6.2 "Entering payload data" Page 107)

   Or press **ABC 2-point** or **ABC World**. The data are saved and a window is opened in which the orientation of the TOOL coordinate system can be defined.

   (>>> 5.5.2.3 "Defining the orientation: ABC 2-point method" Page 96)

   (>>> 5.5.2.4 "Defining the orientation: ABC World method" Page 98)

5.5.2.2 TCP calibration: XYZ Reference method

Description

In the case of the XYZ Reference method, a new tool is calibrated with a tool that has already been calibrated. The robot controller compares the flange positions and calculates the TCP of the new tool.

Precondition

- A previously calibrated tool is mounted on the mounting flange.
- Operating mode T1 or T2

Preparation

Calculate the TCP data of the calibrated tool:

1. Select the menu **Setup** > **Measure** > **Tool** > **XYZ Reference**.
2. Enter the number of the calibrated tool.
3. Note the X, Y and Z values.
4. Close the window by pressing **Cancel**.

Procedure

1. Select the menu **Setup** > **Measure** > **Tool** > **XYZ Reference**.
2. Assign a number and a name for the new tool. Confirm with **Continue**.
3. Enter the TCP data of the calibrated tool. Confirm with **Continue**.
4. Move the TCP to a reference point. Press **Measure**. Confirm with **Continue**.

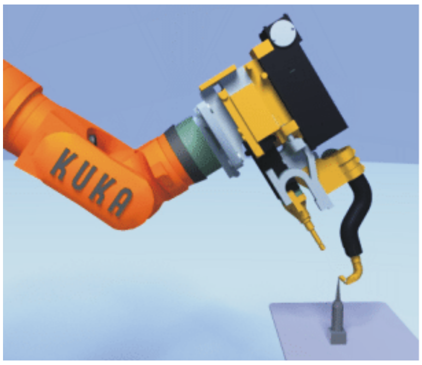


Fig. 5-10: Move to point with calibrated tool

5. Move the tool away and remove it. Mount the new tool.
6. Move the TCP of the new tool to the reference point. Press **Measure**. Confirm with **Continue**.

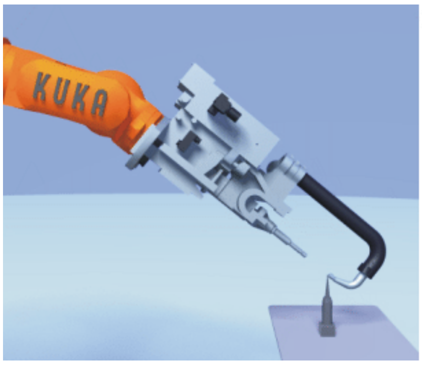


Fig. 5-11: Move to point with new tool

7. Either press **Save**. The data are saved and the window is closed.

   Or press **Load data**. The data are saved and a window is opened in which the payload data can be entered.

   (>>> 5.6.2 "Entering payload data" Page 107)

   Or press **ABC 2-Point** or **ABC World**. The data are saved and a window is opened in which the orientation of the TOOL coordinate system can be defined.

   (>>> 5.5.2.3 "Defining the orientation: ABC 2-point method" Page 96)

   (>>> 5.5.2.4 "Defining the orientation: ABC World method" Page 98)

5.5.2.3 Defining the orientation: ABC 2-point method

Description

The axes of the TOOL coordinate system are communicated to the robot controller by moving to a point on the X axis and a point in the XY plane.

This method is used if it is necessary to define the axis directions with particular precision.

Precondition

- The tool to be calibrated is mounted on the mounting flange.
- The TCP of the tool has already been measured.
- Operating mode T1 or T2

The following procedure applies if the tool direction is the default tool direction (= X axis). If the tool direction has been changed to Y or Z, the procedure must also be changed accordingly. (>>> 5.5.1 "Defining the tool direction" Page 92)

Procedure

1. Select the menu **Setup** > **Measure** > **Tool** > **ABC 2-Point**.

2. Enter the number of the mounted tool. Confirm with **Next**.
3. Move the TCP to any reference point. Press **Calibrate**. Confirm with **Next**.

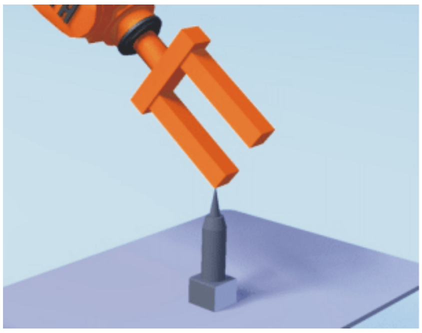


Fig. 5-12: Move to reference point

4. Move the tool so that the reference point on the X axis has a negative X value (i.e. move against the tool direction). Press **Calibrate**. Confirm with **Next**.

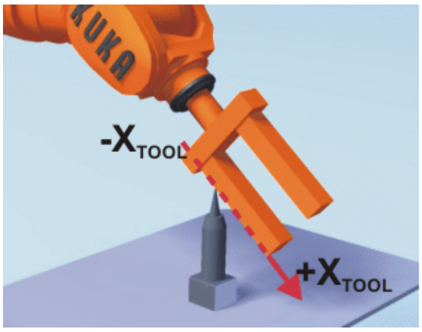


Fig. 5-13: Reference point on the X-axis

5. Move the tool so that the reference point in the XY plane has a negative Y value. Press **Calibrate**. Confirm with **Next**.

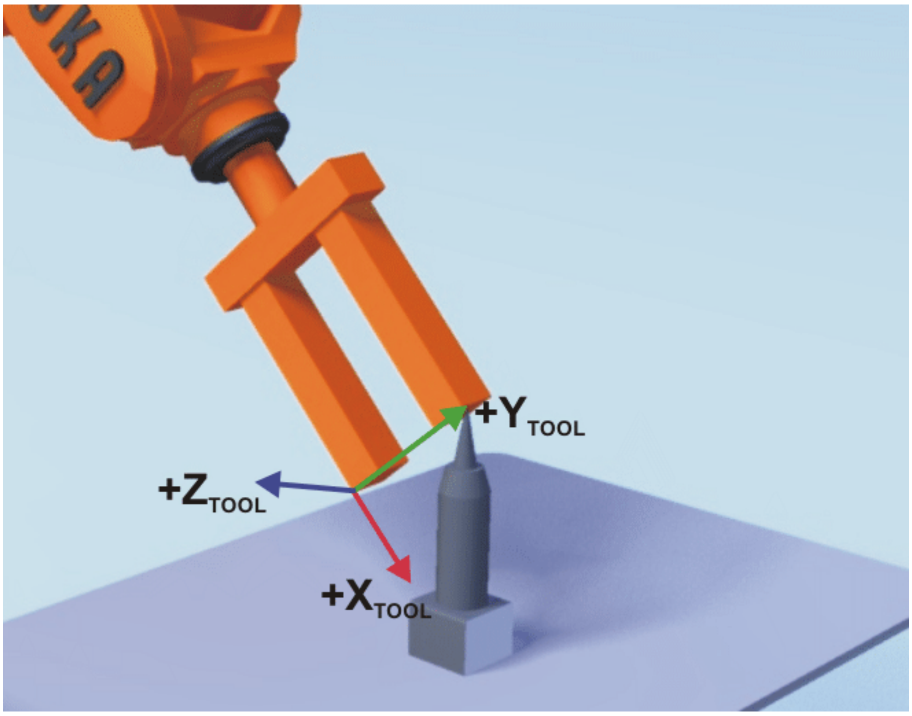


Fig. 5-14: Reference point on XY plane

6. Either press **Save**. The data are saved and the window is closed.

Or press **Load data**. The data are saved and a window is opened in which the payload data can be entered.

(>>> 5.6.2 "Entering payload data" Page 107)

5.5.2.4 Defining the orientation: ABC World method

Description

The axes of the TOOL coordinate system are aligned parallel to the axes of the WORLD coordinate system. This communicates the orientation of the TOOL coordinate system to the robot controller.

There are 2 variants of this method:

- **5D**: Only the tool direction is communicated to the robot controller. By default, the tool direction is the X axis. The directions of the other axes are defined by the system and cannot be detected easily by the user.

  Area of application: e.g. MIG/MAG welding, laser cutting or waterjet cutting
- **6D**: The directions of all 3 axes are communicated to the robot controller.

  Area of application: e.g. for weld guns, grippers or adhesive nozzles

If **6D** is selected: it is advisable to document the alignment of all axes. If the tool subsequently has to be calibrated again, e.g. after a crash, the axes must be aligned the same way as the first time in order to be able to continue moving to existing points correctly.

Precondition

- The tool to be calibrated is mounted on the mounting flange.
- The TCP of the tool has already been measured.
- Operating mode T1 or T2

The following procedure applies if the tool direction is the default tool direction (= X axis). If the tool direction has been changed to Y or Z, the procedure must also be changed accordingly. (>>> 5.5.1 "Defining the tool direction" Page 92)

Procedure

1. Select the menu **Setup** > **Measure** > **Tool** > **ABC World**.
2. Enter the number of the tool. Confirm with **Next**.
3. Select a variant in the box **5D/6D**. Confirm with **Next**.
4. If **5D** is selected:

   Align +X_{TOOL} parallel to -Z_{WORLD}. (+X_{TOOL} = tool direction)

   If **6D** is selected:

   Align the axes of the TOOL coordinate system as follows.

   - +X_{TOOL} parallel to -Z_{WORLD}. (+X_{TOOL} = tool direction)
   - +Y_{TOOL} parallel to +Y_{WORLD}
   - +Z_{TOOL} parallel to +X_{WORLD}
5. Press **Calibrate**. Confirm with **Next**.
6. Either press **Save**. The data are saved and the window is closed.

   Or press **Load data**. The data are saved and a window is opened in which the payload data can be entered.

   (>>> 5.6.2 "Entering payload data" Page 107)

5.5.2.5 Entering the tool numerically

Description

The tool data can be entered manually.

Possible sources of data:

- CAD

- Externally calibrated tool
- Tool manufacturer specifications

> In the case of palletizing robots with 4 axes, e.g. KR 180 PA, the tool data must be entered numerically. The XYZ and ABC methods cannot be used as reorientation of these robots is highly restricted.

Precondition

The following values are known:

- X, Y and Z relative to the FLANGE coordinate system
- A, B and C relative to the FLANGE coordinate system

Procedure

1. Select the menu **Setup** > **Measure** > **Tool** > **Numeric Input**.
2. Assign a number and a name for the tool to be calibrated. Confirm with **Next**.
3. Enter data. Confirm with **Next**.
4. Either press **Save**. The data are saved and the window is closed.

   Or press **Load data**. The data are saved and a window is opened in which the payload data can be entered.

   (>>> 5.6.2 "Entering payload data" Page 107)

5.5.3 Fixed tool calibration

Overview

Calibration of a fixed tool consists of 2 steps:

Step	Description
1	**Calibration of the TCP of the fixed tool** The TCP of a fixed tool is called an external TCP. (>>> 5.5.3.1 "Calibrating an external TCP" Page 99) If the calibration data are already known, they can be entered directly. (>>> 5.5.3.2 "Entering the external TCP numerically" Page 101)
2	**Calibration of the workpiece** The following methods are available: ■ Direct method (>>> 5.5.3.3 "Workpiece calibration: direct method" Page 101) ■ Indirect method (>>> 5.5.3.4 "Workpiece calibration: indirect method" Page 103)

The robot controller saves the external TCP as the BASE coordinate system and the workpiece as the TOOL coordinate system. A maximum of 32 BASE coordinate systems and 16 TOOL coordinate systems can be saved.

5.5.3.1 Calibrating an external TCP

Description

First of all, the TCP of the fixed tool is communicated to the robot controller. This is done by moving a calibrated tool to it.

Then, the orientation of the coordinate system of the fixed tool is communicated to the robot controller. For this purpose, the coordinate system of the calibrated tool is aligned parallel to the new coordinate system. There are 2 variants:

- **5D**: Only the tool direction of the fixed tool is communicated to the robot controller. By default, the tool direction is the X axis. The orientation of the other axes is defined by the system and cannot be detected easily by the user.
- **6D**: The orientation of all 3 axes is communicated to the robot controller.

> If **6D** is selected: it is advisable to document the alignment of all axes. If the tool subsequently has to be calibrated again, e.g. after a crash, the axes must be aligned the same way as the first time in order to be able to continue moving to existing points correctly.

Precondition

- A previously calibrated tool is mounted on the mounting flange.
- Operating mode T1 or T2

> The following procedure applies if the tool direction is the default tool direction (= X axis). If the tool direction has been changed to Y or Z, the procedure must also be changed accordingly. (>>> 5.5.1 "Defining the tool direction" Page 92)

Procedure

1. Select the menu **Setup** > **Measure** > **Fixed tool** > **Tool**.
2. Assign a number and a name for the fixed tool. Confirm with **Next**.
3. Enter the number of the calibrated tool. Confirm with **Next**.
4. Select a variant in the box **5D/6D**. Confirm with **Next**.
5. Move the TCP of the calibrated tool to the TCP of the fixed tool. Press **Calibrate**. Confirm with **Next**.

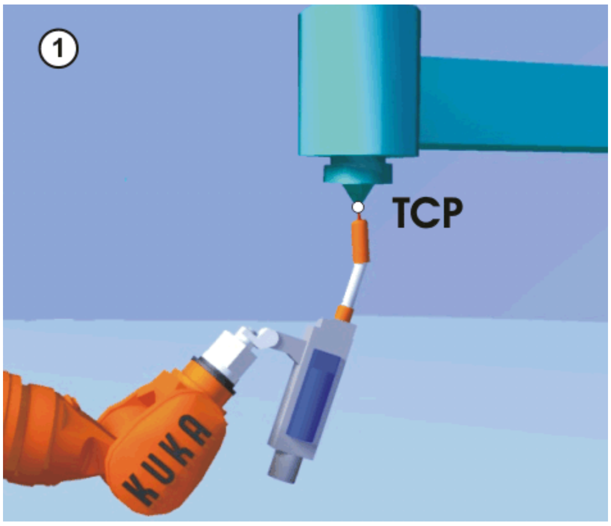


Fig. 5-15: Moving to the external TCP

6. If **5D** is selected:

   Align +X_{BASE} parallel to -Z_{FLANGE}.

   (i.e. align the mounting flange perpendicular to the tool direction of the fixed tool.)

   If **6D** is selected:

   Align the mounting flange so that its axes are parallel to the axes of the fixed tool:

   - +X_{BASE} parallel to -Z_{FLANGE}

     (i.e. align the mounting flange perpendicular to the tool direction.)
   - +Y_{BASE} parallel to +Y_{FLANGE}
   - +Z_{BASE} parallel to +X_{FLANGE}

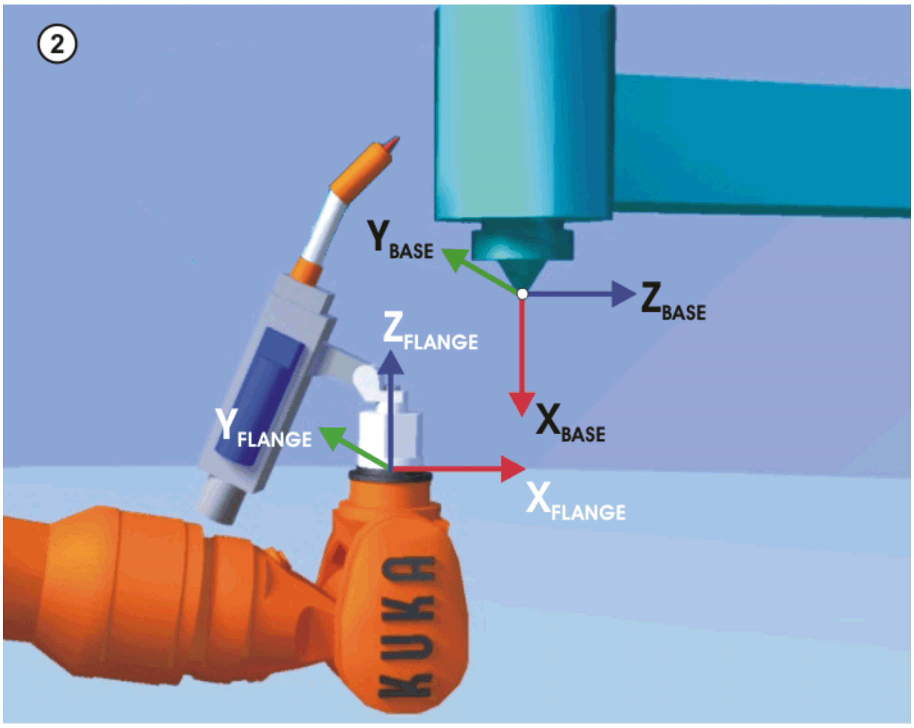


Fig. 5-16: Aligning the coordinate systems parallel to one another

7. Press **Calibrate**. Confirm with **Next**.
8. Press **Save**.

5.5.3.2 Entering the external TCP numerically

Precondition

The following numerical values are known, e.g. from CAD data:

- Distance between the TCP of the fixed tool and the origin of the WORLD coordinate system (X, Y, Z)
- Rotation of the axes of the fixed tool relative to the WORLD coordinate system (A, B, C)

Procedure

1. Select the menu **Setup** > **Measure** > **Fixed tool** > **Numeric Input**.
2. Assign a number and a name for the fixed tool. Confirm with **Next**.
3. Enter data. Confirm with **Next**.
4. Press **Save**.

5.5.3.3 Workpiece calibration: direct method

Description

The origin and 2 further points of the workpiece are communicated to the robot controller. These 3 points uniquely define the workpiece.

Precondition

- The workpiece is mounted on the mounting flange.
- A previously calibrated fixed tool is mounted.
- Operating mode T1 or T2

Procedure

1. Select the menu **Setup** > **Measure** > **Fixed tool** > **Workpiece** > **Direct measuring**.
2. Assign a number and a name for the workpiece. Confirm with **Continue**.
3. Enter the number of the fixed tool. Confirm with **Continue**.
4. Move the origin of the workpiece coordinate system to the TCP of the fixed tool. Press **Measure**. Confirm with **Continue**.

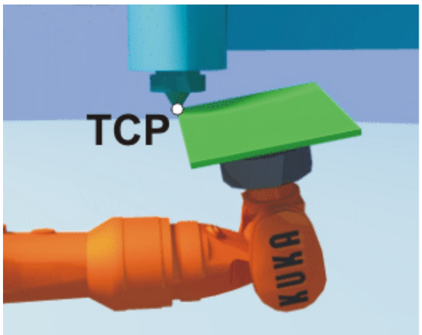


Fig. 5-17: Origin of the workpiece coordinate system

5. Move a point on the positive X axis of the workpiece coordinate system to the TCP of the fixed tool. Press **Measure**. Confirm with **Continue**.

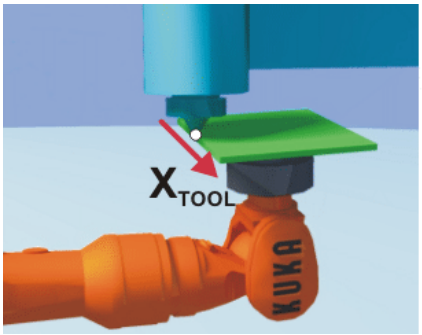


Fig. 5-18: Point on the positive X axis

6. Move a point with a positive Y value in the XY plane of the workpiece coordinate system to the TCP of the fixed tool. Press **Measure**. Confirm with **Continue**.

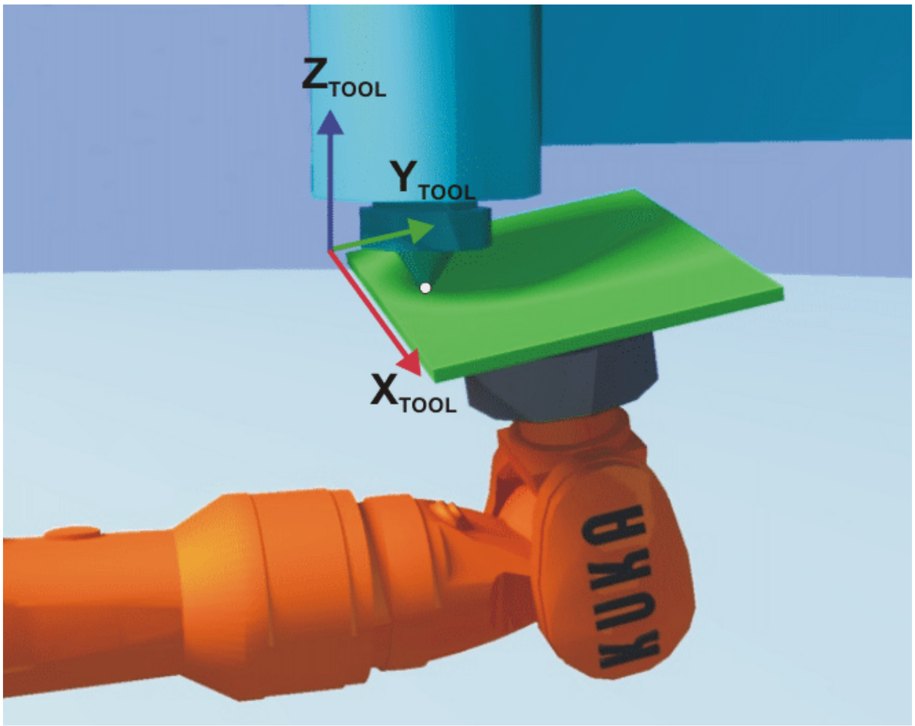


Fig. 5-19: Point on XY plane with a positive Y value

7. Either press **Save**. The data are saved and the window is closed.

   Or press **Load data**. The data are saved and a window is opened in which the payload data can be entered.

(>>> 5.6.2 "Entering payload data" Page 107)

5.5.3.4 Workpiece calibration: indirect method

Description

The robot controller calculates the workpiece on the basis of 4 points whose coordinates must be known. The robot does not move to the origin of the workpiece.

Precondition

- A previously calibrated fixed tool is mounted.
- The workpiece to be calibrated is mounted on the mounting flange.
- The coordinates of 4 points of the new workpiece are known, e.g. from CAD data. The 4 points are accessible to the TCP.
- Operating mode T1 or T2

Procedure

1. Select the menu **Setup** > **Measure** > **Fixed tool** > **Workpiece** > **Indirect measuring**.
2. Assign a number and a name for the workpiece. Confirm with **Continue**.
3. Enter the number of the fixed tool. Confirm with **Continue**.
4. Enter the coordinates of a known point on the workpiece and move this point to the TCP of the fixed tool. Press **Measure**. Confirm with **Continue**.

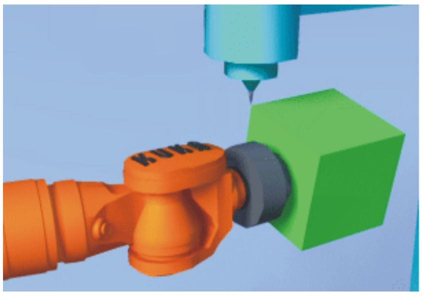


Fig. 5-20: Moving to a known point on the TCP

5. Repeat step 4 three times.

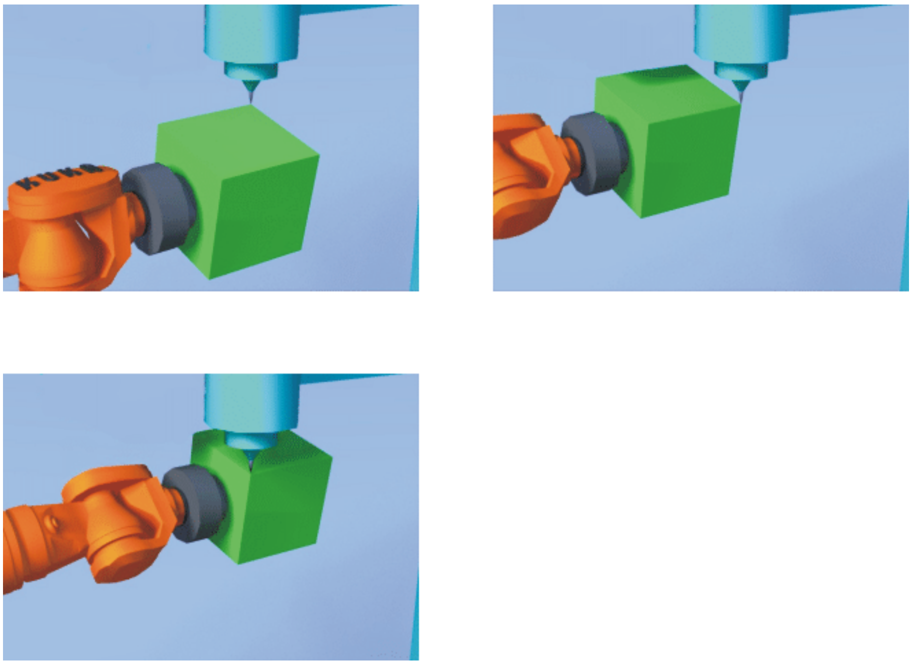


Fig. 5-21: Repeat three times

6. Either press **Save**. The data are saved and the window is closed.

Or press **Load data**. The data are saved and a window is opened in which the payload data can be entered.

(>>> 5.6.2 "Entering payload data" Page 107)

5.5.4 Base calibration

Description

During base calibration, the user assigns a Cartesian coordinate system (BASE coordinate system) to a work surface or the workpiece. The BASE coordinate system has its origin at a user-defined point.

> ℹ If the workpiece is mounted on the mounting flange, the type of calibration described here must not be used. A separate type of calibration must be used for workpieces mounted on the mounting flange. (>>> 5.5.3 "Fixed tool calibration" Page 99)

Advantages of base calibration:

- The TCP can be jogged along the edges of the work surface or workpiece.
- Points can be taught relative to the base. If it is necessary to offset the base, e.g. because the work surface has been offset, the points move with it and do not need to be retaught.

A maximum of 32 BASE coordinate systems can be saved. Variable: BASE_DATA[1…32].

Overview

There are 2 ways of calibrating a base:

- 3-point method (>>> 5.5.4.1 "3-point method" Page 104)
- Indirect method (>>> 5.5.4.2 "Indirect method" Page 105)

If the calibration data are already known, they can be entered directly. (>>> 5.5.4.3 "Entering the base numerically" Page 106)

5.5.4.1 3-point method

Description

The robot moves to the origin and 2 further points of the new base. These 3 points define the new base.

Precondition

- A previously calibrated tool is mounted on the mounting flange.
- Operating mode T1 or T2

Procedure

1. Select the menu **Setup** > **Measure** > **Base** > **ABC 3-Point**.
2. Assign a number and a name for the base. Confirm with **Next**.
3. Enter the number of the mounted tool. Confirm with **Next**.
4. Move the TCP to the origin of the new base. Press **Measure**. Confirm with **Next**.

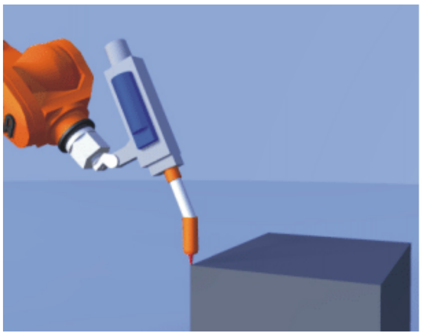


Fig. 5-22: Origin of the base coordinate system

5. Move the TCP to a point on the positive X axis of the new base. Press **Measure**. Confirm with **Next**.

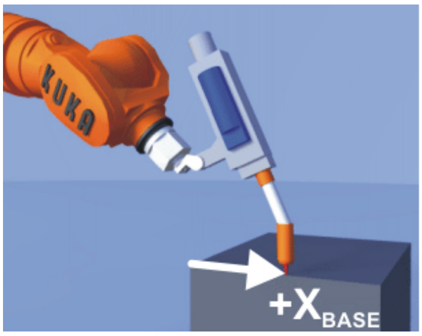


Fig. 5-23: Point on the positive X axis

6. Move the TCP to a point in the XY plane with a positive Y value. Press **Measure**. Confirm with **Next**.

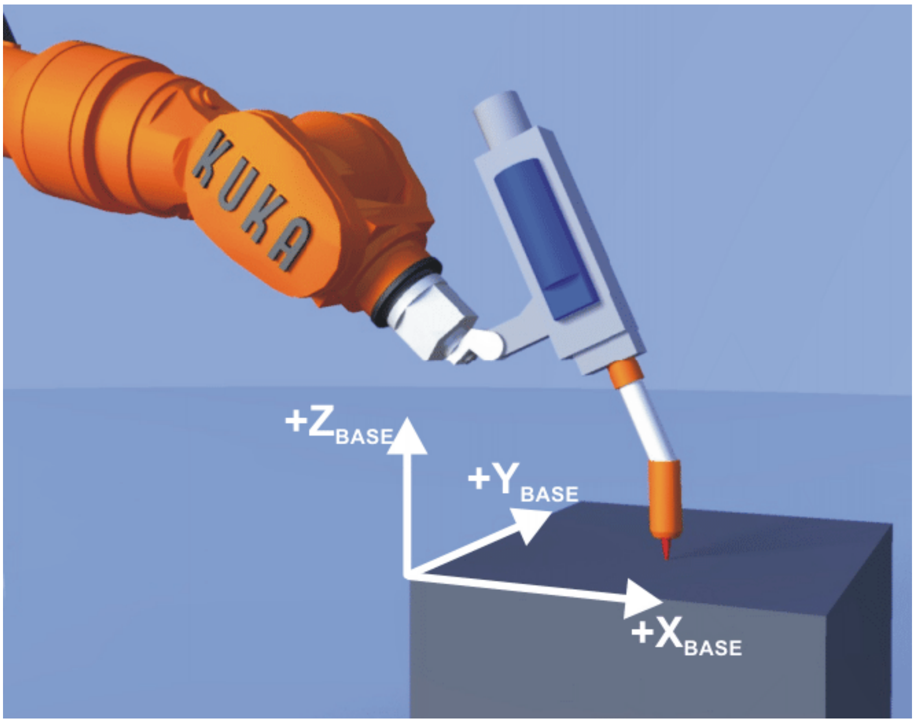


Fig. 5-24: Point on XY plane with a positive Y value

7. Press **Save**.

5.5.4.2 Indirect method

Description

The indirect method is used if it is not possible to move to the origin of the base, e.g. because it is inside a workpiece or outside the workspace of the robot.

The TCP is moved to 4 points in the base, the coordinates of which must be known. The robot controller calculates the base from these points.

Precondition

- A calibrated tool is mounted on the mounting flange.
- The coordinates of 4 points in the new base are known, e.g. from CAD data. The 4 points are accessible to the TCP.
- Operating mode T1 or T2

Procedure

1. Select the menu **Setup** > **Measure** > **Base** > **Indirect**.
2. Assign a number and a name for the base. Confirm with **Next**.

3. Enter the number of the mounted tool. Confirm with **Next**.
4. Enter the coordinates of a known point in the new base and move the TCP to this point. Confirm with **Next**.

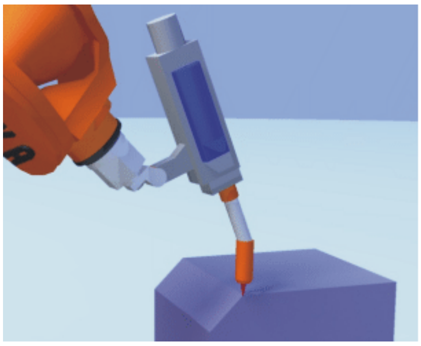


Fig. 5-25: Moving to a known point

5. Repeat step 4 three times.

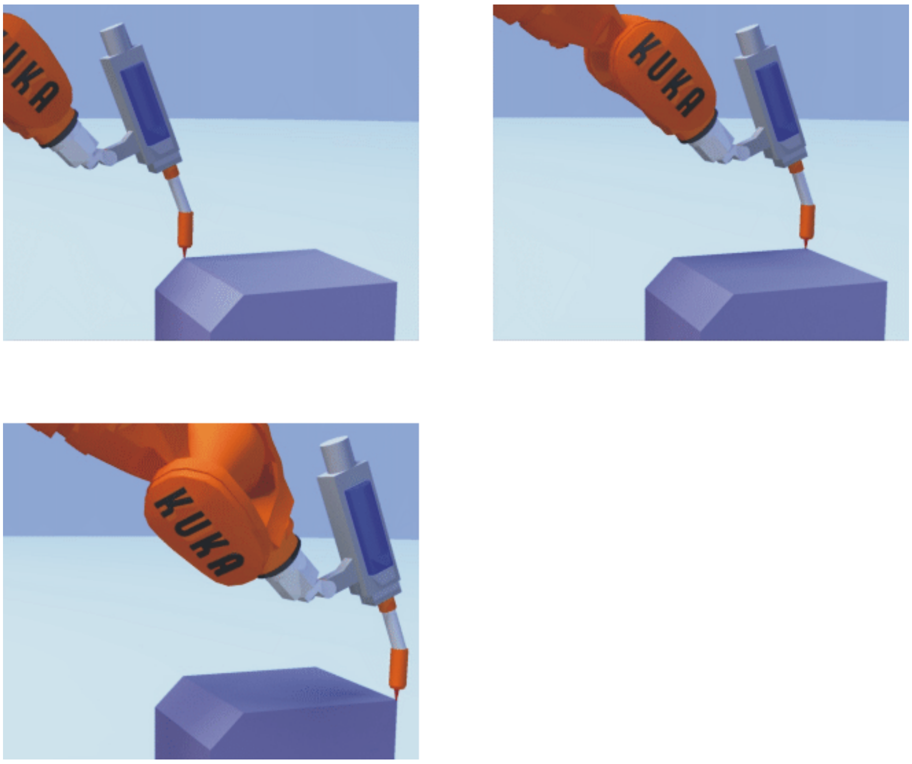


Fig. 5-26: Repeat three times

6. Press **Save**.

5.5.4.3 Entering the base numerically

Precondition

The following numerical values are known, e.g. from CAD data:

- Distance between the origin of the base and the origin of the WORLD coordinate system
- Rotation of the base axes relative to the WORLD coordinate system

Procedure

1. Select the menu **Setup** > **Measure** > **Base** > **Numeric Input**.
2. Assign a number and a name for the base. Confirm with **Next**.
3. Enter data. Confirm with **Next**.
4. Press **Save**.

5.6 Load data

The load data are factored into the calculation of the paths and accelerations and help to optimize the cycle times. The load data must be entered in the robot controller.

⚠ WARNING The robot must not be operated with incorrect load data or unsuitable loads. Failure to observe this precaution may result in severe physical injuries or considerable damage to property.

Sources

Load data can be obtained from the following sources:

- Manufacturer information
- Manual calculation
- CAD programs

5.6.1 Checking loads with KUKA.Load

A payload can be mounted on the flange of the lightweight robot. The payload must be verified with the KUKA.Load software.

A sign-off sheet can be generated for the loads with KUKA.Load. KUKA.Load can be downloaded free of charge, complete with the documentation, from the KUKA website www.kuka.com.

More information is contained in the **KUKA.Load** documentation.

5.6.2 Entering payload data

Description

The payload data must be entered in the robot controller and assigned to the correct tool.

Precondition

- The payload data have been checked with KUKA.Load and the robot is suitable for these payloads.

Procedure

1. Select the menu **Setup** > **Measure** > **Tool** > **Payload data**.
2. Enter the number of the tool in the box **Tool no.**. Confirm with **Continue**.
3. Enter the payload data:
   - Box **M**: Mass
   - Boxes **X**, **Y**, **Z**: Position of the center of gravity relative to the flange
   - Boxes **A**, **B**, **C**: Enter the value zero.
   - Boxes **JX**, **JY**, **JZ**: Mass moments of inertia

     (JX is the inertia about the X axis of the coordinate system that is rotated relative to the flange by A, B and C. JY and JZ are the analogous inertia about the Y and Z axes.)

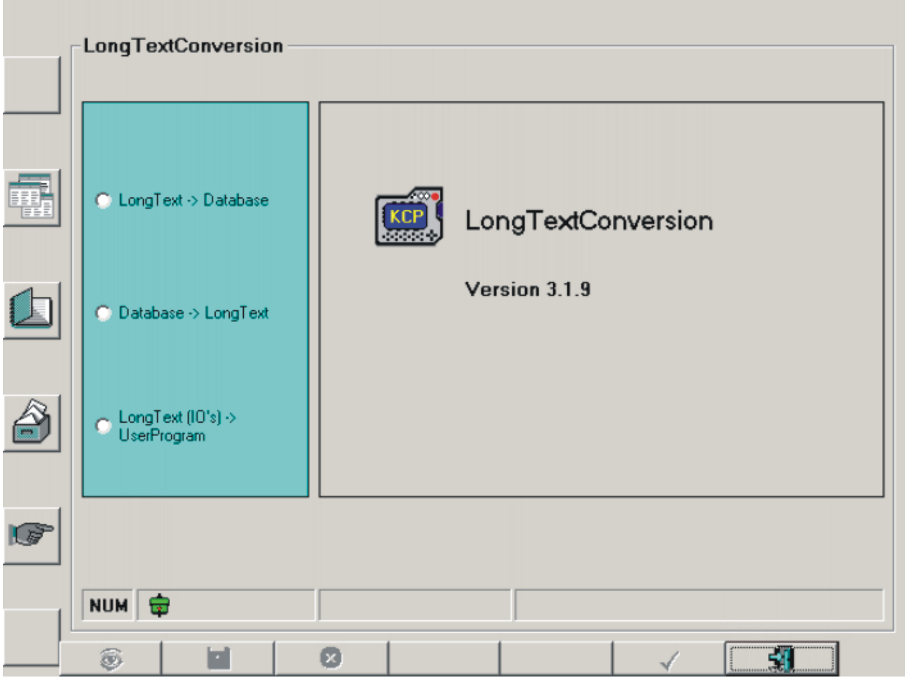
If the mass moments of inertia are unknown or have not been determined accurately, enter the value zero.

4. Confirm with **Continue**.
5. Press **Save**.

5.7 Transferring long text names

This function makes it possible to save the long text names of inputs/outputs, flags, etc., in a text file or to read saved long text names. In this way, the long texts do not need to be re-entered manually for each robot after reinstallation.

Furthermore, the long text names can be updated in application programs.



LongTextConversion

○ LongText -> Database

○ Database -> LongText

○ LongText (IO's) -> UserProgram

LongTextConversion

Version 3.1.9

Fig. 5-27: Long text conversion window

5.7.1 Saving long text names

Precondition

- “Expert” user group

Procedure

1. Select the menu sequence **Setup** > **Service** > **Long text**.
2. Press the status key .
3. Enter the path and name of the text file to be generated.
4. Press the softkey ✓.
5. To close the window, press the softkey.

5.7.2 Reading long text names

Precondition

- “Expert” user group

Procedure

1. Select the menu sequence **Setup** > **Service** > **Long text**.
2. Press the status key .
3. Select the directory containing the text file.
4. If necessary, activate the option **Insert long text**.
5. Press the softkey ✓.

6. To close the window, press the 🚪 softkey.

Description

The ☞ status key activates/deactivates the option **Insert long text**.

- Active: The existing entries in the long text database remain unaffected. New entries will be added.
- Inactive: The existing entries in the long text database are deleted and replaced by new entries.

5.7.3 Updating long text names in programs

Precondition

- “Expert” user group

Procedure

1. Select the menu sequence **Setup** > **Service** > **Long text**.
2. Press the status key 🗃.
3. Select the directory containing the programs to be updated.
4. Either select the individual programs to be updated,

   or activate the option **Select all files**.
5. Press the softkey ✓. The long text names are transferred.
6. To close the window, press the 🚪 softkey.

Description

The ☞ status key activates/deactivates the option **Select all files**.

- Active: All files in the selected directory are updated.
- Inactive: Individual files can be selected for updating in the selected directory.

5.7.4 Editing the long text data base

Precondition

- “Expert” user group

Procedure

1. Select the menu sequence **Setup** > **Service** > **Long text**.
2. Press the softkey 👁. The long text database is displayed.
3. Make the desired changes. The entries must have the specified format.
4. Press the softkey 💾. The changes are saved.

   If you do not wish to save the changes:

   Press the softkey ✖. The database view is closed.

Description

Every entry in the long text database must have the following format:

KeywordNumber LongText

Examples:

- IN_23 Valve
- OUT_215 LH slide
- FlagText5 Alarm

The following elements are permissible:

Keyword	Number	Meaning
IN_	1 … 1024 (… 2048/… 4096)	Digital input
OUT_	1 … 1024 (… 2048/… 4096)	Digital output
NoticeText	1 … 32	Cyclical flag
FlagText	1 … 999	Flag
TimerText	1 … 20	Timer
CounterText	1 … 20	Counter
ANIN_	1 … 32	Analog input
ANOUT_	1 … 32	Analog output

6 Configuration

6.1 Overview of servo controllers

Controller no.	Description
10	Position controller (>>> 6.1.1 "Position controller" Page 111)
20	Cartesian stiffness controller (>>> 6.1.2 "Cartesian stiffness controller" Page 111)
30	Axis-specific stiffness controller (>>> 6.1.3 "Axis-specific stiffness controller" Page 112)
101	Gravitation compensation (>>> 6.1.4 "Gravitation compensation" Page 112)

6.1.1 Position controller

Description

This controller is a joint controller that enables exact positioning. In position control, the robot exhibits no compliance and always moves to the programmed position.

The controller gain settings depend on the current robot position (mass inertia). One precondition for optimal functioning of the controller is that the load data have been entered correctly.

NOTICE Due to the model-based control, unsuitable load data can cause the controller to buzz, e.g. if the distances from the center of gravity are too great. If possible, an error must be documented by means of a test program and passed on to KUKA Roboter GmbH. (>>> 16 "KUKA Service" Page 415)

6.1.2 Cartesian stiffness controller

Description

This controller is a Cartesian impedance controller that works in conjunction with the torque controllers of the joints. If a stiffness controller is used, the robot behaves like a spring with stiffness and damping parameters.

The following parameters are predefined:

Parameter	
Cartesian setpoint position	Motion programming

Parameter	
Cartesian setpoint stiffness	$STIFFNESS (>>> 6.2 "Configuring servo controllers via $STIFFNESS" Page 113)
Cartesian setpoint damping	
Reference system	
Zero space stiffness	
Zero space damping	
Maximum Cartesian deviation from the path	
Maximum Cartesian force	
Optionally, separate Cartesian setpoint forces can be activated for each Cartesian direction.	DESIREDFORCE (>>> 6.3.4 "DESIREDFORCE" Page 121)

6.1.3 Axis-specific stiffness controller

This controller can only be configured in program mode by means of $STIFFNESS. There is no softkey available for this controller in the KUKA.HMI user interface.

Description

This controller is a stiffness controller at joint level. It is based on a lower-level torque controller. If a stiffness controller is used, the robot behaves like a spring with stiffness and damping parameters.

The following parameters are predefined for all 7 axes:

Parameter	
Axis-specific setpoint position	Motion programming
Axis-specific setpoint stiffness	$STIFFNESS (>>> 6.2 "Configuring servo controllers via $STIFFNESS" Page 113)
Axis-specific setpoint damping	
Maximum axis-specific deviation	
Maximum axis-specific torque	

6.1.4 Gravitation compensation

Description

In the case of gravitation compensation, only the torque controller at joint level is active. As the setpoint value, the torque controller receives the gravitational torque that is calculated on the basis of a model for each axis. The controller maintains the position if there is no external force acting on the robot. In the case of slight external force, gravitation compensation is activated. The robot can be manually guided by the user with little external force. Precondition: T1 mode.

⚠ CAUTION In the case of gravitation compensation, the velocity in T1 mode is not restricted to a maximum of 250 mm/s. Higher velocities can be reached.

⚠ CAUTION If the robot is moved with gravitation compensation activated, the safety precautions for manual guidance must be observed. Failure to observe these precautions may result in physical injuries and damage to property.
(>>> 4.17 "Manual guidance of the lightweight robot" Page 68)

In the case of gravitation compensation, Automatic mode, moving the robot in program mode and manual jogging with the jog keys and Space Mouse are not possible.

One precondition for optimal functioning of the controller is that the load data and the gravitation vector have been entered correctly. If this precondition is met and an arm moves from the rest position without external force being applied, the offset of the torque sensors must be checked. This may be the case after the arm has been overloaded, for example.

The current value of the torque sensors can be displayed by means of the system variable $TORQUE_AXIS_ACT. In the case of an offset in the torque mastering position, the torque sensors must be reset. The system variable $TORQUE_AXIS_EST shows the deviations from the dynamic model and should have a value close to zero.

(>>> 5.4.4 "Carrying out torque mastering" Page 90)

⚠ **CAUTION** The following parameters must be entered correctly in the case of gravitation compensation. Unexpected robot motions may otherwise result.

- Load data
- Alignment of the robot ($ROBROOT, $GRAVITATION)

  Only required if the robot is not being operated as a floor-mounted robot.

6.2 Configuring servo controllers via $STIFFNESS

The controllers can be configured in program mode by means of the system variable $STIFFNESS.

Components of the structure type:

Components	Data type	Description	10	20	30	101
STRATEGY	INT	Controller type; selection via controller number (>>> 6.1 "Overview of servo controllers" Page 111) **Note**: It is advisable to select gravitation compensation manually. (>>> 4.16 "Manually selecting servo controllers" Page 66)	X	X	X	X
FRAMETYPE	ENUM	Reference coordinate system - **#TOOL**: Cartesian spring stiffness, relative to the TOOL coordinate system. - **#BASE**: Cartesian spring stiffness, relative to the BASE coordinate system.		X		
BASE	FRAME	BASE coordinate system for Cartesian stiffness control		X		
TOOL	FRAME	TOOL coordinate system for Cartesian stiffness control		X		
CPSTIFFNESS	FRAME	Cartesian spring stiffness		X		
CPDAMPING	FRAME	Cartesian spring damping		X		

Components	Data type	Description	10	20	30	101
AXISSTIFFNESS	E6AXIS	Axis-specific spring stiffness **Note**: In the case of a Cartesian stiffness controller, this parameter defines the zero space stiffness.		X	X	
AXISDAMPING	E6AXIS	Axis-specific spring damping **Note**: In the case of a Cartesian stiffness controller, this parameter defines the zero space damping.		X	X	
CPMAXDELTA	FRAME	Limitation of the Cartesian deviation from the path Only positive values are permissible. The value “0” is not permissible. Default value: **99.0 mm** If the Cartesian spring stiffness is relative to the BASE coordinate system (#BASE), the following range of values applies: - **0.01 … 100.00 mm**		X		
MAXFORCE	FRAME	Limitation of the Cartesian force at the TCP: - Positive value: limitation active. Recommended maximum value: 150 N **Note**: Depending on the specific application, a lower value may be appropriate. - Negative value: limitation not active. (>>> 6.2.3 "$STIFFNESS component MAXFORCE" Page 118)		X		
AXISMAXDELTA	E6AXIS	Limitation of the axis-specific deviation Only positive values and the value “0” are permissible. - Positive value: limitation active. - **0**: limitation not active. Default value: **10.0°**			X	

Components	Data type	Description	10	20	30	101
AXISMAXDELTATRQ	E6AXIS	Limitation of the axis-specific torque: ■ Positive value: limitation active. ■ Negative value: limitation not active.			X	
COMMIT	BOOL	Only relevant if values are assigned to individual components of the system variable. These values are not valid until COMMIT is set to TRUE. COMMIT is then automatically reset to FALSE by the system. The next time a value is assigned, the user must reset COMMIT to TRUE.	(>>> 6.2.4 "Assigning values to $STIFFNESS" Page 118)			

6.2.1 Cartesian stiffness controller: parameterization

Minimum and maximum permissible values:

	Stiffness				Damping			
	Min.	Max.	Default	Unit	Min.	Max.	Default	Unit
X	0.01	5 000	2 000	[N/m]	0.1	1.0	0.7	[N*s/m]
Y	0.01	5 000	2 000	[N/m]	0.1	1.0	0.7	[N*s/m]
Z	0.01	5 000	2 000	[N/m]	0.1	1.0	0.7	[N*s/m]
A	0.01	300	200	[Nm/rad]	0.1	1.0	0.7	[Nm*s/rad]
B	0.01	300	200	[Nm/rad]	0.1	1.0	0.7	[Nm*s/rad]
C	0.01	300	200	[Nm/rad]	0.1	1.0	0.7	[Nm*s/rad]

The limit values for the stiffness are valid for TCP positions situated near the robot flange. The further the TCP is from the end of the arm ($STIFFNESS.TOOL very high), the greater the joint stiffness must be with constant Cartesian stiffness.

NOTICE Excessive stiffness (> 5,000 or > 300) or damping (> 1) in stiffness control can cause the robot to buzz. This is also the case if $STIFFNESS.TOOL is too great and the configured stiffness cannot be reached. Damage to the industrial robot may result.

If the stiffness is set to zero, the robot is controlled in the same way as with gravitation compensation. The robot can be freely moved in the specified direction and remains torque controlled. The joint friction and gravitational torque are compensated via the motor.

- Damping value 0: undamped vibration (not recommended)
- Damping value 0.7: default setting
- Damping value 1: completely damped vibration (no overshoot)

The positioning accuracy in the individual Cartesian directions diminishes with decreasing setpoint stiffness, because in the case of very low stiffness values spring force F is no longer sufficient to overcome the static friction.

Solution strategies:

- TRIGBYCONTACT
  (>>> 6.3.2 "TRIGBYCONTACT" Page 120)
- Alternating use of position and stiffness controllers

If the robot moves the TCP to the contact point and the programmed position has not yet been reached, a force is generated.

Represented simply, the difference in position resulting from Hooke's law and spring stiffness C (CPSTIFFNESS) is:

$\Delta x = F / C$

For spring force F:

$F = C * \Delta x$

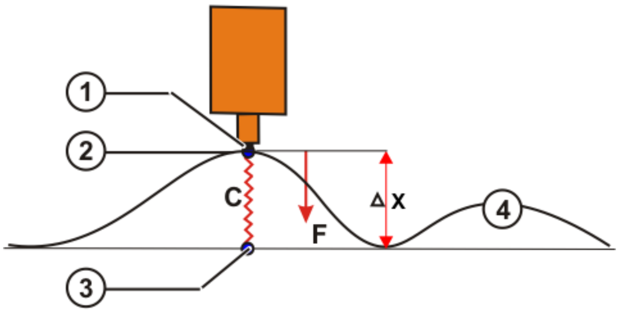


Fig. 6-1: Position difference

1	TCP ($STIFFNESS.TOOL)	3	Programmed position
2	Measured position	4	Profile of surface

The Cartesian difference between programmed and measured position can be displayed via the system variable $POS_DELTA_ACT.
$POS_DELTA_ACT = $POS_ACT_CMD - $POS_ACT_MES

The force exerted by the TCP at the contact point depends on the difference between the command position and the actual position and the Cartesian stiffness.

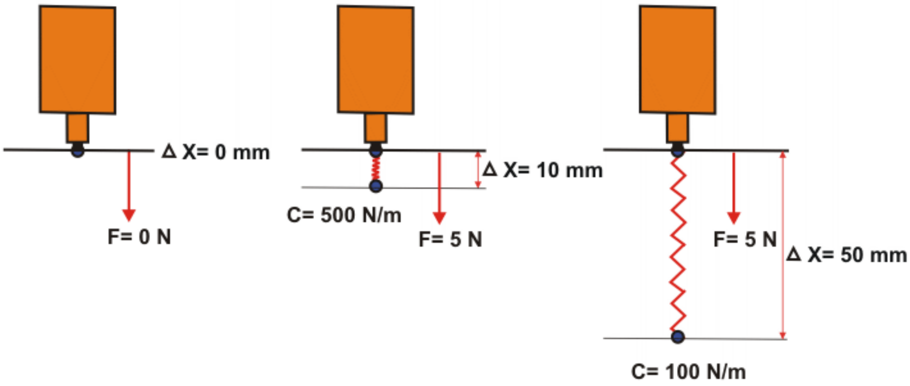


Fig. 6-2: Cartesian force at TCP (example)

A large position difference and low stiffness can result in the same force as a smaller position difference and greater stiffness. If the Cartesian velocity and the override are identical, the time required to reach this force differs.

In the case of a small position difference and high stiffness, the force is reached more quickly, as the command position is reached earlier. This may result in a jerk, for example.

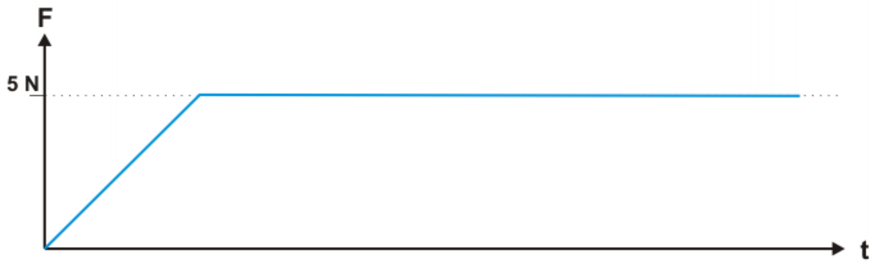


Fig. 6-3: Force over time (high stiffness, small position difference)

In the case of a large position difference and low stiffness, the force is built up more slowly. This can be used, for example, if the robot moves to the contact point and the impact loads are to be reduced.

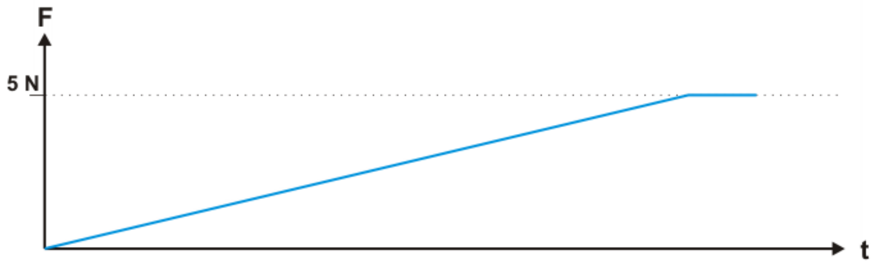


Fig. 6-4: Force over time (low stiffness, large position difference)

6.2.2 Axis-specific stiffness controller: parameterization

Minimum and maximum permissible values:

	Stiffness [Nm/rad]			Damping [Nm*s/rad]		
	Min.	Max.	Default	Min.	Max.	Default
A6	0.01	2 000	1 000	0.1	1.0	0.7
A5	0.01	2 000	1 000	0.1	1.0	0.7
A4	0.01	2 000	1 000	0.1	1.0	0.7
A3	0.01	2 000	1 000	0.1	1.0	0.7
E1	0.01	2 000	1 000	0.1	1.0	0.7
A2	0.01	2 000	1 000	0.1	1.0	0.7
A1	0.01	2 000	1 000	0.1	1.0	0.7

NOTICE Excessive stiffness (> 2,000) or damping (> 1) in stiffness control can cause the robot to buzz. Damage to the industrial robot may result.

If the stiffness is set to zero, the axis is controlled in the same way as with gravitation compensation. The axis can be moved freely and remains torque controlled. The joint friction and gravitational torque are compensated via the motor.

- Damping value 0: undamped vibration (not recommended)
- Damping value 0.7: default setting
- Damping value 1: completely damped vibration (no overshoot)

6.2.3 $STIFFNESS component MAXFORCE

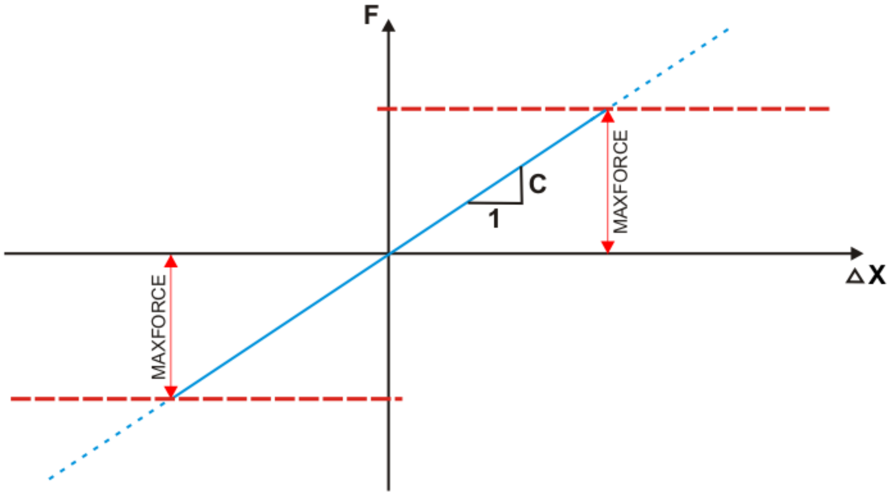


Fig. 6-5: MAXFORCE

F	Force exerted at the TCP
ΔX	Position difference
C	Spring stiffness

6.2.4 Assigning values to $STIFFNESS

ℹ The assignment of values to $STIFFNESS triggers an advance run stop. The advance run stop can be avoided by programming interrupt or trigger commands.

Description

The following possibilities exist for assigning values to the variable $STIFFNESS:

- Assigning values via a user-defined variable. This is the recommended procedure.
- Assigning values for individual components. COMMIT must be set to TRUE for these values to become valid.

  This option should only be used in exceptional cases, e.g. if, for a specific process, only the stiffness of a controller is to be modified.

Examples

Assignment via the variable USERSTIFF

ℹ A controller does not use all components of $STIFFNESS. It is advisable to assign suitable values even to the unused components. An unexpected control response may otherwise occur.

```
DEF SetTaskStiffness ()

DECL STIFFNESS USERSTIFF
USERSTIFF = $STIFFNESS
USERSTIFF.STRATEGY = 20
USERSTIFF.FRAMETYPE = #TOOL
USERSTIFF.CPSTIFFNESS = {X 2000.0,Y 2000.0,Z 2000.0,A 150.0,
                         B 150.0,C 150.0}
USERSTIFF.CPDAMPING = {X 0.7,Y 0.7,Z 0.7,A 0.7,B 0.7,C 0.7}
USERSTIFF.TOOL = TOOL_DATA[1]
USERSTIFF.AXISSTIFFNESS = {A1 2000.0,A2 2000.0,A3 2000.0,
                           A4 2000.0,A5 2000.0,A6 2000.0,
                           E1 2000.0}
USERSTIFF.AXISDAMPING = {A1 0.7,A2 0.7,A3 0.7,A4 0.7,A5 0.7,
                         A6 0.7,E1 0.7}
USERSTIFF.CPMAXDELTA = {X 7.0,Y 7.0,Z 7.0,A 2.0,B 2.0,C 2.0}
USERSTIFF.MAXFORCE = {X 70.0,Y 70.0,Z 70.0,A 10.0,B 10.0,C 10.0}
USERSTIFF.AXISMAXDELTA = {A1 10.0,A2 10.00,A3 10.0,A4 10.0,
                          A5 10.0,A6 10.0,E1 10.0}
USERSTIFF.AXISMAXDELTATRQ = {A1 2.0,A2 2.0,A3 2.0,A4 2.0,
                                A5 2.0,A6 2.0,E1 2.0}
$STIFFNESS = USERSTIFF

END
```

Assignment for individual components

```
DEF SetPositionControl()

$STIFFNESS.STRATEGY = 10
$STIFFNESS.COMMIT = TRUE

END

DEF SetExecutionStiff()

$STIFFNESS.CPSTIFFNESS = {X 2000.0,Y 2000.0,Z 2000.0,A 150.0,
                          B 150.0,C 150.0}
$STIFFNESS.TOOL = TOOL_DATA[3]
$STIFFNESS.COMMIT = TRUE

END
```

6.3 Motion Driver commands

Overview

Additional functions for stiffness control can be configured in program mode by means of Motion Driver commands.

Switching controllers	
TRIGBYCONTACT	(>>> 6.3.2 "TRIGBYCONTACT" Page 120)
SWITCHNOW	(>>> 6.3.3 "SWITCHNOW" Page 121)

Activating Cartesian force	
DESIREDFORCE	(>>> 6.3.4 "DESIREDFORCE" Page 121)
DESIREDFORCESTART	(>>> 6.3.5 "DESIREDFORCESTART" Page 125)
DESIREDFORCECLEAR	(>>> 6.3.6 "DESIREDFORCECLEAR" Page 126)

6.3.1 Syntax of Motion Driver commands

Syntax

result = MD_CMD("*PAPAS*","*Name of command*",md_int[],md_real[])

Explanation of the syntax

Element	Description
PAPAS	Name of the driver
Name of command	(>>> 6.3 "Motion Driver commands" Page 119)
`md_int[]`	INTEGER parameter of the command Up to 16 parameters can be defined for each command.
`md_real[]`	REAL parameter of the command Up to 16 parameters can be defined for each command.

6.3.2 TRIGBYCONTACT

Description

TRIGBYCONTACT prepares fast switching from a position controller to another controller. This controller is specified after the TRIGBYCONTACT statement and is not initially active. The system only switches to the new controller when the torque sensors measure a contact of the strength set by TRIGBYCONTACT.

If TRIGBYCONTACT is used to switch from a position controller to a Cartesian stiffness controller, DESIREDFORCE can also be used, if required, to activate a setpoint force. This force activation, like the controller, is specified after the TRIGBYCONTACT statement and is not activated until a contact of the set strength is measured.

Switching controllers with TRIGBYCONTACT does not trigger an advance run stop, i.e. the controller is switched during the motion.

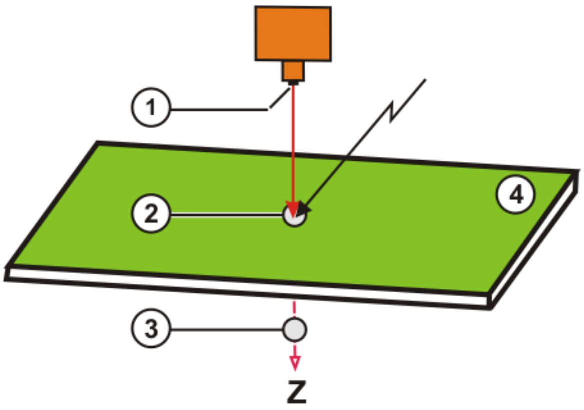


Fig. 6-6: Functional principle of TRIGBYCONTACT

1	TCP	3	Programmed position
2	TRIGBYCONTACT	4	Workpiece

⚠ CAUTION The TCP may not move towards the workpiece at the maximum velocity of 250 mm/s. This can cause severe damage to the robot or workpiece. The switching of the controller must be tested first at sufficiently reduced velocity. The velocity can then be increased gradually to optimize the operation.

Parameter	Description
md_real[1]	Force threshold for switching the controller The factor specified here refers to the maximum torque of the individual joints. With a factor of 0.05, for example, the force threshold for switching controller is 5% of the maximum torque.
md_int[1] … md_int[16]	No function
md_real[2] … md_real[16]	No function

6.3.3 SWITCHNOW

Description

SWITCHNOW can be used to switch to the controller prepared by means of TRIGBYCONTACT without there being a contact of the set strength present.

Parameter	Description
md_int[1] ... md_int[16]	No function
md_real[1] ... md_real[16]	No function

6.3.4 DESIREDFORCE

Description

In the case of the Cartesian stiffness controller, DESIREDFORCE can be used to activate a separate Cartesian setpoint force for each Cartesian direction:

- DESIREDFORCEX
- DESIREDFORCEY
- DESIREDFORCEZ
- DESIREDFORCEA
- DESIREDFORCEB
- DESIREDFORCEC

The reference coordinate system for the force activation must be parameterized in $STIFFNESS.

This function can be used to support the following processes, for example:

- Joining processes: generation of vibration at the TCP.
- Tracing a surface profile: keeping the force constant in the activated direction.

> ⚠ **CAUTION** When activating the force, the robot is deflected in the direction of the activated force. It must be ensured that this cannot result in personal injury or material damage. This can be achieved, for example, by limiting the permissible deflection and velocity when the force is activated.

The deflection or difference in position resulting from Hooke’s law and the spring stiffness CPSTIFFNESS is:

Δx = DESIREDFORCE / CPSTIFFNESS

In the case of low stiffness values, the difference in position can be so great that a BCO run is no longer possible. The BCO run must then be carried out in a different controller mode.

Force activations configured with DESIREDFORCE are deleted in the following cases:

- After new parameters have been set for $STIFFNESS
- After the drives have been switched off, e.g. by pressing the EMERGENCY STOP button or by fully pressing down or releasing the enabling switch in T1/T2 mode
- After setting DESIREDFORCECLEAR

Parameter	Description
md_int[1]	Force profile - **0**: No force - **1**: Pulse (constant force) - **2**: Sine curve
md_int[2]	Force activation - **0**: Force must be activated with DESIREDFORCE-START. - **1**: Force is activated immediately.
md_int[3]	Time base relative to the force profile - **0**: Shared time base for the Cartesian directions (synchronization) - **1**: Separate time base for a single Cartesian direction
md_real[1]	Amplitude Unit: - Force in the X, Y, Z direction: **N** - Torque about A, B, C: **Nm**
md_real[2]	Duration Unit: s **Note**: If a program is stopped with HALT in jog mode, pressing the enabling switch after the duration has elapsed will cause the robot to move, even without the Start key.
md_real[3]	Maximum permissible deflection Unit: - Deflection in the X, Y, Z direction: **mm** - Deflection about A, B, C: **°** Limitation is only active if the value is positive. There is no limitation in the case of a negative value. **Note**: If the maximum permissible deflection is exceeded, the force is deactivated The force is reactivated as soon as the robot is back in the permissible range.

Parameter	Description
md_real[4]	Maximum permissible velocity: Unit: - Velocity in the X, Y, Z direction: **m/s** - Velocity about A, B, C: **°/s** Limitation is only active if the value is positive. There is no limitation in the case of a negative value. **Note**: If the maximum permissible velocity is exceeded, the force is deactivated The force is reactivated as soon as the robot is back in the permissible range.
md_real[5] … md_real[9]	The function of the parameters depends on the force profile. **Profile 1:** pulse (md_int[1] = 1): - md_real[5]: rise time [s] - md_real[6]: fall time [s] Total duration of profile 1: md_real[2]+md_real[5]+md_real[6] **Profile 2:** sine curve (md_int[1] = 2): - md_real[5]: frequency [Hz]; max. 15 Hz - md_real[6]: phase position [°] - md_real[7]: force offset [N] or [Nm] relative to the amplitude - md_real[8]: rise time [s] - md_real[9]: fall time [s] Total duration of profile 2: md_real[2]+md_real[8]+md_real[9]
md_real[10] … md_real[16]	No function
md_int[4] … md_int[16]	No function

6.3.4.1 DESIREDFORCE – force profiles

Example 1 Activation of a constant force

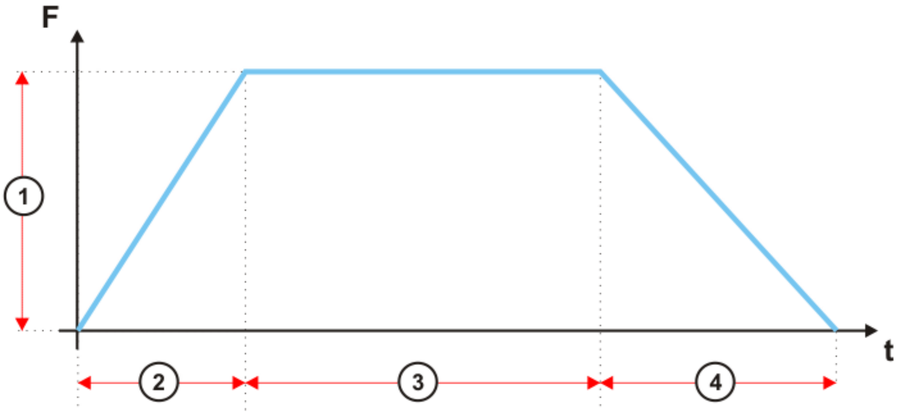


Fig. 6-7: Example: profile 1 – pulse

1	Amplitude	3	Duration
2	Rise time	4	Fall time

Example 2

Activation of a sinusoidal force with force offset

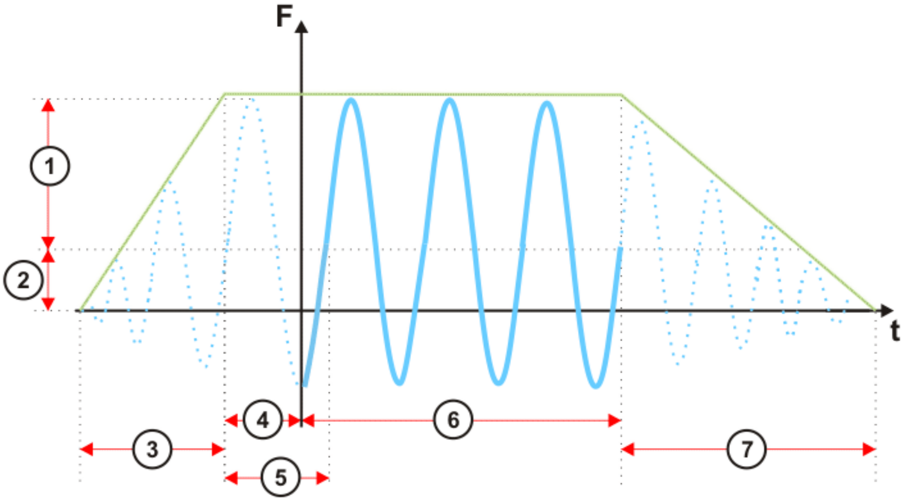


Fig. 6-8: Example: profile 2 – sine curve with offset

1	Amplitude	5	Frequency
2	Force offset	6	Duration
3	Rise time	7	Fall time
4	Phase position		

Example 3

Sine in X and Y direction are to be combined to form a circle. This presupposes synchronization of the sine curves (shared time base). After 5 s, a pulse is to be generated in the Z direction for 1 s. A separate time base must be used for the Z direction, as a pulse always starts at zero.

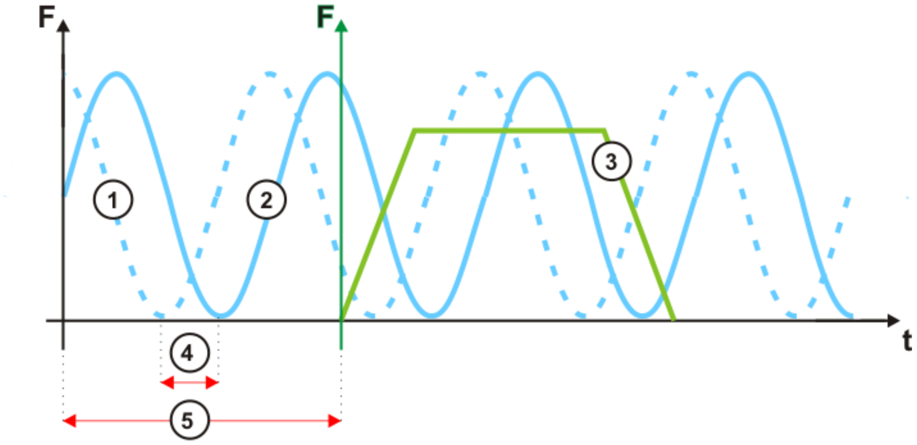


Fig. 6-9: Example: profile 2+1 – pulse with separate time base

1	Sine in the Y direction	4	Time base for sine curves
2	Sine in the X direction	5	Time base for the pulse
3	Pulse in the Z direction		

Example 4

2 sine functions of different frequencies can be overlapped to generate Lissajous curves and thus vibrations at the TCP. A force overlap of this kind can be used, for example, when inserting parts into a workpiece. If the part to be in-

serted into the workpiece is jammed, this can be resolved by means of vibration at the TCP.

In jog mode, vibration starts as soon as the enabling switch is pressed and the drives are switched on (without the Start key).

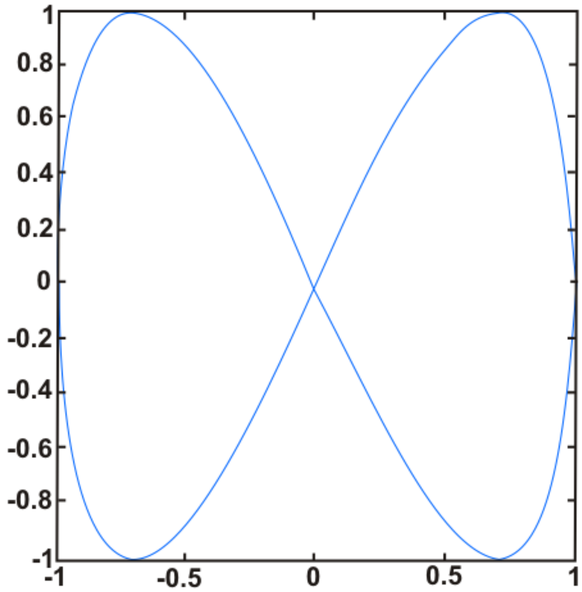


Fig. 6-10: Shape of vibration – Lissajous curve (bow)

(>>> 6.5.4 "Vibration for joining process (Lissajous)" Page 134)

Example 5 In combination with rise and fall times, it is possible to generate helical patterns.

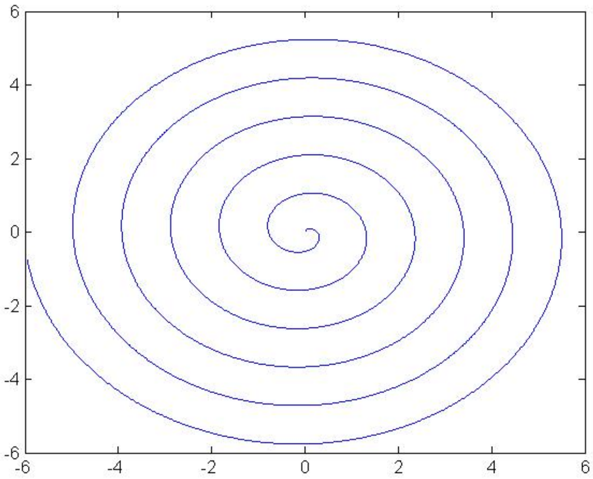


Fig. 6-11: Shape of vibration – spiral

(>>> 6.5.5 "Vibration for joining process (spiral)" Page 136)

6.3.5 DESIREDFORCESTART

Description DESIREDFORCESTART can be used to activate the force defined with DESIREDFORCE. Precondition: for DESIREDFORCE: md_int[2]=0.

DESIREDFORCESTART cannot be used if force activations configured with DESIREDFORCE have been deleted, e.g. with DESIREDFORCECLEAR.

Parameter	Description
md_int[1] ... md_int[16]	No function
md_real[1] ... md_real[16]	No function

6.3.6 DESIREDFORCECLEAR

Description

DESIREDFORCECLEAR can be used to delete all force activations configured with DESIREDFORCE. These cannot be reactivated with DESIREDFORCESTART; instead, they must be set again.

Parameter	Description
md_int[1] ... md_int[16]	No function
md_real[1] ... md_real[16]	No function

6.4 Back estimation of the force measurement

Description

The system variable $TORQUE_TCP_EST can be used to carry out a back estimate of the force measurement of the torque sensors relative to the reference coordinate system.

Components of the structure type	Data type	Description
FT	FRAME	Force estimate in [N] or [Nm]
VARIANCE	FRAME	Evaluation of the quality of the estimate Singular axis positions are included in the evaluation of the force measurement. In the case of a variance >10, the force estimate is not reliable. **Note**: Depending on the specific application, variances ≥10 may also be reliable.
FRAMETYPE	ENUM	Reference coordinate system ▪ **#TOOL**: Cartesian spring stiffness, relative to the TOOL coordinate system. ▪ **#BASE**: Cartesian spring stiffness, relative to the BASE coordinate system.

Programming

```
DECL TORQUE_TCP_EST MyTorque
MyTorque = $TORQUE_TCP_EST

IF MyTorque.VARIANCE.X < 10 THEN
;MyTorque.FT.X is reliable
ELSE
;MyTorque.FT.X is not reliable
ENDIF
```

This is the recommended programming for 2 reasons:

- The variance is checked before the estimated value is used in subsequent program execution.
- Data consistency is assured, as variance and estimated value are read simultaneously.

6.5 Configuration examples

> **⚠ WARNING** Death, serious physical injuries or major damage to property may occur. If a program is executed in test mode T2 or after a change of controller or parameters (irrespective of the operating mode), the operator must be in a position outside the danger zone.

6.5.1 Parameterization of $STIFFNESS

```
1  DECL STIFFNESS USERSTIFF
2
3  INI
4
5  $STIFFNESS.STRATEGY = 10
6  $STIFFNESS.COMMIT = TRUE
7  $OV_PRO = 75
8  PTP StartPos Vel= 100% PDAT3 Tool[4]:Test Base[0]
9  HALT
10
11 USERSTIFF = $STIFFNESS
12 USERSTIFF.STRATEGY = 20
13 USERSTIFF.FRAMETYPE = #TOOL
14 USERSTIFF.TOOL = {X 0.0,Y 0.0,Z 180.0,A 0.0,B 0.0,C 0.0}
15 USERSTIFF.CPSTIFFNESS = {X 200.0,Y 2000.0,Z 400.0,A 200.0,
                         B 200.0,C 200.0}
16 USERSTIFF.CPDAMPING = {X 0.7,Y 0.7,Z 0.7,A 0.7,B 0.7,C 0.7}
17 USERSTIFF.AXISSTIFFNESS = {A1 2000.0,A2 2000.0,A3 2000.0,
                           A4 2000.0,A5 2000.0,A6 2000.0,
                           E1 2000.0}
18 USERSTIFF.AXISDAMPING = {A1 0.7,A2 0.7,A3 0.7,A4 0.7,A5 0.7,
                         A6 0.7,E1 0.7}
19 USERSTIFF.CPMAXDELTA = {X 7.0,Y 7.0,Z 7.0,A 2.0,B 2.0,C 2.0}
20 USERSTIFF.MAXFORCE = {X 70.0,Y 70.0,Z 70.0,A 10.0,B 10.0,C 10.0}
21 USERSTIFF.AXISMAXDELTA = {A1 10.0,A2 10.00,A3 10.0,A4 10.0,
                          A5 10.0,A6 10.0,E1 10.0}
22 USERSTIFF.AXISMAXDELTATRQ = {A1 2.0,A2 2.0,A3 2.0,A4 2.0,
                             A5 2.0,A6 2.0,E1 2.0}
23 $STIFFNESS = USERSTIFF
24
25 PTP StartPos Vel= 100% PDAT3 Tool[4]:Test Base[0]
```

Line	Description
5 … 8	For BCO run to start position Always select controller 10 for this (position controller); start position can also be adapted by teaching if required.
12	Controller 20: Cartesian stiffness controller
13	Cartesian stiffness, relative to the TOOL coordinate system
14	Data of the TOOL coordinate system
15	Cartesian stiffness The TCP can be moved easily by hand in the X direction, slightly less easily in the Z direction and hardly at all in the Y direction. Motions about angles A, B and C are not possible.
16	Cartesian damping (default setting)
17	Axis-specific stiffness

Line	Description
18	Axis-specific damping (default setting)
19	Maximum Cartesian deflection
20	Maximum applied Cartesian force
21	Maximum axis-specific deflection
22	Maximum applied axis torque

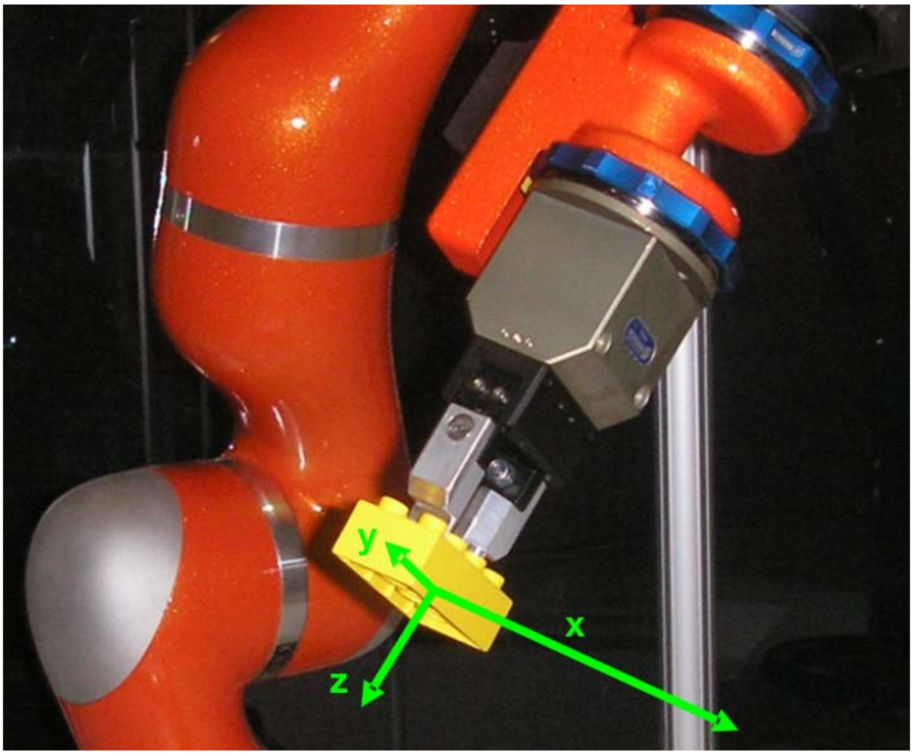


Fig. 6-12: Example: parameterization of $STIFFNESS

6.5.2 Tracing a surface profile

Main program

```
DEF FollowShape()

INI

$STIFFNESS.STRATEGY = 10
$STIFFNESS.COMMIT = TRUE
$OV_PRO = 75
PTP {A1 -20.0,A2 120.0,A3 -45.0,A4 -10.0,A5 90.0,A6 -130.0,
     E1 20.0}
PTP StartPos Vel= 100% PDAT2 Tool[4]:Test Base[0]
HALT

SoftInZ()

$OV_PRO = 10

LIN_REL {Z -100}
LIN_REL {X 150}
HALT
LIN_REL {X 150}
HALT

END
```

Line	Description
5 … 9	For BCO run to start position just above the surface Always select controller 10 for this (position controller); start position can also be adapted by teaching if required.
12	Subprogram **SoftInZ**: ■ Switching to Cartesian stiffness control (soft in Z direction)
14	Reduced program override
16	TCP is moved in Z direction to make contact with the surface. The end point is beneath the contact point on the surface.
17, 19	TCP is twice moved 150 mm in X direction. Since the TCP is set as soft in the Z direction, it can follow the profile of the surface.

Subprogram SoftInZ

```
24 DEF SoftInZ()
25 DECL STIFFNESS USERSTIFF
26
27 USERSTIFF = $STIFFNESS
28 USERSTIFF.STRATEGY = 20
29 USERSTIFF.FRAMETYPE = #TOOL
30 USERSTIFF.TOOL = {X 0.0,Y 0.0,Z 180.0,A 0.0,B 0.0,C 0.0}
31 USERSTIFF.CPSTIFFNESS = {X 2000.0,Y 2000.0,Z 100.0,A 200.0,
                            B 200.0,C 200.0}
32 USERSTIFF.CPDAMPING = {X 0.7,Y 0.7,Z 0.7,A 0.7,B 0.7,C 0.7}
33 USERSTIFF.AXISSTIFFNESS = {A1 2000.0,A2 2000.0,A3 2000.0,
                              A4 2000.0,A5 2000.0,A6 2000.0,
                              E1 2000.0}
34 USERSTIFF.AXISDAMPING = {A1 0.7,A2 0.7,A3 0.7,A4 0.7,A5 0.7,
                            A6 0.7,E1 0.7}
35 USERSTIFF.CPMAXDELTA = {X 7.0,Y 7.0,Z 50.0,A 2.0,B 2.0,C 2.0}
36 USERSTIFF.MAXFORCE = {X 70.0,Y 70.0,Z 70.0,A 10.0,B 10.0,C 10.0}
37 USERSTIFF.AXISMAXDELTA = {A1 10.0,A2 10.00,A3 10.0,A4 10.0,
                             A5 10.0,A6 10.0,E1 10.0}
38 USERSTIFF.AXISMAXDELTATRQ = {A1 2.0,A2 2.0,A3 2.0,A4 2.0,
                                A5 2.0,A6 2.0,E1 2.0}
39 $STIFFNESS = USERSTIFF
40
41 END
```

Line	Description
27 … 38	Parameterization of the controller
28	Controller 20: Cartesian stiffness controller
29	Cartesian stiffness, relative to the TOOL coordinate system
30	Data of the TOOL coordinate system
31	Cartesian stiffness TCP does not yield in the X and Y directions and is set to soft in the Z direction. Motions about angles A, B and C are not possible.
32	Cartesian damping (default setting)
33	Axis-specific stiffness
34	Axis-specific damping (default setting)
35	Maximum Cartesian deflection (50 mm in the Z direction and 2°)
36	Maximum applied Cartesian force
37	Maximum axis-specific deflection
38	Maximum applied axis torque

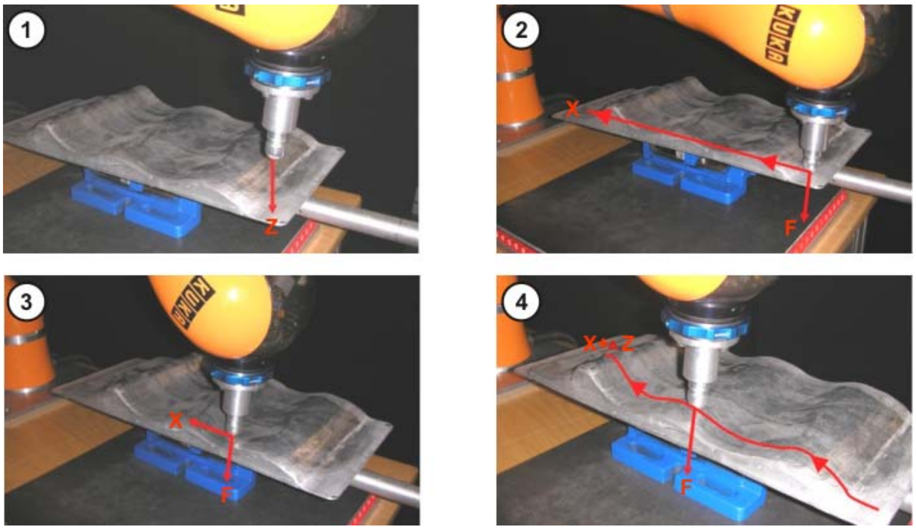


Fig. 6-13: Example: following the profile of a surface

Item	Description
1	The TCP is moved in the Z direction from the start position.
2	Contact is made with the surface. The programmed point is beneath the contact point on the surface. The difference in position between the contact point and the programmed point, in conjunction with spring stiffness CPSTIFFNESS, gives force F which is exerted on the surface (F = CPSTIFFNESS * Δx).
3	TCP moves in the X direction (programmed path).
4	TCP does not follow the programmed path, but the profile of the surface. As the deflection in the Z direction changes, force F varies according to the position of the TCP on the surface.

6.5.3 Activating constant force with DESIREDFORCE

Main program

```
1  DEF FollowShape()
2
3  DECL TORQUE_TCP_EST MyTorque
4
5  INI
6
7  LOOP
8  $STIFFNESS.STRATEGY = 10
9  $STIFFNESS.COMMIT = TRUE
10 $OV_PRO = 75
11 PTP {A1 -20.0,A2 120.0,A3 -45.0,A4 -10.0,A5 90.0,A6 -130.0,E1
20.0}
12 HALT
13 PTP StartPos Vel= 100% PDAT2 Tool[4]:Test Base[0]
14
15 ForceZ()
16
17 $OV_PRO = 10
18
19 REPEAT
20 MyTorque = $TORQUE_TCP_EST
21 UNTIL MyTorque.FT.Z >= 15
22
23 LIN_REL {X 150}
24 HALT
25 LIN_REL {X 150}
26 WAIT FOR TRUE
27
28 ForceZ Stop()
29 HALT
30 ENDLOOP
31
32 END
```

Line	Description
8 … 13	For BCO run to start position just above the surface Always select controller 10 for this (position controller); start position can also be adapted by teaching if required.
15	Subprogram **ForceZ**: ■ Switching to Cartesian stiffness control (soft in Z direction) ■ Activation of a constant force in the Z direction with DESIREDFORCEZ
17	Reduced program override
19 … 21	When the force is activated, the TCP is deflected in the Z direction until contact is made with the surface and the force estimated by $TORQUE_TCP_EST is at least 15 N. This minimum force must be less than the force activated with DESIREDFORCEZ.
23 … 26	TCP is twice moved 150 mm in X direction. Since the TCP is set as soft in the Z direction, it can follow the profile of the surface.
28	Subprogram **ForceZ Stop**: ■ Deletion of the activated force in the Z direction with DESIREDFORCECLEAR

Subprogram ForceZ

```
33 DEF ForceZ()
34 DECL STIFFNESS USERSTIFF
35
36 DECL INT md_int[16]
37 DECL REAL md_real[16]
38 DECL INT n
39 DECL INT result
40
41 USERSTIFF = $STIFFNESS
42 USERSTIFF.STRATEGY = 20
43 USERSTIFF.FRAMETYPE = #TOOL
44 USERSTIFF.CPSTIFFNESS = {X 2000.0,Y 2000.0,Z 1.0,A 200.0,
                            B 200.0,C 200.0}
45 USERSTIFF.CPDAMPING = {X 0.7,Y 0.7,Z 0.7,A 0.7,B 0.7,C 0.7}
46 USERSTIFF.AXISSTIFFNESS = {A1 2000.0,A2 2000.0,A3 2000.0,
                              A4 2000.0,A5 2000.0,A6 2000.0,
                              E1 2000.0}
47 USERSTIFF.AXISDAMPING = {A1 0.7,A2 0.7,A3 0.7,A4 0.7,A5 0.7,
                            A6 0.7,E1 0.7}
48 USERSTIFF.CPMAXDELTA = {X 7.0,Y 7.0,Z 50.0,A 2.0,B 2.0,C 2.0}
49 USERSTIFF.MAXFORCE = {X 70.0,Y 70.0,Z 70.0,A 10.0,B 10.0,C 10.0}
50 USERSTIFF.AXISMAXDELTA = {A1 10.0,A2 10.00,A3 10.0,A4 10.0,
                             A5 10.0,A6 10.0,E1 10.0}
51 USERSTIFF.AXISMAXDELTATRQ = {A1 2.0,A2 2.0,A3 2.0,A4 2.0,
                                A5 2.0,A6 2.0,E1 2.0}
52 $STIFFNESS = USERSTIFF
53
54 FOR n=1 TO 16
55  md_int[n] = 0
56  md_real[n] = 0
57 ENDFOR
58
59 md_int[1] = 1
60 md_int[2] = 0
61 md_int[3] = 0
62 md_real[1] = 20.0
63 md_real[2] = 50.0
64 md_real[3] = 100.0
65 md_real[4] = 0.2
66 md_real[5] = 0.1
67 md_real[6] = 0.0
68 md_real[7] = 0.0
69
70 result=MD_CMD("PAPAS","DESIREDFORCEZ",md_int[],md_real[])
71 result=MD_CMD("PAPAS","DESIREDFORCESTART",md_int[],md_real[])
72
73 END
```

Line	Description
36 … 39	Declaration of the parameters for the force activation
41 … 51	Parameterization of the controller
42	Controller 20: Cartesian stiffness controller
43	Cartesian stiffness, relative to the TOOL coordinate system
44	Cartesian stiffness TCP does not yield in the X and Y directions and is set to soft in the Z direction. Motions about angles A, B and C are not possible.
45	Cartesian damping (default setting)
46	Axis-specific stiffness
47	Axis-specific damping (default setting)
48	Maximum Cartesian deflection (50 mm in the Z direction and 2°)
49	Maximum applied Cartesian force
50	Maximum axis-specific deflection
51	Maximum applied axis torque

Line	Description
54 … 57	All parameters must be set to zero.
59 … 68	Parameterization of DESIREDFORCEZ
59	A constant force is activated in the Z direction.
60	Force must be activated with DESIREDFORCESTART.
62	Constant force in the Z direction: 20 N
63	Duration: 50 s
64	Deflection in Z direction is limited to 100 mm.
65	Velocity in Z direction is limited to 0.2 m/s.
66	Rise and fall time: each 0.1 s
71	Start DESIREDFORCEZ

Subprogram ForceZ Stop

```
75 DEF ForceZ Stop()
76
77 DECL INT md_int[16]
78 DECL REAL md_real[16]
79 DECL INT n
80 DECL INT result
81
82 FOR n=1 TO 16
83  md_int[n] = 0
84  md_real[n] = 0
85 ENDFOR
86
87 result=MD_CMD("PAPAS","DESIREDFORCECLEAR",md_int[],md_real[])
88
89 END
```

Line	Description
77 … 80	Declaration of the parameters for cancelation of the force activation
82 …85	All parameters must be set to zero.
87	Deletion of DESIREDFORCEZ

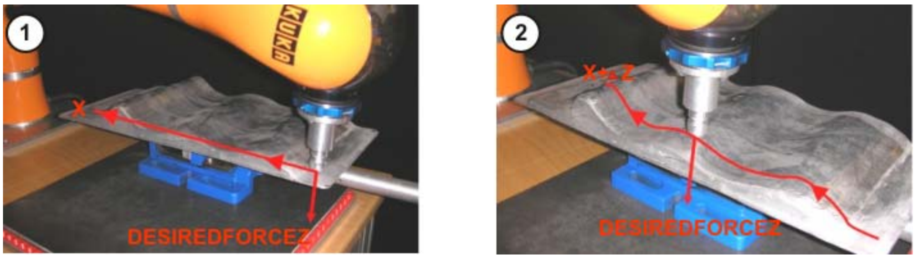


Fig. 6-14: Example: DESIREDFORCEZ

Item	Description
1	Contact is made with the surface. The force DESIREDFORCEZ is exerted on the surface. TCP moves in the X direction (programmed path). DESIREDFORCEZ is deleted after TCP has twice moved 150 mm in X direction.
2	TCP does not follow the programmed path, but the profile of the surface. The force DESIREDFORCEZ that is exerted on the surface is always constant. The profile is executed repeatedly in a loop.

6.5.4 Vibration for joining process (Lissajous)

```
 1 DEF LoopVib()
 2 DECL STIFFNESS USERSTIFF
 3
 4 DECL INT md_int[16]
 5 DECL REAL md_real[16]
 6 DECL INT result
 7
 8 PTP{A1 -90,A2 45,A3 45}
 9
10 USERSTIFF=$STIFFNESS
11 USERSTIFF.STRATEGY = 20
12 USERSTIFF.FRAMETYPE = #TOOL
13 USERSTIFF.CPSTIFFNESS = {X 2000.0,Y 2000.0,Z 2000.0,A 200.0,
                          B 200.0,C 200.0}
14 USERSTIFF.CPDAMPING = {X 0.7,Y 0.7,Z 0.7,A 0.7,B 0.7,C 0.7}
15 USERSTIFF.TOOL = {X 0.0,Y 0.0,Z 70.0,A 0.0,B 0.0,C 0.0}
16 USERSTIFF.AXISSTIFFNESS = {A1 2000.0,A2 2000.0,A3 2000.0,
                            A4 2000.0,A5 2000.0,A6 2000.0,
                            E1 2000.0}
17 USERSTIFF.AXISDAMPING = {A1 0.7,A2 0.7,A3 0.7,A4 0.7,A5 0.7,
                          A6 0.7,E1 0.7}
18 USERSTIFF.CPMAXDELTA = {X 7.0,Y 7.0,Z 7.0,A 2.0,B 2.0,C 2.0}
19 USERSTIFF.MAXFORCE = {X 70.0,Y 70.0,Z 70.0,A 10.0,B 10.0,C 10.0}
20 USERSTIFF.AXISMAXDELTA = {A1 10.0,A2 10.00,A3 10.0,A4 10.0,
                           A5 10.0,A6 10.0,E1 10.0}
21 USERSTIFF.AXISMAXDELTATRQ = {A1 2.0,A2 2.0,A3 2.0,A4 2.0,
                              A5 2.0,A6 2.0,E1 2.0}
22 $STIFFNESS=USERSTIFF
23
24 md_int[1] = 2
25 md_int[2] = 0
26 md_int[3] = 0
27 md_real[1] = 10.0
28 md_real[2] = 15.0
29 md_real[3] = 100.0
30 md_real[4] = 2.0
31 md_real[5] = 1.0
33 md_real[7] = 0.0
34
35 result=MD_CMD("PAPAS","DESIREDFORCEX",md_int[],md_real[])
36
37 md_int[1] = 2
38 md_int[2] = 0
39 md_int[3] = 0
40 md_real[1] = 10.0
41 md_real[2] = 15.0
42 md_real[3] = 100.0
43 md_real[4] = 2.0
44 md_real[5] = 2.0
45 md_real[6] = 0.0
46 md_real[7] = 0.0
47
48 result=MD_CMD("PAPAS","DESIREDFORCEY",md_int[],md_real[])
49 result=MD_CMD("PAPAS","DESIREDFORCESTART",md_int[],md_real[])
50
51 END
```

Line	Description
8	Start position A suitable start position must be selected for the specific joining process.
10 … 21	Parameterization of the controller
11	Controller 20: Cartesian stiffness controller
12	Cartesian stiffness, relative to the TOOL coordinate system
13	Cartesian stiffness (default setting)

Line	Description
14	Cartesian damping (default setting)
15	Data of the TOOL coordinate system
16	Axis-specific stiffness (maximum stiffness)
17	Axis-specific damping (default setting)
18	Maximum Cartesian deflection
19	Maximum applied Cartesian force
20	Maximum axis-specific deflection
21	Maximum applied axis torque
24 … 33	Parameterization of DESIREDFORCEX ■ Force profile: sine curve ■ Force must be activated with DESIREDFORCESTART. ■ Amplitude: 10 N or Nm ■ Duration: 15 s ■ Maximum permissible deflection: 100 mm or ° ■ Frequency: **1 Hz**
37 … 46	Parameterization of DESIREDFORCEY ■ Force profile: sine curve ■ Force must be activated with DESIREDFORCESTART. ■ Amplitude: 10 N or Nm ■ Duration: 15 s ■ Maximum permissible deflection: 100 mm or ° ■ Frequency: **2 Hz**
49	Start DESIREDFORCEZ and DESIREDFORCEY

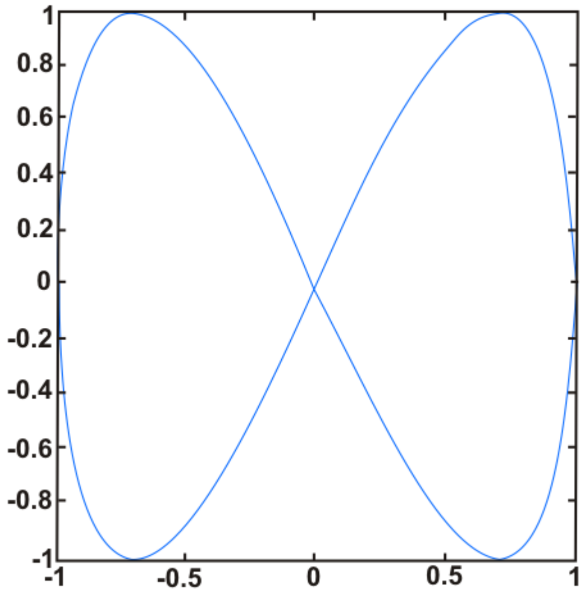


Fig. 6-15: Shape of vibration – Lissajous curve (bow)

6.5.5 Vibration for joining process (spiral)

```
 1 DEF SpiralVib()
 2 DECL STIFFNESS USERSTIFF
 3
 4 DECL INT md_int[16]
 5 DECL REAL md_real[16]
 6 DECL INT result
 7
 8 PTP{A1 -90,A2 45,A3 45}
 9
10 USERSTIFF=$STIFFNESS
11 USERSTIFF.STRATEGY = 20
12 USERSTIFF.FRAMETYPE = #TOOL
13 USERSTIFF.CPSTIFFNESS = {X 2000.0,Y 2000.0,Z 2000.0,A 200.0,
                            B 200.0,C 200.0}
14 USERSTIFF.CPDAMPING = {X 0.7,Y 0.7,Z 0.7,A 0.7,B 0.7,C 0.7}
15 USERSTIFF.TOOL = {X 0.0,Y 0.0,Z 70.0,A 0.0,B 0.0,C 0.0}
16 USERSTIFF.AXISSTIFFNESS = {A1 2000.0,A2 2000.0,A3 2000.0,
                              A4 2000.0,A5 2000.0,A6 2000.0,
                              E1 2000.0}
17 USERSTIFF.AXISDAMPING = {A1 0.7,A2 0.7,A3 0.7,A4 0.7,A5 0.7,
                            A6 0.7,E1 0.7}
18 USERSTIFF.CPMAXDELTA = {X 7.0,Y 7.0,Z 7.0,A 2.0,B 2.0,C 2.0}
19 USERSTIFF.MAXFORCE = {X 70.0,Y 70.0,Z 70.0,A 10.0,B 10.0,C 10.0}
20 USERSTIFF.AXISMAXDELTA = {A1 10.0,A2 10.00,A3 10.0,A4 10.0,
                              A5 10.0,A6 10.0,E1 10.0}
21 USERSTIFF.AXISMAXDELTATRQ = {A1 2.0,A2 2.0,A3 2.0,A4 2.0,
                                 A5 2.0,A6 2.0,E1 2.0}
22 $STIFFNESS=USERSTIFF
23
24 md_int[1] = 2
25 md_int[2] = 0
26 md_int[3] = 0
27 md_real[1] = 10.0
28 md_real[2] = 15.0
29 md_real[3] = 100.0
30 md_real[4] = 2.0
31 md_real[5] = 1.0
32 md_real[6] = 0.0
33 md_real[7] = 0.0
34 md_real[8] = 6.0
35
36 result=MD_CMD("PAPAS","DESIREDFORCEX",md_int[],md_real[])
37
38 md_int[1] = 2
39 md_int[2] = 0
40 md_int[3] = 0
41 md_real[1] = 10.0
42 md_real[2] = 15.0
43 md_real[3] = 100.0
44 md_real[4] = 2.0
45 md_real[5] = 1.0
46 md_real[6] = 90.0
47 md_real[7] = 0.0
48 md_real[8] = 6.0
49
50 result=MD_CMD("PAPAS","DESIREDFORCEY",md_int[],md_real[])
51 result=MD_CMD("PAPAS","DESIREDFORCESTART",md_int[],md_real[])
52
53 END
```

Line	Description
8	Start position A suitable start position must be selected for the specific joining process.
10 … 21	Parameterization of the controller
11	Controller 20: Cartesian stiffness controller

Line	Description
12	Cartesian stiffness, relative to the TOOL coordinate system
13	Cartesian stiffness (default setting)
14	Cartesian damping (default setting)
15	Data of the TOOL coordinate system
16	Axis-specific stiffness (maximum stiffness)
17	Axis-specific damping (default setting)
18	Maximum Cartesian deflection
19	Maximum applied Cartesian force
20	Maximum axis-specific deflection
21	Maximum applied axis torque
24 … 34	Parameterization of DESIREDFORCEX ■ Force profile: sine curve ■ Force must be activated with DESIREDFORCESTART. ■ Amplitude: 10 N or Nm ■ Duration: 15 s ■ Maximum permissible deflection: 100 mm or ° ■ Frequency: **1 Hz** ■ Rise time: 6 s
38 … 48	Parameterization of DESIREDFORCEY ■ Force profile: sine curve ■ Force must be activated with DESIREDFORCESTART. ■ Amplitude: 10 N or Nm ■ Duration: 15 s ■ Maximum permissible deflection: 100 mm or ° ■ Frequency: **1 Hz** ■ Phase position: 90° ■ Rise time: 6 s
51	Start DESIREDFORCEZ and DESIREDFORCEY

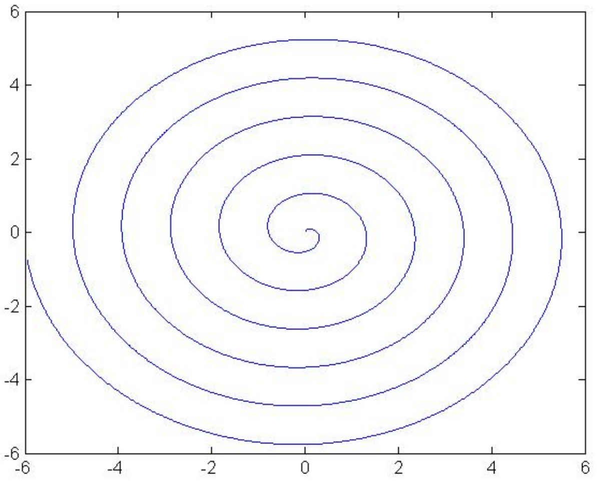


Fig. 6-16: Shape of vibration – spiral

6.5.6 Switching controllers with TRIGBYCONTACT

Main program

```
 1 DEF SwitchToStiffness()
 2
 3 INI
 4
 5 $STIFFNESS.STRATEGY = 10
 6 $STIFFNESS.COMMIT = TRUE
 7 PTP StartPos CONT Vel= 100% PDAT3 Tool[1]:Test Base[0]
 8 $OV_PRO = 50
 9 $ORI_TYPE = #CONSTANT
10 LIN P1 Vel= 0.5 m/s CPDAT5 Tool[1]:Test Base[0]
11
12 Trig()
13
14 $STIFFNESS.STRATEGY = 10
15 $STIFFNESS.COMMIT = TRUE
16 $OV_PRO = 50
17 LIN_REL {Y -50,Z -40}
18 PTP StartPos Vel= 100% PDAT3 Tool[1]:Test Base[0]
19
20 END
```

Line	Description
5	Robot is under position control.
7	For BCO run to start position Start position can be adapted by teaching if necessary.
10	LIN motion with reduced program override and constant orientation to point P1 just above the surface
12	Subprogram **Trig**: ■ Switching to Cartesian stiffness control with TRIGBYCONTACT ■ Motion in Y direction at surface level
15	Robot is once again under position control.
17	LIN_REL motion to move TCP away from contact

Subprogram Trig

```
21 DEF Trig()
22
23 DECL STIFFNESS USERSTIFF
24
25 DECL INT md_int[16]
26 DECL REAL md_real[16]
27 DECL INT n
28 DECL INT result
29
30 USERSTIFF = $STIFFNESS
31 USERSTIFF.STRATEGY = 20
32 USERSTIFF.FRAMETYPE = #TOOL
33 USERSTIFF.TOOL = {X 0.0,Y 0.0,Z 180.0,A 0.0,B 0.0,C 0.0}
34 USERSTIFF.CPSTIFFNESS = {X 2000.0,Y 2000.0,Z 2000.0,A 1.0,
                            B 200.0,C 200.0}
35 USERSTIFF.CPDAMPING = {X 0.7,Y 0.7,Z 0.7,A 0.7,B 0.7,C 0.7}
36 USERSTIFF.AXISSTIFFNESS = {A1 2000.0,A2 2000.0,A3 2000.0,
                              A4 2000.0,A5 2000.0,A6 2000.0,
                              E1 2000.0}
37 USERSTIFF.AXISDAMPING = {A1 0.7,A2 0.7,A3 0.7,A4 0.7,A5 0.7,
                            A6 0.7,E1 0.7}
38 USERSTIFF.CPMAXDELTA = {X 7.0,Y 7.0,Z 7.0,A 10.0,B 2.0,C 2.0}
39 USERSTIFF.MAXFORCE = {X 70.0,Y 70.0,Z 70.0,A 10.0,B 10.0,C 10.0}
40 USERSTIFF.AXISMAXDELTA = {A1 10.0,A2 10.00,A3 10.0,A4 10.0,
                             A5 10.0,A6 10.0,E1 10.0}
41 USERSTIFF.AXISMAXDELTATRQ = {A1 2.0,A2 2.0,A3 2.0,A4 2.0,
                                A5 2.0,A6 2.0,E1 2.0}
42
43 FOR n=1 TO 16
44  md_int[n] = 0
45  md_real[n] = 0
46 ENDFOR
47
48 md_real[1] = 0.01
49
50 result=MD_CMD("PAPAS","TRIGBYCONTACT",md_int[],md_real[])
51
52 $STIFFNESS = USERSTIFF
53 LIN_REL {Z 40}
54 WAIT SEC 0.1
55 PTP $POS_ACT
56
57 $OV_PRO = 10
58 LIN_REL {Y 50}
59
60 END
```

Line	Description
25 … 28	Declaration of the parameters for switching controller
30 … 37	Parameterization of the controller
31	Controller 20: Cartesian stiffness controller
32	Cartesian stiffness, relative to the TOOL coordinate system
33	Data of the TOOL coordinate system
34	Cartesian stiffness TCP does not yield in the X, Y, Z directions. Motions about angles B and C are not possible. The TCP yields easily about angle A.
35	Cartesian damping (default setting)
36	Axis-specific stiffness
37	Axis-specific damping (default setting)
38	Maximum Cartesian deflection
39	Maximum applied Cartesian force
40	Maximum axis-specific deflection
41	Maximum applied axis torque

Line	Description
43 … 46	All parameters must be set to zero.
48	Parameterization of the TRIGBYCONTACT force threshold
50	Preparation for switching the controller Controller 20 is activated when the torque sensors measure 1% of their maximum torque. The factor for this force threshold was transferred with md_real[1] (see line 48).
53	TCP is moved max. 40 mm in Z direction to make contact with the surface.
55	TCP is kept at the level of the contact point, even if continued motion means that there is no longer any contact.
58	TCP is moved in Y direction.

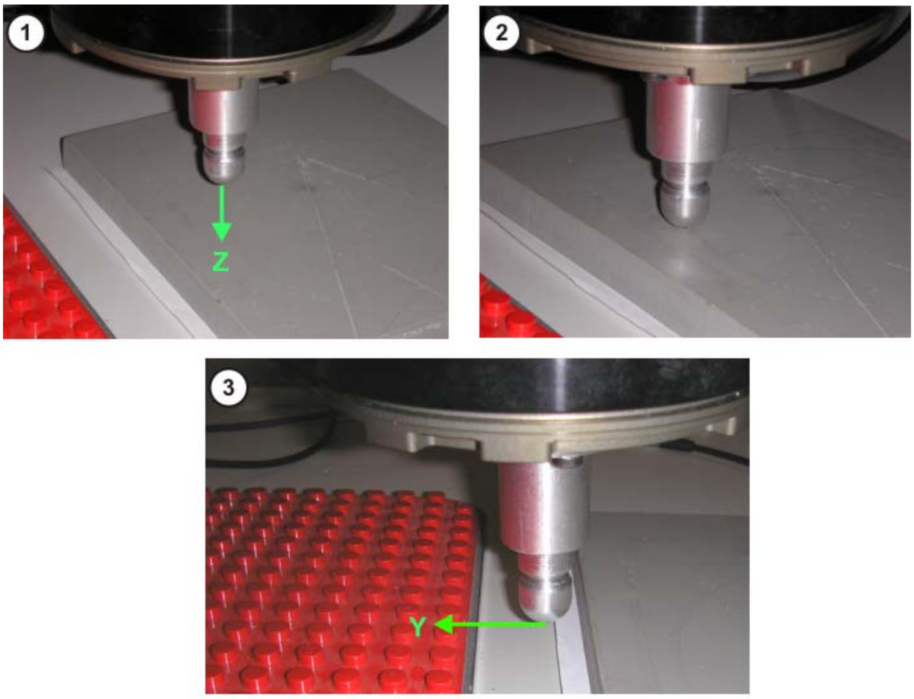


Fig. 6-17: Example: TRIGBYCONTACT

Item	Description
1	TCP is moved in the Z direction from point P1.
2	Contact is made with the surface. When the torque sensors measure the set strength, the robot controller switches to stiffness control.
3	TCP moves at the level of the contact point in the Y direction.

6.5.7 Switching controllers with TRIGBYCONTACT and activating a force

If TRIGBYCONTACT is used to switch from a position controller to a Cartesian stiffness controller, DESIREDFORCE can also be used to activate a setpoint force.

Main program

```
 1 DEF SwitchToStiffness()
 2
 3 INI
 4
 5 $STIFFNESS.STRATEGY = 10
 6 $STIFFNESS.COMMIT = TRUE
 7 PTP StartPos CONT Vel= 100% PDAT3 Tool[1]:Test Base[0]
 8 $OV_PRO = 50
 9 $ORI_TYPE = #CONSTANT
10 LIN P1 Vel= 0.5 m/s CPDAT5 Tool[1]:Test Base[0]
11
12 TrigForce()
13
14 $STIFFNESS.STRATEGY = 10
15 $STIFFNESS.COMMIT = TRUE
16 $OV_PRO = 50
17 LIN_REL {Y -40,Z -20}
18 PTP StartPos Vel= 100% PDAT3 Tool[1]:Test Base[0]
19
20 END
```

Line	Description
5	Robot is under position control.
7	For BCO run to start position Start position can be adapted by teaching if necessary.
10	LIN motion with reduced program override and constant orientation to point P1 just above the surface
12	Subprogram **TrigForce**: ■ Switching to Cartesian stiffness control with TRIGBYCONTACT and activation of a force ■ Motion in Y direction at surface level
15	Robot is once again under position control.
17	LIN_REL motion to move TCP away from contact

Subprogram TrigForce

```
21 DEF TrigForce()
22
23 DECL INT md_int[16]
24 DECL REAL md_real[16]
25 DECL INT n
26 DECL INT result
27
28 FOR n=1 TO 16
29  md_int[n] = 0
30  md_real[n] = 0
31 ENDFOR
32
33 md_real[1] = 0.01
34
35 result=MD_CMD("PAPAS","TRIGBYCONTACT",md_int[],md_real[])
36
37 SoftInZ()
38 ForceZ()
39
40 $OV_PRO = 10
41 LIN_REL {Z 20}
42 LIN_REL {Y 40}
43
44 FOR n=1 TO 16
45  md_int[n] = 0
46  md_real[n] = 0
47 ENDFOR
48
49 result=MD_CMD("PAPAS","DESIREDFORCECLEAR",md_int[],md_real[])
50
51 END
```

Line	Description
23 … 26	Declaration of the parameters for switching controller
28 … 31	All parameters must be set to zero.
33	Parameterization of the TRIGBYCONTACT force threshold
35	Preparation for switching the controller A defined controller is activated when the torque sensors measure 1% of their maximum torque. The factor for this force threshold was transferred with md_real[1] (see line 33).
37	Subprogram **SoftInZ** ■ Switching to Cartesian stiffness control (soft in Z direction)
38	Subprogram **ForceZ**: ■ Activation of a constant force in the Z direction with DESIREDFORCEZ
40	Reduced program override
41	TCP is moved to the contact point. If the TRIGBYCONTACT force threshold is reached, the control mode is switched to Cartesian stiffness control and the force DESIREDFORCEZ is activated.
42	TCP is moved in the Y direction with Cartesian stiffness control. While the TCP is moving, the activated force is exerted constantly in the Z direction.
44 … 47	All parameters must be set to zero.
49	Cancelation of the force activation with DESIREDFORCE-CLEAR

Subprogram SoftInZ

```
52 DEF SoftInZ()
53
54 DECL STIFFNESS USERSTIFF
55
56 USERSTIFF = $STIFFNESS
57 USERSTIFF.STRATEGY = 20
58 USERSTIFF.FRAMETYPE = #TOOL
59 USERSTIFF.CPSTIFFNESS = {X 2000.0,Y 2000.0,Z 1.0,A 200.0,
                            B 200.0,C 200.0}
60 USERSTIFF.CPDAMPING = {X 0.7,Y 0.7,Z 0.7,A 0.7,B 0.7,C 0.7}
61 USERSTIFF.AXISSTIFFNESS = {A1 2000.0,A2 2000.0,A3 2000.0,
                              A4 2000.0,A5 2000.0,A6 2000.0,
                              E1 2000.0}
62 USERSTIFF.AXISDAMPING = {A1 0.7,A2 0.7,A3 0.7,A4 0.7,A5 0.7,
                            A6 0.7,E1 0.7}
63 USERSTIFF.CPMAXDELTA = {X 7.0,Y 7.0,Z 50.0,A 2.0,B 2.0,C 2.0}
64 USERSTIFF.MAXFORCE = {X 70.0,Y 70.0,Z 70.0,A 10.0,B 10.0,C 10.0}
65 USERSTIFF.AXISMAXDELTA = {A1 10.0,A2 10.00,A3 10.0,A4 10.0,
                             A5 10.0,A6 10.0,E1 10.0}
66 USERSTIFF.AXISMAXDELTATRQ = {A1 2.0,A2 2.0,A3 2.0,A4 2.0,
                                A5 2.0,A6 2.0,E1 2.0}
67 $STIFFNESS = USERSTIFF
68
69 END
```

Line	Description
56 … 66	Parameterization of the controller
57	Controller 20: Cartesian stiffness controller
58	Cartesian stiffness, relative to the TOOL coordinate system
59	Cartesian stiffness TCP does not yield in the X and Y directions and is set to soft in the Z direction. Motions about angles A, B and C are not possible.
60	Cartesian damping (default setting)
61	Axis-specific stiffness
62	Axis-specific damping (default setting)
63	Maximum Cartesian deflection
64	Maximum applied Cartesian force
65	Maximum axis-specific deflection
66	Maximum applied axis torque

Subprogram ForceZ

```
72 DEF ForceZ()
73
74 DECL INT md_int[16]
75 DECL REAL md_real[16]
76 DECL INT n
77 DECL INT result
78
79 FOR n=1 TO 16
80  md_int[n] = 0
81  md_real[n] = 0
82 ENDFOR
83
84 md_int[1] = 1
85 md_int[2] = 0
86 md_int[3] = 0
87 md_real[1] = 20.0
88 md_real[2] = 50.0
89 md_real[3] = 100.0
90 md_real[4] = 0.2
91 md_real[5] = 0.1
92 md_real[6] = 0.0
93 md_real[7] = 0.0
94
95 result=MD_CMD("PAPAS","DESIREDFORCEZ",md_int[],md_real[])
96 result=MD_CMD("PAPAS","DESIREDFORCESTART",md_int[],md_real[])
97
98 END
```

Line	Description
74 … 77	Declaration of the parameters for the force activation
79 … 82	All parameters must be set to zero.
84 … 93	Parameterization of DESIREDFORCEZ
84	A constant force is activated in the Z direction.
85	Force must be activated with DESIREDFORCESTART.
87	Constant force in the Z direction: 20 N
88	Duration: 50 s
89	Deflection in Z direction is limited to 100 mm.
90	Velocity in Z direction is limited to 0.2 m/s.
91	Rise and fall time: each 0.1 s
96	Start DESIREDFORCEZ

6.6 Reconfiguring the I/O driver

Precondition

- User group "Expert".
- Operating mode T1 or T2.

⚠ **WARNING** All outputs are reset!

Procedure

1. Select the menu sequence **Configure** > **I/O Driver** > **Reconfigure I/O Driver**.
2. Acknowledge messages.

6.7 Displaying status keys for technology packages

Description

- If one or more technology packages are installed, this menu sequence can be used to display the status keys for a specific technology package.

Procedure

- Select the menu sequence **Configure** > **Status keys** and the relevant technology package.

6.8 Renaming the tool/base

Description

The following menu items are available:

- **Tool type**: For renaming a tool (not a fixed tool!) or workpiece.
- **Base type**: For renaming a base or fixed tool.

Procedure

1. Select the menu sequence **Configure** > **Tool definition** > **Tool type** or **Base type**.
2. Select the tool or base.
3. Press the **Name** softkey.
4. Enter the new name. Confirm with **OK**.
5. Save the changes and close the window with **OK**.

6.9 Configuring the variable overview

This is where the variables to be displayed in the variable overview and the number of groups are defined. A maximum of 10 groups is possible. A maximum of 25 variables per group is possible. System variables and user-defined variables can be displayed.

Precondition

- “Expert” user group

Procedure

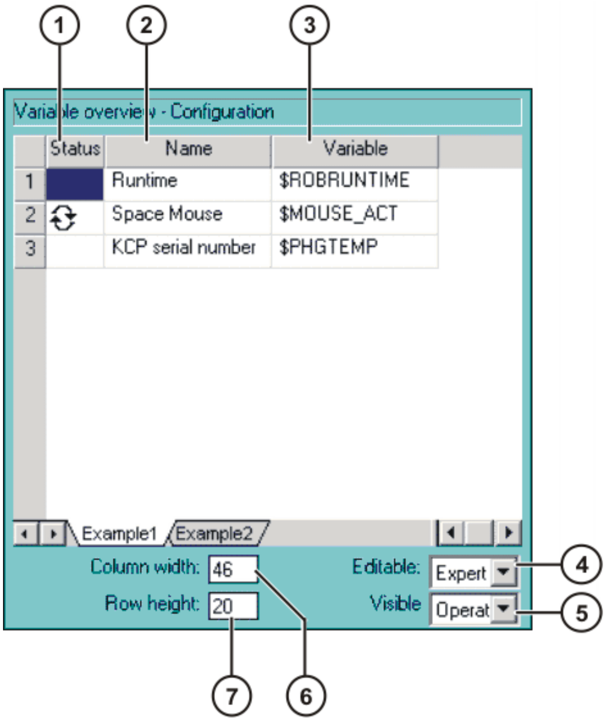
1. Select the menu sequence **Monitor** > **Variable** > **Overview** > **Configuration**.

   The **Variable overview - Configuration** window is opened.
2. Make the desired settings.

> To edit a cell, select it and press the Enter key. Then confirm by pressing the Enter key.

3. Press the **OK** softkey to save the configuration and close the window.

Description



Variable overview - Configuration

	Status	Name	Variable
1	■	Runtime	$ROBRUNTIME
2	⟳	Space Mouse	$MOUSE_ACT
3		KCP serial number	$PHGTEMP

Example1 Example2

Column width: 46

Row height: 20

Editable: Expert

Visible Operat

Fig. 6-18: Variable overview - Configuration window

Item	Description
1	Arrow symbol ⟳: If the value of the variable changes, the display is automatically updated. No arrow symbol: The display is not automatically updated.
2	Descriptive name
3	Path and name of the variable **Note:** For system variables, the name is sufficient. Other variables must be specified as follows: /R1/*Program name*/*Variable name* Do not specify a folder between /R1/ and the program name. Do not add a file extension to the file name.
4	There is one tab per group.
5	Lowest user group in which the variable overview can be modified.
6	Lowest user group in which the variable overview can be displayed.
7	Column width in mm
8	Row height in mm

Column width and row height can also be modified by moving the dividing lines using the mouse.

The following softkeys are available:

Softkey	Description
Display	Switches to the variable overview. (>>> 4.20.7 "Displaying the variable overview and modifying variables" Page 80)
Tab +	Switches to the next group.
Jump	Sets the focus on the next element in the user interface.

Softkey	Description
Paste	Shows additional softkeys: - **R. above**: Inserts a new row above the one currently selected. - **R. below**: Inserts a new row below the one currently selected. - **G. before**: Inserts a new group to the left of the one currently selected. - **G. after**: Inserts a new group to the right of the one currently selected.
Delete	Shows additional softkeys: - **Row**: The selected row is deleted. - **Group**: The current group is deleted.

6.10 Reducing the wait time when shutting down the system

Description

By default, a wait time of 15 seconds is set in the system for when it is shut down. The purpose of the wait time is to ensure that the system does not immediately shut down, for example, in the event of a very sudden, brief power failure, but bridges the power failure for the duration of the wait time.

If the wait time is to be ignored, it can be deactivated for the next shutdown procedure.

Precondition

- User group “Expert”

Procedure

- Select the menu sequence **Configure** > **On/Off Options** > **Disable PowerOff Delay**.

  The next time the system is shut down, the controller shuts down immediately. The wait time is ignored.

6.11 Changing the password

Procedure

1. Select the menu sequence **Configure** > **User group**. The current user group is displayed.
2. Press the **Log On...** softkey.
3. Select the user group whose password you wish to change and confirm with the **Log On...** softkey.
4. Press the **Change Pwd ...** softkey.
5. Enter the old password. Enter the new password twice.

   For security reasons, the entries are displayed encrypted. Upper and lower case are taken into consideration.
6. Press the **OK** softkey. The new password is valid immediately.

6.12 Simulating inputs/outputs

> ⚠ **WARNING** Before using the simulation function, the user must have understood its functional principle. In particular, he must be aware of the states the inputs/outputs revert to when simulation is deactivated and the effects this can have in the specific application.
> Severe physical injuries or considerable damage to property may otherwise result.

Description

Inputs and outputs can be simulated. If, for example, the input periphery is not yet available, the inputs can be set simply by means of simulation. The same principle applies to outputs.

- To simulate an input[x], $INSIM_TBL[x] must be set to the desired simulated state (#SIM_TRUE or #SIM_FALSE).
- To simulate an output[x], $OUTSIM_TBL[x] must be set to the desired simulated state (#SIM_TRUE or #SIM_FALSE).
- Simulation must also be activated. ($IOSIM_OPT =TRUE)

  If simulation is not activated, the real state of all inputs/outputs applies, i.e. $IN[x] and $OUT[x]. The simulated state is then irrelevant.

Some inputs/outputs are write-protected and cannot be simulated. Simulated inputs/outputs are labeled with **SIM** in the KUKA System Software while write-protected ones are labeled with **SYS**.

(>>> 4.20.2 "Displaying digital inputs/outputs" Page 74)

Robot controller response:

- If an output[x] is simulated, its real state (i.e. $OUT[x]) can no longer be modified. $OUTSIM_TBL[x] must first be set to #NONE.
- The robot controller processes both simulated input signals and real input signals. If an output is assigned to an input in the [IOLINKING] section of the file iosys.ini, the simulated input also sets the physical output!
- When simulation is deactivated again:
  - All outputs resume the state they had prior to simulation.
  - All inputs resume their real state.
- When the robot controller is rebooted:
  - Simulation is deactivated.
  - All $INSIM_TBL[x] and all $OUTSIM_TBL[x] are set to #NONE.

System variables

System variable	Description
$IOSIM_OPT	■ **FALSE**: Simulation is deactivated (default). ■ **TRUE**: Simulation is activated.

System variable	Description
$INSIM_TBL[x]	■ **#NONE**: The input is not simulated (default). ■ **#SIM_TRUE**: The input is TRUE (simulated). ■ **#SIM_FALSE**: The input is FALSE (simulated).
$OUTSIM_TBL[x]	■ **#NONE**: The output is not simulated (default). ■ **#SIM_TRUE**: The output is TRUE (simulated). ■ **#SIM_FALSE**: The output is FALSE (simulated).

These system variables are automatically set to FALSE when the robot controller is started.

Precondition

- The entry 'office' in the file hw_inf.ini in the directory C:\KRC\ROBOTER\INIT is set to TRUE.

```
[Version]
office=TRUE
```

- Outputs can only be set if the enabling switch is pressed.

Procedure

1. Select the menu sequence **Monitor** > **Variable** > **Single**. The **Variable Overview - Single** window is opened.
2. Simulate inputs or outputs: set $INSIM_TBL[x] or $OUTSIM_TBL[x] to #SIM_TRUE or #SIM_FALSE.

   Inputs/outputs that are not to be simulated must be set to #NONE.
3. Activate simulation: set $IOSIM_OPT to TRUE.
4. To deactivate simulation, set $IOSIM_OPT to FALSE.
5. If the entry 'office' in the file hw_inf.ini has been specially set to TRUE for simulation: set 'office' back to FALSE.

Example 1

State before simulation: $OUT[8]==FALSE

1. The simulated state of the output is set to TRUE. ($OUTSIM_TBL[8]=#SIM_TRUE)
2. Simulation is activated. ($IOSIM_OPT =TRUE)

   The real state now reflects the simulated state, i.e. $OUT[8]==TRUE.

   $OUT[8] can no longer be modified.
3. Simulation is deactivated. ($IOSIM_OPT =FALSE)

   Now $OUT[8]==FALSE!

Deactivation of the simulation has reset $OUT[8] to the state it had before the simulation, i.e. FALSE. $OUT[8] can now be modified again.

Example 2

State before simulation: $OUT[9]==FALSE.

Furthermore, $OUTSIM_TBL[9]==#NONE, i.e. the output is not simulated.

1. Simulation is activated. ($IOSIM_OPT =TRUE)
2. The real state of the output is changed to TRUE. ($OUT[9]=TRUE)

3. Simulation is deactivated again. ($IOSIM_OPT=FALSE)
   Now $OUT[9]==FALSE!

Deactivation of the simulation has reset $OUT[9] to the state it had before the simulation, i.e. FALSE.

6.13 Configuring workspaces

Workspaces can be configured for a robot. These serve to protect the system.

A maximum of 8 Cartesian (=cubic) and 8 axis-specific workspaces can be configured at any one time. The workspaces can overlap.

(>>> 6.13.1 "Configuring Cartesian workspaces" Page 150)

(>>> 6.13.2 "Configuring axis-specific workspaces" Page 152)

There are 2 types of workspace:

- Non-permitted spaces. The robot may only move outside such a space.
- Permitted spaces. The robot must not move outside such a space.

Exactly what reactions occur when the robot violates a workspace depends on the configuration.

6.13.1 Configuring Cartesian workspaces

> In the case of Cartesian workspaces, only the position of the TCP is monitored. It is not possible to monitor whether other parts of the robot violate the workspace.

Description

The following parameters define the position and size of a Cartesian workspace:

- Origin of the workspace relative to the WORLD coordinate system
- Dimensions of the workspace, starting from the origin

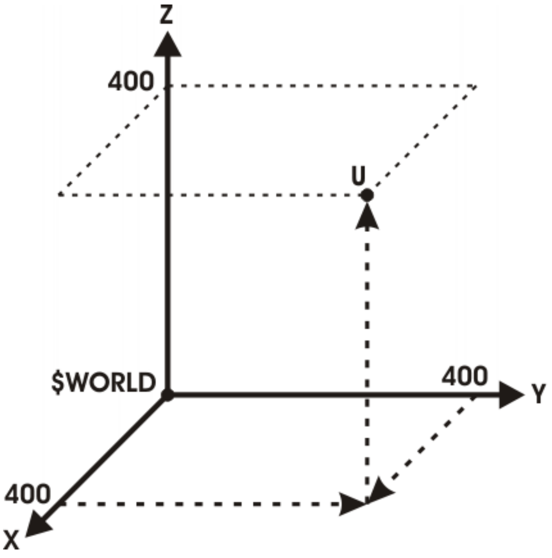


Fig. 6-19: Cartesian workspace, origin U

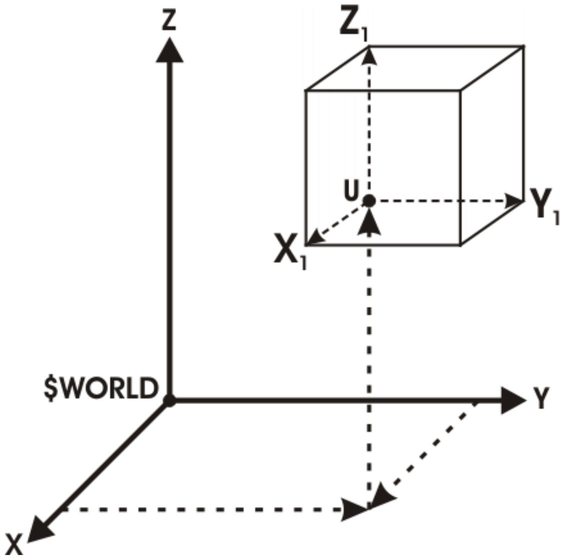


Fig. 6-20: Cartesian workspace, dimensions

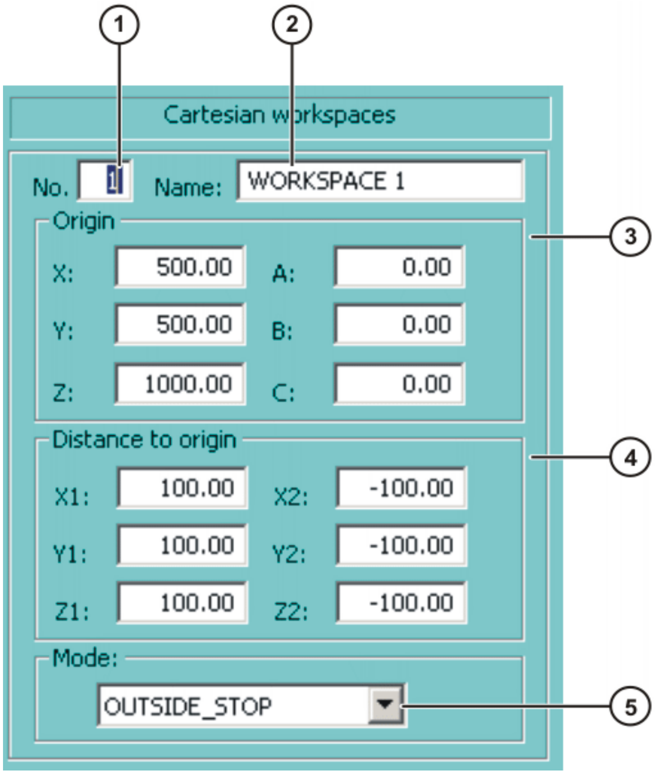
Precondition

- User group "Expert".
- Operating mode T1 or T2.

Procedure

1. Select the menu sequence **Configure** > **Tools** > **Monitoring working envelope** > **Configuration**.

   The **Cartesian workspaces** window is opened.
2. Enter values and save them by pressing the **Apply** softkey.
3. Press the **Signal** softkey. The **Signals** window is opened.
4. In the **Cartesian** column: next to the number of the workspace, enter the output that is to be set if the workspace is violated.
5. Press the **Apply** softkey.
6. Press the **Close** softkey.



Cartesian workspaces

No.: 1 Name: WORKSPACE 1

Origin

X: 500.00 A: 0.00
Y: 500.00 B: 0.00
Z: 1000.00 C: 0.00

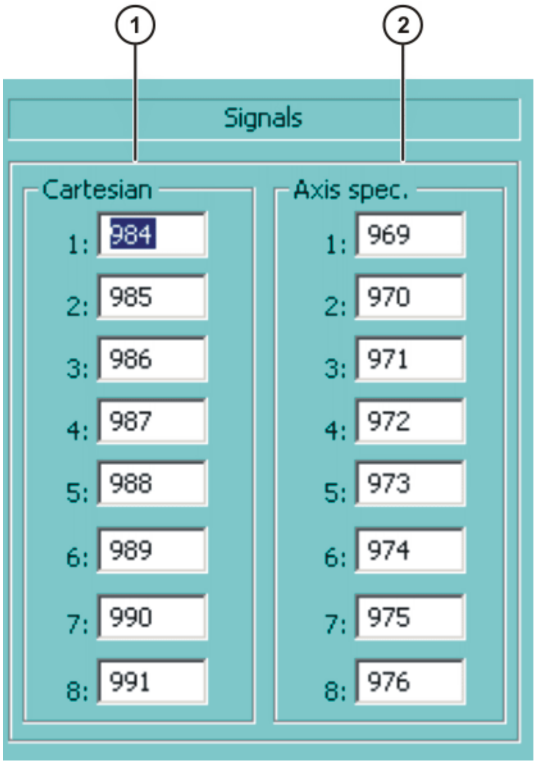
Distance to origin

X1: 100.00 X2: -100.00
Y1: 100.00 Y2: -100.00
Z1: 100.00 Z2: -100.00

Mode: OUTSIDE_STOP

Fig. 6-21: Cartesian workspaces window

Item	Description
1	Number of the workspace (max. 8)
2	Designation of the workspace
3	Origin and orientation of the workspace relative to the WORLD coordinate system
4	Dimensions of the workspace in mm
5	Mode (>>> 6.13.3 "Mode for workspaces" Page 155)



Signals

Cartesian		Axis spec.	
1:	984	1:	969
2:	985	2:	970
3:	986	3:	971
4:	987	4:	972
5:	988	5:	973
6:	989	6:	974
7:	990	7:	975
8:	991	8:	976

Fig. 6-22: Signals window

Item	Description
1	Outputs for the monitoring of the Cartesian workspaces
2	Outputs for the monitoring of the axis-specific workspaces

If no output is to be set when the workspace is violated, the value FALSE must be entered.

6.13.2 Configuring axis-specific workspaces

Description

Axis-specific workspaces can be used to restricted yet further the areas defined by the software limit switches in order to protect the robot, tool or workpiece.

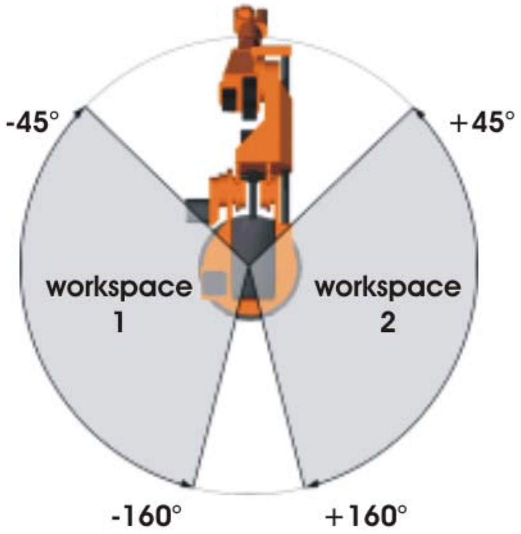


Fig. 6-23: Example of axis-specific workspaces for A1

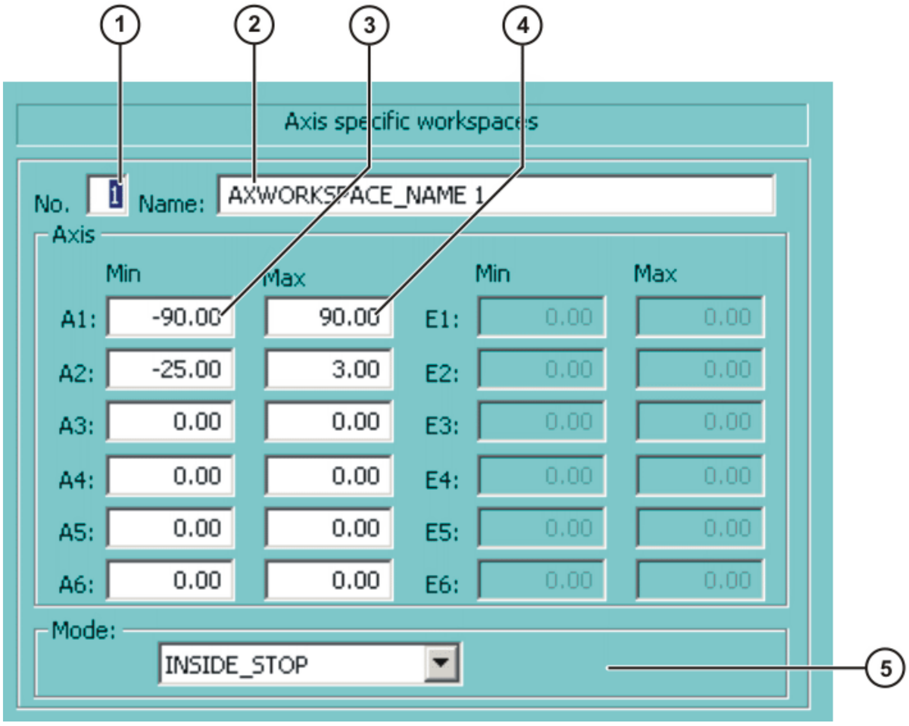
Precondition

- User group "Expert".
- Operating mode T1 or T2.

Procedure

1. Select the menu sequence **Configure** > **Tools** > **Monitoring working envelope** > **Configuration**.

   The **Cartesian workspaces** window is opened.
2. Press the **Axis spec.** softkey to toggle to the **Axis-specific workspaces** window.
3. Enter values and save them by pressing the **Apply** softkey.
4. Press the **Signal** softkey. The **Signals** window is opened.
5. In the **Axis-specific** column: next to the number of the workspace, enter the output that is to be set if the workspace is violated.
6. Press the **Apply** softkey.
7. Press the **Close** softkey.



Axis specific workspaces

No. 1 Name: AXWORKSPACE_NAME 1

Axis

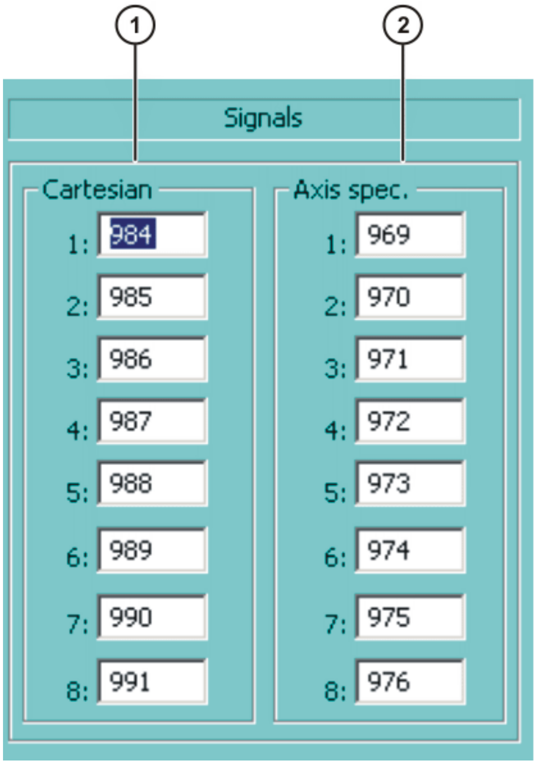
	Min	Max		Min	Max
A1:	-90.00	90.00	E1:	0.00	0.00
A2:	-25.00	3.00	E2:	0.00	0.00
A3:	0.00	0.00	E3:	0.00	0.00
A4:	0.00	0.00	E4:	0.00	0.00
A5:	0.00	0.00	E5:	0.00	0.00
A6:	0.00	0.00	E6:	0.00	0.00

Mode: INSIDE_STOP

Fig. 6-24: Axis-specific workspaces window

Item	Description
1	Number of the workspace (max. 8)
2	Designation of the workspace
3	Lower limit for axis angle
4	Upper limit for axis angle
5	Mode (>>> 6.13.3 "Mode for workspaces" Page 155)

If the value 0 is entered for an axis under Item 3 and Item 4, the axis is not monitored, irrespective of the mode.



	Cartesian		Axis spec.
1:	984	1:	969
2:	985	2:	970
3:	986	3:	971
4:	987	4:	972
5:	988	5:	973
6:	989	6:	974
7:	990	7:	975
8:	991	8:	976

Fig. 6-25: Signals window

Item	Description
1	Outputs for the monitoring of the Cartesian workspaces
2	Outputs for the monitoring of the axis-specific workspaces

If no output is to be set when the workspace is violated, the value FALSE must be entered.

6.13.3 Mode for workspaces

Mode	Description
#OFF	Workspace monitoring is deactivated.
#INSIDE	■ Cartesian workspace: The defined output is set if the TCP is located inside the workspace. ■ Axis-specific workspace: The defined output is set if the axis is located inside the workspace.
#OUTSIDE	■ Cartesian workspace: The defined output is set if the TCP is located outside the workspace. ■ Axis-specific workspace: The defined output is set if the axis is located outside the workspace.

Mode	Description
#INSIDE_STOP	- Cartesian workspace: The defined output is set if the TCP is located inside the workspace. - Axis-specific workspace: The defined output is set if the axis is located inside the workspace. The robot is also stopped and a message is displayed. The robot cannot be moved again until the workspace monitoring is deactivated or bypassed. (>>> 4.19 "Bypassing workspace monitoring" Page 73)
#OUTSIDE_STOP	- Cartesian workspace: The defined output is set if the TCP is located outside the workspace. - Axis-specific workspace: The defined output is set if the axis is located outside the workspace. The robot is also stopped and a message is displayed. The robot cannot be moved again until the workspace monitoring is deactivated or bypassed. (>>> 4.19 "Bypassing workspace monitoring" Page 73)

6.14 Refreshing the user interface

Description

This function can be used to refresh the user interface, e.g. to display status keys created by the user.

The graphical user interface is reinitialized without rebooting the system. The progress of the reinitialization is indicated in the message window.

Precondition

- “Expert” user group

Procedure

- Select the menu sequence **Configure** > **Tools** > **Reinit** > **BOF Reinitialization**.

6.15 Optimizing the cycle time

This function can be used to modify the maximum permissible accelerations for different technologies.

By default, the accelerations are set so as to trigger the robot controller monitoring functions as rarely as possible. The higher the maximum acceleration, the more likely the monitoring functions are to be triggered.

Precondition

- “Expert” user group

Procedure

1. Select the menu sequence **Configure** > **Tools** > **Cycle Time Optimizer**.
2. Select the desired technology in the **Application** box.
3. Modify the acceleration using the status key.
4. Press the **Apply** softkey.

Description

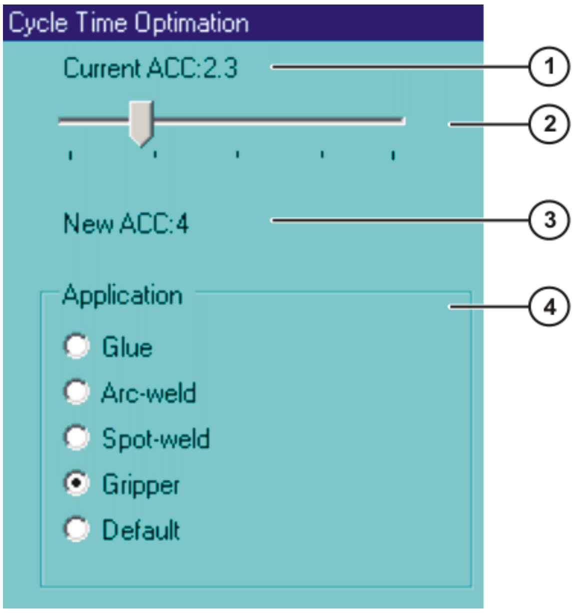


Fig. 6-26: Cycle Time Optimation window

Item	Description
1	Current maximum acceleration. The value is saved in the variable DEF_ACC_CP.
2	Slider control for modifying the acceleration.
3	Value to which the slider control is currently set.
4	The entries offered depend on the technology package being used.

6.16 Defining limits for point correction

Overview

If this function is active, existing points may only be corrected within certain limits in the user groups “User” and “Operator”. If the limits are exceeded, a message is displayed, indicating that the correction has been rejected. Global positions, e.g. the HOME position, can no longer be corrected at all in these user groups.

In the user group “Expert”, points can still be corrected without restrictions. If the function is active and points are corrected in the user group “Expert”, the master point is updated in the external editor, if present.

> The correction limits only apply to changes made using the softkeys **Change** and **TouchUp** or **Change** and **Cmd OK**.
> Corrections made using different functions, e.g. calibration or variable correction, remain possible for the user groups “User” and “Operator”. If these functions are also to be restricted, this must be configured in the files SoftKeyUser.ini and MenueKeyUser.ini.

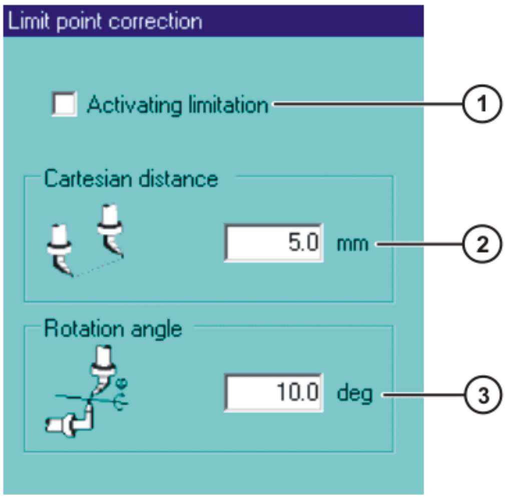
Precondition

- “Expert” user group

Procedure

1. Select the menu sequence **Configure** > **Tools** > **Limit point correction**.
2. Enter the desired values and activate correction limit.
3. Press the **OK** softkey. The entries are saved and the option window is closed.

Description



Limit point correction

□ Activating limitation

Cartesian distance: 5.0 mm

Rotation angle: 10.0 deg

Fig. 6-27: Option window “Limit point correction”

Item	Description
1	Check box activated: the correction limit is active. Check box not active: the correction limit is inactive.
2	Maximum permissible correction for the X, Y and Z values The value defines the radius of a sphere about the original point (or about the master point in the external editor, if present).
3	Maximum permissible correction for the A, B and C values Value 0: a correction is not allowed.

The following softkeys are available:

Softkey	Description
Correction	Activates or deactivates the check box **Activating limitation**.

6.17 Configuring inline forms for motions

6.17.1 Indicating the approximation distance for motion commands

Description

Inline commands for approximated motions can be indicated in the program with or without the approximation distance.

Example without approximation distance:

```
LIN P8 CONT Vel= 100 % CPDAT9 Tool[1] Base[0]
```

Example with approximation distance:

```
LIN P8 CONT = 50 mm Vel= 100 % CPDAT9 Tool[1] Base[0]
```

If the display is changed, this applies for all subsequently programmed commands. The change is not applied to existing commands unless these are also changed.

Precondition

- If the display is changed with the KSS running:
  User group “Expert”

Procedure

1. Open the directory C:\KRC\UTIL\REGUTIL.
2. Double-click on the desired file.

3. Answer the request for confirmation with **Yes**. Then confirm the message indicating that the change has been made with **OK**.
4. If the change is made with the KSS running:

   Select the menu sequence **Configure** > **Tools** > **Reinit** > **BOF Reinitialization**.

File

File	Description
ContStatingOn.reg	The approximation distance is displayed in inline commands for approximated motions.
ContStatingOff.reg	The approximation distance is not displayed in inline commands for approximated motions.

6.17.2 Changing the unit of the approximation distance for PTP

The approximation distance in the option window **Motion parameter** (PTP) can be specified as a percentage or in mm. The default setting is percent.

If the unit is changed, this applies for all subsequently programmed commands. The original unit is retained for existing commands.

Exception: If an existing command is changed (e.g. the point name), the new unit is automatically applied for this command. The value remains the same, however! This causes the path to change.

Example:

1. Existing command with approximation distance 88%.
2. The unit is changed from percent to mm.
3. The existing command is changed (e.g. the name of the point).

   The change to the unit thus takes effect. The approximation distance is now 88 mm.

> ⚠ **WARNING** In this case, the system generates a message prompting the user to check the new path. This message must be acknowledged.
> The new path must be tested in jog mode and adapted as required before returning to program mode. Physical injuries or damage to property may otherwise result.

Precondition

- If the unit is changed with the KSS running:

  “Expert” user group

Procedure

1. Open the directory C:\KRC\UTIL\REGUTIL.
2. Double-click on the desired file.
3. Answer the request for confirmation with **Yes**. Then confirm the message indicating that the change has been made with **OK**.
4. If the change is made with the KSS running:

   Select the menu sequence **Configure** > **Tools** > **Reinit** > **BOF Reinitialization**.

Description

File	Description
DistCriterionPTPOn.reg	Unit of the approximation distance in the option window **Motion parameter** (PTP): mm
DistCriterionPTPOff.reg	Unit of the approximation distance in the option window **Motion parameter** (PTP): %

6.18 Defining calibration tolerances

Only modify the default values in exceptional cases. Otherwise, increased error messages and inaccuracy may result.

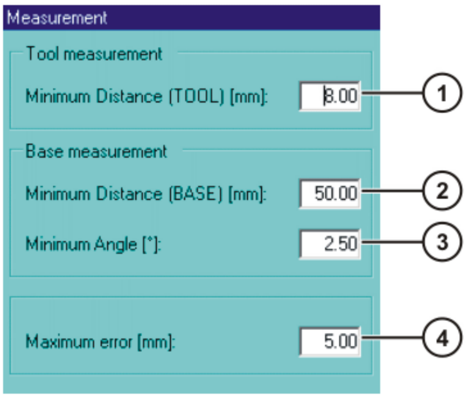
Precondition

- "Expert" user group

Procedure

- Select the menu sequence **Setup** > **Measure** > **Tolerances**.

Description



Measurement

Tool measurement

Minimum Distance (TOOL) [mm]: 8.00

Base measurement

Minimum Distance (BASE) [mm]: 50.00

Minimum Angle [°]: 2.50

Maximum error [mm]: 5.00

Fig. 6-28: Error tolerances

Item	Description
1	The minimum distance for tool calibration. ■ **0 … 200 mm**
2	The minimum distance for base calibration. ■ **0 … 200 mm**
3	The minimum angle between the straight lines through the 3 calibration points in base calibration. ■ **0 … 360°**
4	Maximum error in calculation. ■ **0 … 200 mm**

The following softkeys are available:

Softkey	Description
Default	Restores the default settings. The data must then be saved by pressing the softkey **OK**.

6.19 Backward motion

There are 2 methods for executing a program backwards:

- TRACE (>>> 6.19.1 "TRACE method" Page 161)
- SCAN (>>> 6.19.2 "SCAN method" Page 162)

TRACE method and SCAN method can be activated simultaneously. During backward motion, in this case, the TRACE recordings are taken into consideration first. Once all TRACE recordings have been taken into consideration, program execution continues with SCAN.

All interrupts are deactivated during backward motion. The updating of cyclical flags is also deactivated.

6.19.1 TRACE method

Description

With the TRACE method, the forward motions of the robot are recorded. During backward motion, the recorded motions are executed in the reverse order. Unlike with the SCAN method, the program is not interpreted backwards.

Advantages

- Outputs, flags and cyclical flags can be recorded at every point in forward motion. Their states can be restored for every point that is addressed backwards. All outputs, flags and cyclical flags that are set by TRIGGER are also correctly restored in this case.

  “Outputs” here refers to user outputs. System outputs (e.g. $ALARM_STOP or $T2) are not affected by backward motion.
- Branches and loops can also be executed backwards.

Disadvantages

- A program must first be executed forwards before backward motion is possible.
- The recording is deleted if the program is modified or reset or in the case of block selection. Backward motion is no longer possible in this case.

Response in the case of subprograms

- If a subprogram has been completely executed during forward motion, it cannot be executed with backward motion. Depending on the configuration of **Finished_Sub** (>>> 6.19.4 "TRACE section" Page 163), the subprogram is either skipped during backward motion or the backward motion is stopped.
- If the forward motion was stopped in a subprogram, the response depends on the position of the advance run pointer:

Position of the advance run pointer	Response
Advance run pointer is in the subprogram.	Backward motion is possible.
Advance run pointer has already left the subprogram.	Backward motion is not possible. Prevention: Trigger an advance run stop before the END of the subprogram, e.g. with WAIT SEC 0. However, it is then no longer possible to carry out approximate positioning at this point. Or set $ADVANCE to "1". This does not always prevent the error message, but it reduces the probability. Approximate positioning is still possible.

- Hidden subprograms are skipped during backward motion.

Example

Example of a TRACE recording. Backward motions of the robot are not recorded!

1. Forward motion:

   P1 -> P2 -> P3 -> P4

   The position of the robot is now P4. Points P1 to P3 were recorded during the forward motion. (Only points that the robot has left are recorded; this is why P4 was not recorded.)
2. Backward motion:

P2 <- P3 <- P4

The position of the robot is now P2. Points P3 and P2 have been deleted from the recording. P1 is still present.

3. Forward motion:

P2 -> P3 -> P4 -> P5

The position of the robot is now P5. Points P2 to P4 were recorded during the forward motion. All in all, points P1 to P4 are now recorded.

6.19.2 SCAN method

In the SCAN method, the program is interpreted backwards from the current position of the program interpreter.

Advantage

- Backward motion is still possible after a program modification or block selection.

Disadvantages

- Outputs, flags and cyclical flags cannot be restored.
- Relative motions cannot be executed backwards.
- If the program interpreter encounters a branch or loop, backward interpretation cannot be continued. This is because the system no longer knows which part of the branch to jump to or how often the loop is to be executed.

Response in the case of subprograms

- If a subprogram has been completely executed during forward motion, it is skipped during backward motion.
- If the forward motion was stopped in a subprogram, then backward motion can be executed as far as the start of the subprogram.

6.19.3 Configuring backward motion

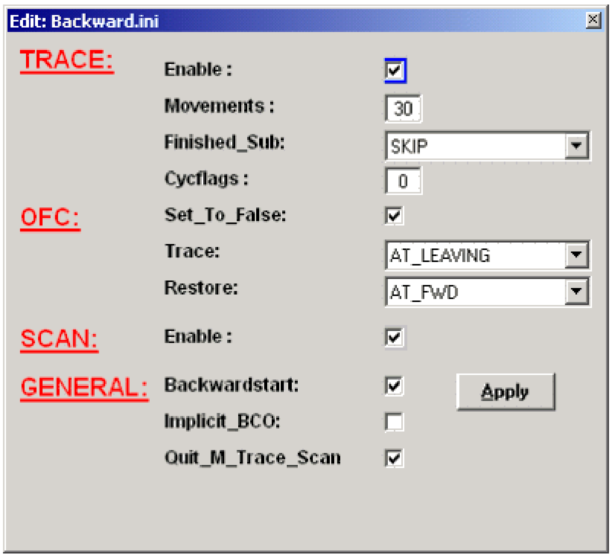
Procedure

1. Double-click on the file BW_INI.EXE in the directory C:\KRC\UTIL.

   The **Edit: Backward.ini** window is opened.
2. Make the desired settings and save them by pressing **Apply**. Close the window.

> **Apply** saves the settings in the file "C:\KRC\ROBOTER\INIT\Backward.Ini". The settings are only saved, however, if the system is not currently in backward mode and no program is active.

Description



Edit: Backward.ini

TRACE:
- Enable : ☑
- Movements : 30
- Finished_Sub: SKIP
- Cycflags : 0

OFC:
- Set_To_False: ☑
- Trace: AT_LEAVING
- Restore: AT_FWD

SCAN:
- Enable : ☑

GENERAL:
- Backwardstart: ☑
- Implicit_BCO: □
- Quit_M_Trace_Scan ☑

Fig. 6-29: Edit: Backward.ini window

- **TRACE** section (>>> 6.19.4 "TRACE section" Page 163)
- **OFC** section (>>> 6.19.5 "OFC section" Page 163)
- **SCAN** section (>>> 6.19.6 "SCAN section" Page 164)
- **GENERAL** section (>>> 6.19.7 "GENERAL section" Page 165)

6.19.4 TRACE section

Parameter	Description
Enable	■ **TRUE**: TRACE method activated, i.e. the forward motions of the robot are recorded. ■ **FALSE**: TRACE method deactivated, i.e. the forward motions are not recorded.
Movements	Maximum number of motions to be recorded. ■ **0 … 60**
Finished_Sub	■ **SKIP**: If a subprogram is reached, a message is displayed which must be acknowledged. The subprogram is then skipped and the backward motion is continued. ■ **STOP**: If a subprogram is reached, further backward motion is not possible.
Cycflags	Maximum number of cyclical flags to be recorded per motion. ■ **0 … 26**

6.19.5 OFC section

OFC stands for: outputs, flags, cyclical flags.

All parameters in the OFC section refer to the TRACE method. "Outputs" here refers to user outputs. System outputs (e.g. $ALARM_STOP or $T2) are not affected by backward motion.

Parameter	Description
Set_To_False	■ **TRUE**: When switching from forward to backward motion, all outputs, flags and cyclical flags are set to FALSE. ■ **FALSE**: When switching from forward to backward motion, all outputs, flags and cyclical flags retain their values.
Trace	■ **At_Leaving**: The assignment of the outputs, flags and cyclical flags is recorded on leaving a programmed point. ■ **No_Trace**: The assignment of the outputs, flags and cyclical flags is not recorded.
Restore	■ **At_Bwd**: When a point is reached during backward motion, the outputs, flags and cyclical flags are restored according to the trace. ■ **At_Fwd**: When switching from backward to forward motion, the outputs, flags and cyclical flags are restored according to the trace.

6.19.6 SCAN section

Parameter	Description
Enable	■ **TRUE**: SCAN method activated ■ **FALSE**: SCAN method deactivated

6.19.7 GENERAL section

Parameter	Description
Backwardstart	**Backwardstart** is only relevant for the SCAN method. With the TRACE method, all parameters for tools, workpieces, etc., are automatically taken into consideration. ■ **TRUE**: The parameters for tool, workpiece, etc., are taken into consideration during backward motion. There must be a Fold with a defined structure for every point that is to be interpreted backwards (see following example for a freely-selected point P6). ■ **FALSE**: The parameters for tool, workpiece, etc., are not taken into consideration during backward motion. In other words, in the case of a tool change between two points, the backward motion may be different from what the forward motion would have been.
Implicit_BCO	■ **TRUE**: If the robot is not on the programmed path, the next motion is a BCO run, irrespective of whether START plus or START minus was pressed. ■ **FALSE**: If the robot is not on the programmed path, a BCO run with the START plus key is required. START minus generates an error message.
Quit_M_Trace_scan	If both TRACE and SCAN methods are activated, the following message is displayed when passing from TRACE to SCAN during backward motion: “Trace buffer empty, start with backward scan”. **Quit_M_Trace_Scan** defines the message type. ■ **TRUE**: Message is an acknowledgement message (message no. 1055). ■ **FALSE**: Message is a notification message (message no. 1194).

Example of the structure of the Fold for **Backwardstart**:

```
;FOLD PTP P6 CONT Vel= 30 % PDAT6 Tool[1]:tool_1 Base[0];%{PE}%R
5.2.17,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:P6, 3:C_PTP, 5:30,
7:PDAT6

$BWDSTART = FALSE

PDAT_ACT=PPDAT6

FDAT_ACT=FP6

BAS(#PTP_PARAMS,30)

PTP XP6 C_PTP

;ENDFOLD
```

The first instruction after the start of the Fold must be $BWDSTART. It is irrelevant whether the $BWDSTART line is set to TRUE or FALSE.

During backward interpretation, the interpreter finds the motion first (in the example: PTP XP6). It then searches further as far as the $BWDSTART line. When it finds it, it takes all the parameter modifications into consideration before planning the backward motion.

In programs created in the user group “User”, all points automatically have the required Folds.
In programs created in the user group “Expert”, the Folds must be created manually as required.
In a program that does not contain such Folds, the parameters for tool, workpiece, etc., cannot be taken into consideration. Tip: In this case, set **Backwardstart** to FALSE to prevent an error message from being displayed for every point.

6.20 Configuring the log-in

Procedure

1. Open the file C:\KRC\HMI\Config\Authentication.config.
2. Make the desired changes.
3. Save and close the file.

Description

Depending on the release, the file Authentication.config may vary slightly from the example given here.

```
<configuration>
  <authenticationManagement>
     <AuthorizationConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <AutoLock>ModeOP2Aut ModeOP2Ext</AutoLock>
        <CanLockSystem>false</CanLockSystem>
        <MethodList>
           <AuthenticationMethod>
              <AuthenticationType>SelectionOnly</AuthenticationType>
              <UserListName>secondaryUserList</UserListName>
              <LeaseTime>300</LeaseTime>
           </AuthenticationMethod>
           <AuthenticationMethod>
              <AuthenticationType>UsernameOnly</AuthenticationType>
              <LeaseTime>300</LeaseTime>
          </AuthenticationMethod>
        </MethodList>
        <DefaultUser>
           <Name>AutoKrcOperator</Name>
           <UserLevel>5</UserLevel>
           <DisplayName>KrcSecurity#Operator</DisplayName>
           <TranslateDisplayName>true</TranslateDisplayName>
           <Password>kukaAuto</Password>
           <UsePasswordAutomaticaly>true</UsePasswordAutomaticaly>
        </DefaultUser>
        <AnnouncementTime>300</AnnouncementTime>
     </AuthorizationConfiguration>
  </authenticationManagement>
   <secondaryUserList>
      <UserList xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
         <UserInfo>
            <Name>AutoKrcProgrammer</Name>
            <UserLevel>10</UserLevel>
            <DisplayName>KrcSecurity#Programmer</DisplayName>
            <TranslateDisplayName>true</TranslateDisplayName>
            <Password>kukaAuto</Password>
            <UsePasswordAutomaticaly>true</UsePasswordAutomaticaly>
         </UserInfo>
         <UserInfo>
            <Name>KrcExpertProgrammer</Name>
            <UserLevel>20</UserLevel>
            <DisplayName>KrcSecurity#ExpertProgrammer</DisplayName>
            <TranslateDisplayName>true</TranslateDisplayName>
         </UserInfo>
         <UserInfo>
            <Name>KrcAdministrator</Name>
            <UserLevel>30</UserLevel>
            <DisplayName>KrcSecurity#KrcAdministrator</DisplayName>
            <TranslateDisplayName>true</TranslateDisplayName>
         </UserInfo>
      </UserList>
   </secondaryUserList>
</configuration>
```

Line	Description	Range of values

4	`AutoLock`
	If the mode is switched to that specified here, the robot controller switches to the default user group.
	If both modes are specified, they must be separated by a space.
		ModeOP2Aut = Automatic mode
		ModeOP2Ext = Automatic External mode

5	`CanLockSystem` TRUE: When the LeaseTime elapses, the robot controller is locked, i.e. it is disabled for all actions except a new log-in (the robot controller is not locked in the default user group). FALSE: When the LeaseTime elapses, the robot controller switches to the default user group.	TRUE, FALSE
6	`MethodList` The MethodList contains all log-in methods. The robot controller always attempts to use the method that is at the top of the list. If this fails, the robot controller attempts to use the next method, etc. The attempt could fail, for example, if the method at the top of the list is External, but the PLC signal required for this method is not received.	External SelectionOnly UsernameOnly Other methods according to customer-specific settings
8, 13	`AuthenticationType` Name of the log-in method See the table “Log-in methods”	External SelectionOnly UsernameOnly Other methods according to customer-specific settings
9	`UserListName` User list that is used. The place where the list is situated and the type of encryption used are defined in the following file: C:\KRC\HMI\KUKA_HMI.exe.config **Note:** When accessing a user list, the system always refers to KUKA_HMI.exe.config. Even if the user list is stored directly in Authentication.config, the system first checks the location in KUKA_HMI.exe.config and then accesses the location.	secondaryUserList primaryUserList Other lists according to customer-specific settings
10, 14	`LeaseTime` If no actions are carried out in the user interface during this time, the robot controller reacts as configured in `CanLockSystem`.	s (INT)

17, 18	`DefaultUser`
	`Name`
	Default user group.
		See the table “User groups”

Line	Description	Range of values

17, 19	`DefaultUser`
	…
	`UserLevel`
	Identification number of the user group
		See the table “User groups”

25	`AnnouncementTime` Only relevant if an external log-in is used. Time available to the user to confirm the external log-in, e.g. via Euchner key, so that it is actually carried out. If the value is set to 0, the external log-in is always active.	0 … 300 s (INT)

Log-in methods

Method	Description

External	Log-in method for log-in via a higher-level controller
	The robot controller must receive a “life signal” from the higher-level controller that toggles with a cycle rate of 500 ms. A digital input indicates to the robot controller which user is logging in.
	The inputs are configured in the file $config.dat.
	■ Signal declaration for life signal:
	`SIGNAL ExtUserIDWatchDog $IN[1026]`
	■ Signal declaration for user log-in:
	`SIGNAL ExtUserID $IN[1026] TO $IN[1026]`
	A user list that only contains automatic Windows user names must be used for this log-in method, as no password is to be requested when logging on via a higher-level controller.

SelectionOnly	If this log-in method is used, the default user group is automatically selected. When changing user group, a list is displayed in the KUKA.HMI from which the new user group can be selected. The list corresponds to the user list assigned to this log-in method.
UsernameOnly	If this log-in method is used, the user must enter the user name, password and domain.

User groups

The Windows user names must be used, with unchangeable passwords, if the request to enter a password is to be avoided when logging on.

User group de/en (identification number)	Windows user name (default password: kuka)	Windows user name (internal password, unchangeable: kuka-Auto)	Key
Bediener / Operator (5)	KrcOperator	AutoOperator	Operator
Anwender / Programmer (User) (10)	KrcProgrammer	AutoKrcProgrammer	Programmer
Inbetriebnehmer / Maintenance (19)	KrcMaintenance	AutoKrcMaintenance	Maintenance
Experte / Expert (20)	KrcExpertProgrammer	AutoKrcExpertProgrammer	ExpertProgrammer

User group de/en (identification number)	Windows user name (default password: kuka)	Windows user name (internal password, unchangeable: kuka-Auto)	Key
Sicherheitsinstandhalter / Safety Recovery (27)	KrcSafetyRecovery	AutoKrcSafetyRec	SafetyRecovery
Sicherheitsinbetriebnehmer / Safety Maintenance (29)	KrcSafetyMaintenance	AutoKrcSafetyMaint	SafetyMaintenance
Administrator / Administrator (30)	KrcAdministrator	AutoKrcAdministrator	KrcAdministrator

6.21 Setting up a new user group and password

Description

In addition to the predefined user groups, other user groups can also be set up.

Depending on the user group, different menu items and softkeys are available in the user interface. This is defined using identification numbers: Each user group is assigned an identification number between 0 and 30. A menu item or softkey is available in a user group if its identification number is less than or equal to the identification number of the user group.

Identification numbers of the predefined user groups

(>>> "User groups" Page 169)

The identification numbers of the menu items and softkeys are defined in the following files:

- Menu keys: C:\KRC\ROBOTER\INIT\MenueKeyKuka.ini
- Softkeys: C:\KRC\ROBOTER\INIT\SoftKeyKuka.ini

If a menu item or softkey has no identification number, it is available in the user group “Operator” or higher.

Overview

Step	Description
1	Set up new user group. (>>> 6.21.1 "Example of setting up a new user group" Page 170)
2	Set up password for new user group. (>>> 6.21.2 "Setting up a password for a new user group" Page 172)
3	Define new user group as default user group (optional). (>>> 6.22 "Defining the default user group" Page 173)

6.21.1 Example of setting up a new user group

Description

In this section, the user group “MyGroup” will be set up by way of an example.

The user group “MyGroup” is to have the same functions as the user group “User” (“Programmer”) plus the function “Monitoring working envelope - Override”. This function is called as follows in the user interface: menu sequence **Configure** > **Tools** > **Monitoring working envelope** > **Override**.

Overview

Step	Description
1	Define user group. (>>> 6.21.1.1 "Defining a user group" Page 171)
2	Define position of the softkey. (>>> 6.21.1.2 "Defining the position of the softkey" Page 171)
3	Enable function. (>>> 6.21.1.3 "Enabling the function" Page 172)

6.21.1.1 Defining a user group

Procedure

1. In the file SoftKeyKuka.ini, go to the section UserMode-OCX.

```
;*****  U s e r M o d e - O C X  *****
USER_UMODE   = UserModeUser  , 20, USERMODE, 10
EXPERT_UMODE = UserModeExpert, 20, USERMODE, 20
ADMIN_UMODE  = UserModeAdmin , 20, USERMODE, 30
```

2. Insert the new user group after the line ADMIN_UMODE.

```
;*****  U s e r M o d e - O C X  *****
USER_UMODE   = UserModeUser  , 20, USERMODE, 10
EXPERT_UMODE = UserModeExpert, 20, USERMODE, 20
ADMIN_UMODE  = UserModeAdmin , 20, USERMODE, 30
MYGROUP_UMODE  = MyGroup , 20, USERMODE, 11
```

The optional elements are indicated in bold type. All other elements must be entered unchanged.

Element	Description
MYGROUP_UMODE	Internal system designation. This designation must end with _UMODE.
MyGroup	Designation of the softkey in the user interface. Freely selectable
11	Identification number ■ **0 … 28**, except **5, 10, 19, 20, 27**! (These are the identification numbers of the predefined user groups. (>>> "User groups" Page 169))

6.21.1.2 Defining the position of the softkey

Procedure

1. In the file SoftKeyKuka.ini, go to the section [#USERMODE].

```
[#USERMODE]
UserGroup = USER_UMODE, EXPERT_UMODE, ADMIN_UMODE, , , , CANCEL_DISP
```

The entry contains the internal system designation of the softkeys. The order corresponds to the sequence in the softkey bar.

2. Enter the internal system designation in the desired position.

```
[#USERMODE]
UserGroup = USER_UMODE, EXPERT_UMODE, ADMIN_UMODE, MYGROUP_UMODE, ,
, CANCEL_DISP
```

Do not delete or overwrite any commas!

3. Save and close the file “SoftKeyKuka.ini”.

6.21.1.3 Enabling the function

Procedure

1. In the file MenueKeyKuka.ini, go to the entry for the menu item **Override**.

```
; ************** Konfigurieren / Extras / Arbeitsraum-Menu *****
miDisableWBox     = ConfigWBoxDisable, 165, KrcRobotLogic,
$WBOXDISABLE;TRUE,   , , 20
```

2. Change the identification number from 20 to 11.

```
; ************** Konfigurieren / Extras / Arbeitsraum-Menu *****
miDisableWBox     = ConfigWBoxDisable, 165, KrcRobotLogic,
$WBOXDISABLE;TRUE,   , , 11
```

   The menu item **Override** is now available in user groups whose identification number ≥ 11.

> The identification number must always be the 7th element in the enumerated list to the right of the equals sign. The elements are separated by commas. Semicolons are not valid as separators.
> If a menu item does not yet have an identification number, it may be necessary to add not only an identification number, but also the corresponding commas.

3. **Override** is a sub-item of the menu item **Monitoring working envelope**. This must also be made available for the new user group, as the menu item **Override** is not otherwise accessible.

   In the file MenueKeyKuka.ini, go to the entry for the menu item **Monitoring working envelope**. Change the identification number from 20 to 11.

```
; ************** Konfigurieren / Extras-Menu *****
…
mpWorkspace      = Workspace, 1035, HMI, ,POPUP, mWorkspace, 11
```

4. Save and close the file MenueKeyKuka.ini.

6.21.2 Setting up a password for a new user group

Precondition

- A new user group has been set up.
- Windows interface (CTRL+ESC)

Procedure

1. Press the Windows **Start** button and select **Run...**.
2. In the **Open** box, enter “regedit” and press **OK**. The **Registry editor** window opens.
3. Select the following folder in the tree structure:

   HKEY_CURRENT_USER\Software\KUKA Roboter GmbH\KUKA BOF\OCX Controls\UserMode\PassWord
4. Select the menu sequence **Edit** > **New** > **Binary value**.
5. Change the designation of the new value to “PassWordxx”. Instead of “xx”, enter the identification number of the new user group.
6. Close the **Registry editor** window.

   The password now consists of a blank character string. It can be left like this or modified.

   - If the blank character string is to be left: simply press the Enter key when the password is requested.
   - If the password is to be changed: (>>> 6.11 "Changing the password" Page 147)

By default, the values entered under **HKEY_CURRENT_USER\...** in the registry database are not overwritten when the KSS is installed.

6.22 Defining the default user group

Description

When the system is booted, the user group “Operator” is activated by default. An alternative user group can be defined as the default user group instead.

Procedure

1. Press the Windows **Start** button and select **Run...**.
2. In the **Open** box, enter “regedit” and press **OK**. The **Registry editor** window opens.
3. Select the following folder in the tree structure:

   HKEY_CURRENT_USER\Software\KUKA Roboter GmbH\KUKA BOF\OCX Controls\UserMode
4. Select the menu sequence **Edit** > **New** > **DWORD value**.
5. Change the designation of the new value to “UserMode”.
6. Right-click on the value and select the option **Change**. The **Edit DWORD value** window opens.
7. Enter the desired identification number as a decimal value or hexadecimal value and press **OK**.
8. Close the **Registry editor** window.

When the system is next booted, the new default user group will automatically be activated.

6.23 Configuring Automatic External

Description

If robot processes are to be controlled centrally by a higher-level controller (e.g. a PLC), this is carried out using the Automatic External interface.

The higher-level controller transmits the signals for the robot processes (e.g. motion enable, fault acknowledgement, program start, etc.) to the robot controller via the Automatic External interface. The robot controller transmits information about operating states and fault states to the higher-level controller.

Overview

To enable use of the Automatic External interface, the following configurations must be carried out:

Step	Description
1	Configuration of the CELL.SRC program. (>>> 6.23.1 "Configuring CELL.SRC" Page 174)
2	Configuration of the inputs/outputs of the Automatic External interface. (>>> 6.23.2 "Configuring Automatic External inputs/outputs" Page 175)
3	Only if error numbers are to be transmitted to the higher-level controller: configuration of the P00.DAT file. (>>> 6.23.3 "Transmitting error numbers to the higher-level controller" Page 180)
4	Set the following variables to “1” in the file PROGRESS.INI in the INIT directory of the robot controller: ■ UPS_CONTROL_MONITORING ■ MFC_AA_AF_SIGNALS

6.23.1 Configuring CELL.SRC

Description

In Automatic External mode, programs are called using the program CELL.SRC.

Program

```
1  DEF  CELL ( )
     …
6    INIT
7    BASISTECH INI
8    CHECK HOME
9    PTP HOME  Vel= 100 % DEFAULT
10    AUTOEXT INI
11    LOOP
12      P00 (#EXT_PGNO,#PGNO_GET,DMY[],0 )
13      SWITCH  PGNO ; Select with Programnumber
14
15      CASE 1
16        P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
17        ;EXAMPLE1 ( ) ; Call User-Program
18
19      CASE 2
20        P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
21        ;EXAMPLE2 ( ) ; Call User-Program
22
23      CASE 3
24        P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
25        ;EXAMPLE3 ( ) ; Call User-Program
26
27      DEFAULT
28        P00 (#EXT_PGNO,#PGNO_FAULT,DMY[],0 )
29      ENDSWITCH
30    ENDLOOP
31  END
```

Line	Description
12	The robot controller calls the program number from the higher-level controller.
15	CASE branch for program number = 1
16	Receipt of program number 1 is communicated to the higher-level controller.
17	The user-defined program EXAMPLE1 is called.
27	DEFAULT = the program number is invalid.
28	Error treatment in the case of an invalid program number

Procedure

1. Open the program CELL.SRC in the Navigator. (This program is located in the folder "R1".)
2. In the section CASE 1, replace the name EXAMPLE1 with the name of the program that is to be called via program number 1. Delete the semicolon in front of the name.

```
  …
15      CASE 1
16        P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
17        MY_PROGRAM ( ) ; Call User-Program
…
```

3. For all further programs, proceed as described in step 2.

   If required, add additional CASE branches. To do so, select a CASE branch by means of SHIFT+CURSOR and copy and paste it using the **Edit** menu.
4. Press the **Close** softkey. Respond to the request for confirmation asking whether the changes should be saved with **Yes**.

6.23.2 Configuring Automatic External inputs/outputs

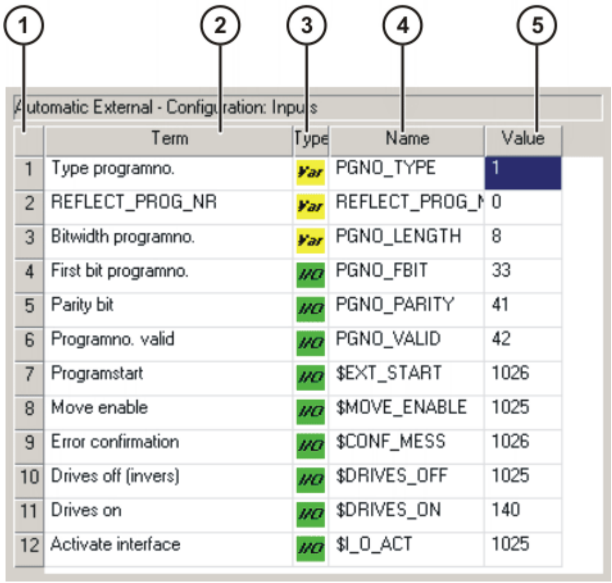
Procedure

1. Select the menu sequence **Configure** > **I/O** > **Automatic External**.
2. In the **Value** column, select the cell to be edited and press the **Value** softkey.
3. Enter the desired value and save it by pressing the **OK** softkey.
4. Exit the configuration by pressing the **Close** softkey.

Description of the inputs and outputs:

- (>>> 6.23.2.1 "Automatic External inputs" Page 176)
- (>>> 6.23.2.2 "Automatic External outputs" Page 178)

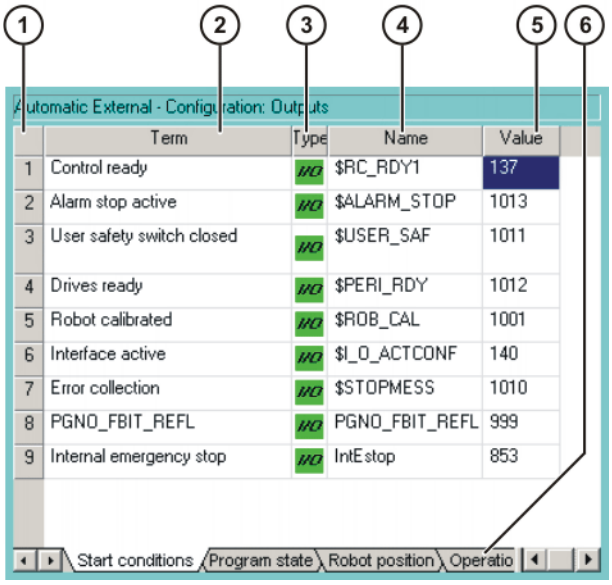
Description



Automatic External - Configuration: Inputs

	Term	Type	Name	Value
1	Type programno.	Var	PGNO_TYPE	1
2	REFLECT_PROG_NR	Var	REFLECT_PROG_N	0
3	Bitwidth programno.	Var	PGNO_LENGTH	8
4	First bit programno.	I/O	PGNO_FBIT	33
5	Parity bit	I/O	PGNO_PARITY	41
6	Programno. valid	I/O	PGNO_VALID	42
7	Programstart	I/O	$EXT_START	1026
8	Move enable	I/O	$MOVE_ENABLE	1025
9	Error confirmation	I/O	$CONF_MESS	1026
10	Drives off (invers)	I/O	$DRIVES_OFF	1025
11	Drives on	I/O	$DRIVES_ON	140
12	Activate interface	I/O	$I_O_ACT	1025

Fig. 6-30: Configuring Automatic External inputs



Automatic External - Configuration: Outputs

	Term	Type	Name	Value
1	Control ready	I/O	$RC_RDY1	137
2	Alarm stop active	I/O	$ALARM_STOP	1013
3	User safety switch closed	I/O	$USER_SAF	1011
4	Drives ready	I/O	$PERI_RDY	1012
5	Robot calibrated	I/O	$ROB_CAL	1001
6	Interface active	I/O	$I_O_ACTCONF	140
7	Error collection	I/O	$STOPMESS	1010
8	PGNO_FBIT_REFL	I/O	PGNO_FBIT_REFL	999
9	Internal emergency stop	I/O	IntEstop	853

Fig. 6-31: Configuring Automatic External outputs

Item	Description
1	Number
2	Long text name of the input/output

Item	Description
3	Type - **Green**: Input/output - **Yellow**: Variable or system variable ($...)
4	Name of the signal or variable
5	Input/output number or channel number
6	The outputs are thematically assigned to the following tabs: - Start conditions - Program state - Robot position - Operating mode

The following softkeys are available:

Softkey	Description
Display	Switches to the Automatic External display. (>>> 4.20.4 "Displaying inputs/outputs for Automatic External" Page 76)
Inputs/Outputs	Toggles between the windows for inputs and outputs.
Value	The selected value is made available for editing.
Tab -/Tab +	Toggles between the tabs. This softkey is only available for outputs.

6.23.2.1 Automatic External inputs

PGNO_TYPE

Type: variable

This variable defines the format in which the program number sent by the higher-level controller is read.

Value	Description	Example
1	Read as binary number. The program number is transmitted by the higher-level controller as a binary coded integer.	0 0 1 0 0 1 1 1 => PGNO = 39
2	Read as BCD value. The program number is transmitted by the higher-level controller as a binary coded decimal.	0 0 1 0 0 1 1 1 => PGNO = 27
3	Read as “1 of n”*. The program number is transmitted by the higher-level controller or the periphery as a "1 of n" coded value.	0 0 0 0 0 0 0 1 => PGNO = 1 0 0 0 0 1 0 0 0 => PGNO = 4

* When using this transmission format, the values of PGNO_REQ, PGNO_PARITY and PGNO_VALID are not evaluated and are thus of no significance.

REFLECT_PROG_NR

Type: variable

This variable defines whether the program number is to be mirrored to an output range. The output of the signal starts with the output defined using PGNO_FBIT_REFL. (>>> 6.23.2.2 "Automatic External outputs" Page 178)

Value	Description
0	Function deactivated
1	Function activated

PGNO_LENGTH

Type: variable

This variable determines the number of bits in the program number sent by the higher-level controller. Range of values: 1 … 16.

Example: PGNO_LENGTH = 4 => the external program number is 4 bits long.

If PGNO_TYPE has the value 2, only 4, 8, 12 and 16 are permissible values for the number of bits.

PGNO_FBIT

Input representing the first bit of the program number. Range of values: 1 … 4096.

Example: PGNO_FBIT = 5 => the external program number begins with the input $IN[5].

PGNO_PARITY

Input to which the parity bit is transferred from the higher-level controller.

Input	Function
Negative value	Odd parity
0	No evaluation
Positive value	Even parity

If PGNO_TYPE has the value 3, PGNO_PARITY is not evaluated.

PGNO_VALID

Input to which the command to read the program number is transferred from the higher-level controller.

Input	Function
Negative value	Number is transferred at the falling edge of the signal.
0	Number is transferred at the rising edge of the signal on the EXT_START line.
Positive value	Number is transferred at the rising edge of the signal.

If PGNO_TYPE has the value 3, PGNO_VALID is not evaluated.

$EXT_START

If the I/O interface is active, this input can be set to start or continue a program.

Only the rising edge of the signal is evaluated.

⚠ WARNING There is no BCO run in Automatic External mode. This means that the robot moves to the first programmed position after the start at the programmed (not reduced) velocity and does not stop there.

$MOVE_ENABLE

This input is used by the higher-level controller to check the robot drives.

Signal	Function
TRUE	Jogging and program execution are possible.
FALSE	All drives are stopped and all active commands inhibited.

If the drives have been switched off by the higher-level controller, the message “GENERAL MOTION ENABLE” is displayed. It is only possible to move the robot again once this message has been reset and another external start signal has been given.

During commissioning, the variable $MOVE_ENABLE is often configured with the value $IN[1025]. If a different input is not subsequently configured, no external start is possible.

$CHCK_MOVENA

Type: variable

If the variable $CHCK_MOVENA has the value FALSE, $MOVE_ENABLE can be bypassed. The value of the variable can only be changed in the file C:\KRC\ROBOTER\KRC\STEU\Mada\$OPTION.DAT.

Signal	Function
TRUE	MOVE_ENABLE monitoring is activated.
FALSE	MOVE_ENABLE monitoring is deactivated.

In order to be able to use MOVE_ENABLE monitoring, $MOVE_ENABLE must have been configured with the input $IN[1025]. Otherwise, $CHCK_MOVENA has no effect.

$CONF_MESS

Setting this input enables the higher-level controller to acknowledge error messages automatically as soon as the cause of the error has been eliminated.

Only the rising edge of the signal is evaluated.

$DRIVES_OFF

If there is a low-level pulse of at least 20 ms duration at this input, the higher-level controller switches off the robot drives.

$DRIVES_ON

If there is a high-level pulse of at least 20 ms duration at this input, the higher-level controller switches on the robot drives.

$I_O_ACT

If this input is TRUE, the Automatic External interface is active. Default setting: $IN[1025].

6.23.2.2 Automatic External outputs

$RC_RDY1

Ready for program start.

$ALARM_STOP

This output is reset in the following EMERGENCY STOP situations:

- The EMERGENCY STOP button on the KCP is pressed.
- External E-STOP

In the case of an EMERGENCY STOP, the nature of the EMERGENCY STOP can be recognized from the states of the outputs **$ALARM_STOP** and **Int. E-Stop**:

- Both outputs are FALSE: the EMERGENCY STOP was triggered on the KCP.
- **$ALARM_STOP** is FALSE, **Int. E-Stop** is TRUE: external EMERGENCY STOP.

$USER_SAF

This output is reset if the safety fence monitoring switch is opened (AUTO mode) or an enabling switch is released (T1 or T2 mode).

$PERI_RDY

By setting this output, the robot controller communicates to the higher-level controller the fact that the robot drives are switched on.

$ROB_CAL

The signal is FALSE as soon as a robot axis has been unmastered

$I_O_ACTCONF

This output is TRUE if Automatic External mode is selected and the input $I_O_ACT is TRUE.

$STOPMESS

This output is set by the robot controller in order to communicate to the higher-level controller any message occurring which requires the robot to be stopped. (Examples: EMERGENCY STOP, Driving condition or Operator safety)

PGNO_FBIT_REFL

Output representing the first bit of the program number. Precondition: The variable REFLECT_PROG_NR has the value 1. (>>> 6.23.2.1 "Automatic External inputs" Page 176)

The size of the output area depends on the number of bits defining the program number (PGNO_LENGTH).

If a program selected by the PLC is deselected by the user, the output area starting with PGNO_FBIT_REFL is set to FALSE. In this way, the PLC can prevent a program from being restarted manually.

PGNO_FBIT_REFL is also set to FALSE if the interpreter is situated in the CELL program.

Int. E-Stop

This output is set to FALSE if the EMERGENCY STOP button on the KCP is pressed.

> In the case of an EMERGENCY STOP, the nature of the EMERGENCY STOP can be recognized from the states of the outputs **$ALARM_STOP** and **Int. E-Stop**:
>
> - Both outputs are FALSE: the EMERGENCY STOP was triggered on the KCP.
> - **$ALARM_STOP** is FALSE, **Int. E-Stop** is TRUE: external EMERGENCY STOP.

$PRO_ACT

This output is always set if a process is active at robot level. The process is therefore active as long as a program or an interrupt is being processed. Program processing is set to the inactive state at the end of the program only after all pulse outputs and all triggers have been processed.

In the event of an error stop, a distinction must be made between the following possibilities:

- If interrupts have been activated but not processed at the time of the error stop, the process is regarded as inactive ($PRO_ACT=FALSE).
- If interrupts have been activated and processed at the time of the error stop, the process is regarded as active ($PRO_ACT=TRUE) until the interrupt program is completed or a STOP occurs in it ($PRO_ACT=FALSE).
- If interrupts have been activated and a STOP occurs in the program, the process is regarded as inactive ($PRO_ACT=FALSE). If, after this, an interrupt condition is met, the process is regarded as active ($PRO_ACT=TRUE) until the interrupt program is completed or a STOP occurs in it ($PRO_ACT=FALSE).

PGNO_REQ

A change of signal at this output requests the higher-level controller to send a program number.

If PGNO_TYPE has the value 3, PGNO_REQ is not evaluated.

APPL_RUN

By setting this output, the robot controller communicates to the higher-level controller the fact that a program is currently being executed.

$PRO_MOVE

Means that a synchronous axis is moving, including in jog mode. The signal is thus the inverse of $ROB_STOPPED.

$IN_HOME

This output communicates to the higher-level controller whether or not the robot is in its HOME position.

$ON_PATH

This output remains set as long as the robot stays on its programmed path. The output ON_PATH is set after the BCO run. This output remains set until the robot leaves the path, the program is reset or block selection is carried out. The ON_PATH signal has no tolerance window, however; as soon as the robot leaves the path the signal is reset.

$NEAR_POSRET

This signal allows the higher-level controller to determine whether or not the robot is situated within a sphere about the position saved in $POS_RET. The higher-level controller can use this information to decide whether or not the program may be restarted.

The user can define the radius of the sphere in the file $CUSTOM.DAT using the system variable $NEARPATHTOL.

$ROB_STOPPED

The signal is set when the robot is at a standstill. In the event of a WAIT statement, this output is set during the wait.

The signal is thus the inverse of $PRO_MOVE.

$T1, $T2, $AUT, $EXT

These outputs are set when the corresponding operating mode is selected.

6.23.3 Transmitting error numbers to the higher-level controller

Error numbers of the robot controller in the range 1 to 255 can be transmitted to the higher-level controller. To transmit the error numbers, the file P00.DAT, in the directory C:\KRC\ROBOTER\KRC\R1\TP, must be configured as follows:

```
DEFDAT  P00

BOOL PLC_ENABLE=TRUE ; Enable error-code transmission to plc
INT I
INT F_NO=1
INT MAXERR_C=1 ; maximum messages for $STOPMESS
INT MAXERR_A=1 ; maximum messages for APPLICATION
DECL STOPMESS MLD
SIGNAL ERR $OUT[25]  TO $OUT[32]
BOOL FOUND

STRUC PRESET INT OUT,CHAR PKG[3],INT ERR
DECL PRESET P[255]
…
P[1]={OUT 2,PKG[] "P00",ERR 10}
…
P[128]={OUT 128,PKG[] "CTL",ERR 1}
…
STRUC ERR_MESS CHAR P[3],INT E
DECL ERR_MESS ERR_FILE[64]
ERR_FILE[1]={P[] "XXX",E 0}
…
ERR_FILE[64]={P[] "XXX",E 0}
ENDDAT
```

Line	Description
3	PLC_ENABLE must be TRUE.
6	Enter the number of controller errors for the transmission of which parameters are to be defined.
7	Enter the number of application errors for the transmission of which parameters are to be defined.
9	Specify which robot controller outputs the higher-level controller should use to read the error numbers. There must be 8 outputs.
13	In the following section, enter the parameters of the errors. P[1] … P[127]: range for application errors P[128] … P[255]: range for controller errors
26	Example of parameters for application errors: ■ OUT 2 = error number 2 ■ PKG[] "P00" = technology package ■ ERR 10 = error number in the selected technology package
30	Example of parameters for controller errors: ■ OUT 128 = error number 128 ■ PKG[] "CTL" = technology package ■ ERR 1 = error number in the selected technology package
37 … 96	The last 64 errors that have occurred are stored in the ERR_FILE memory.

6.23.4 Signal diagrams

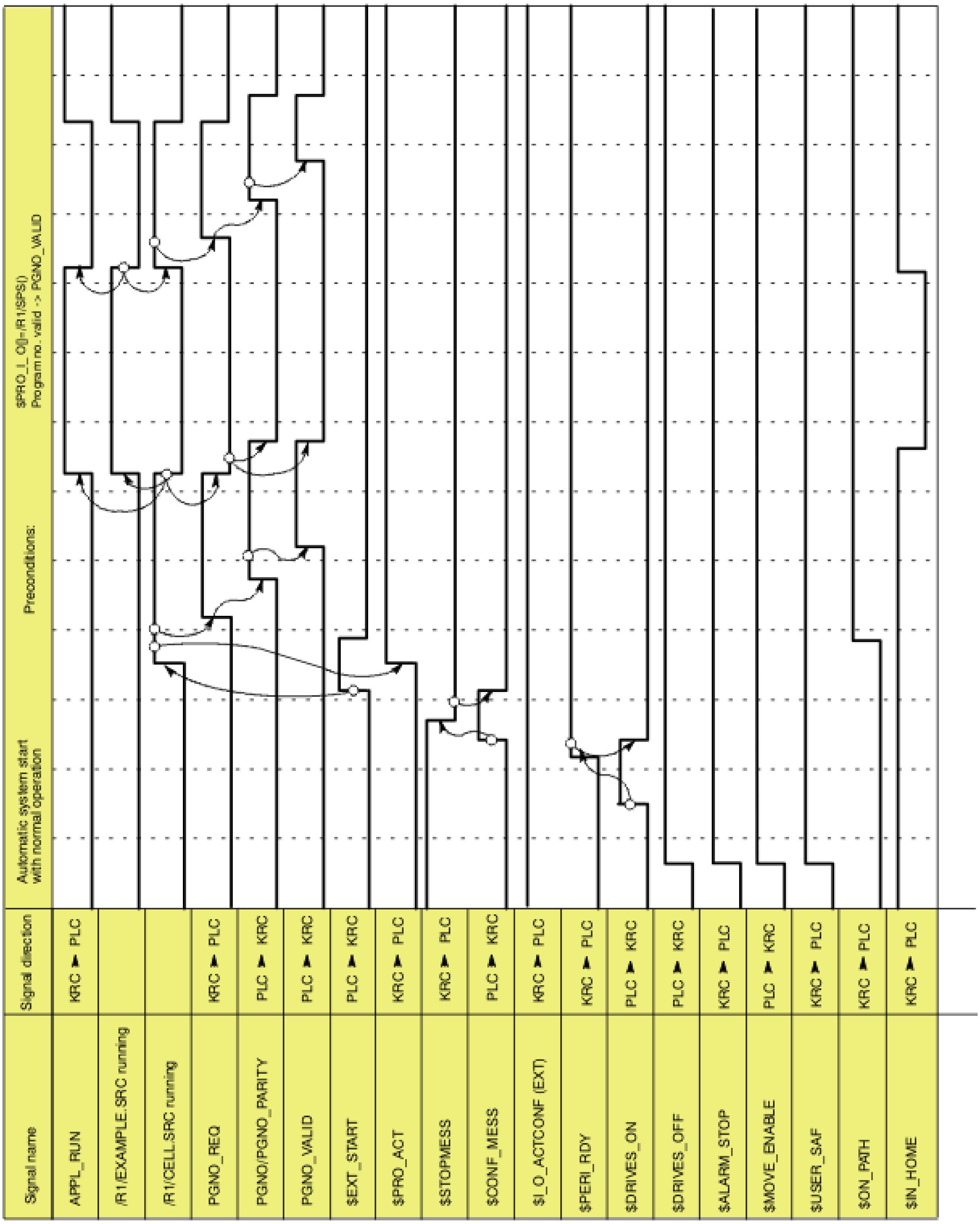
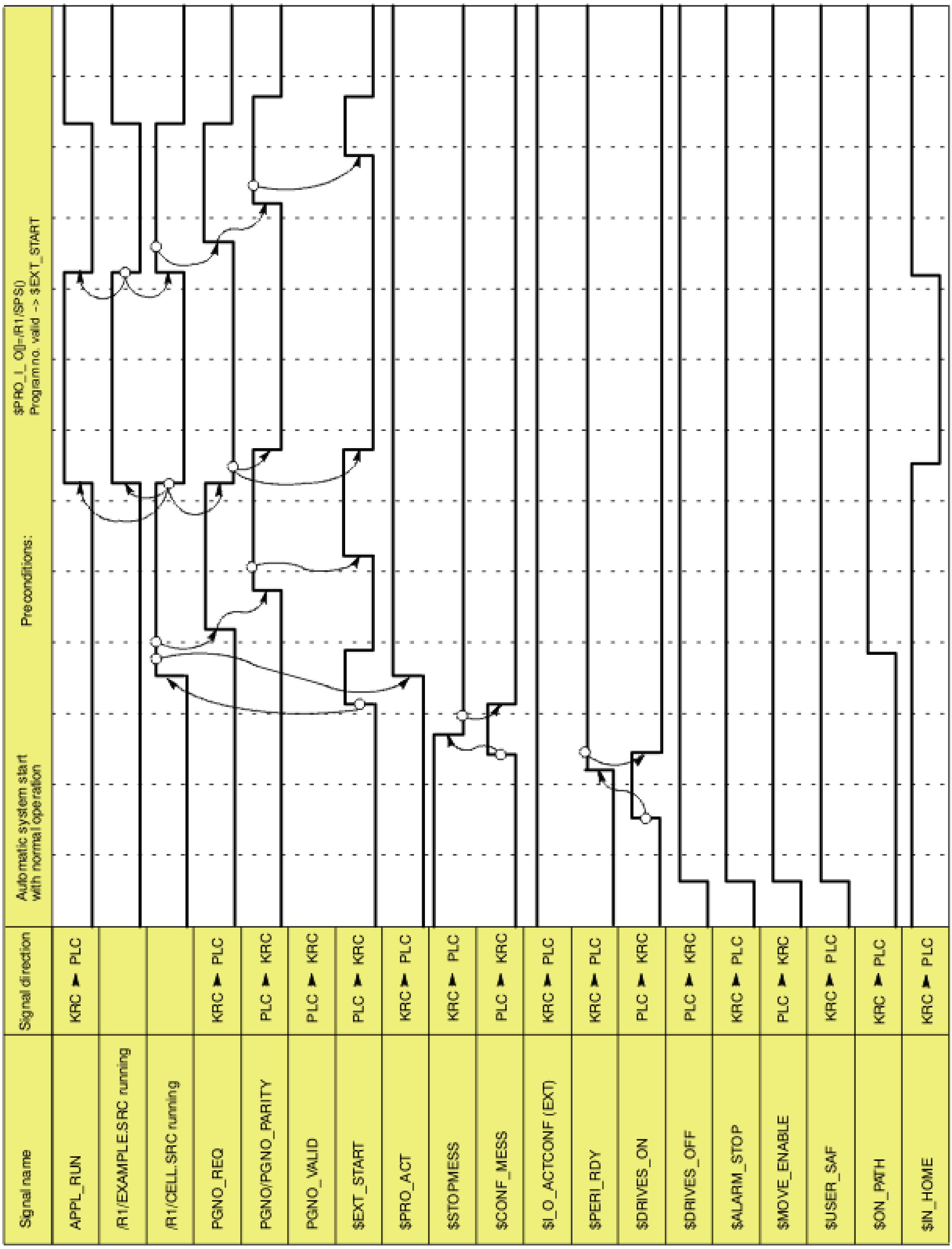
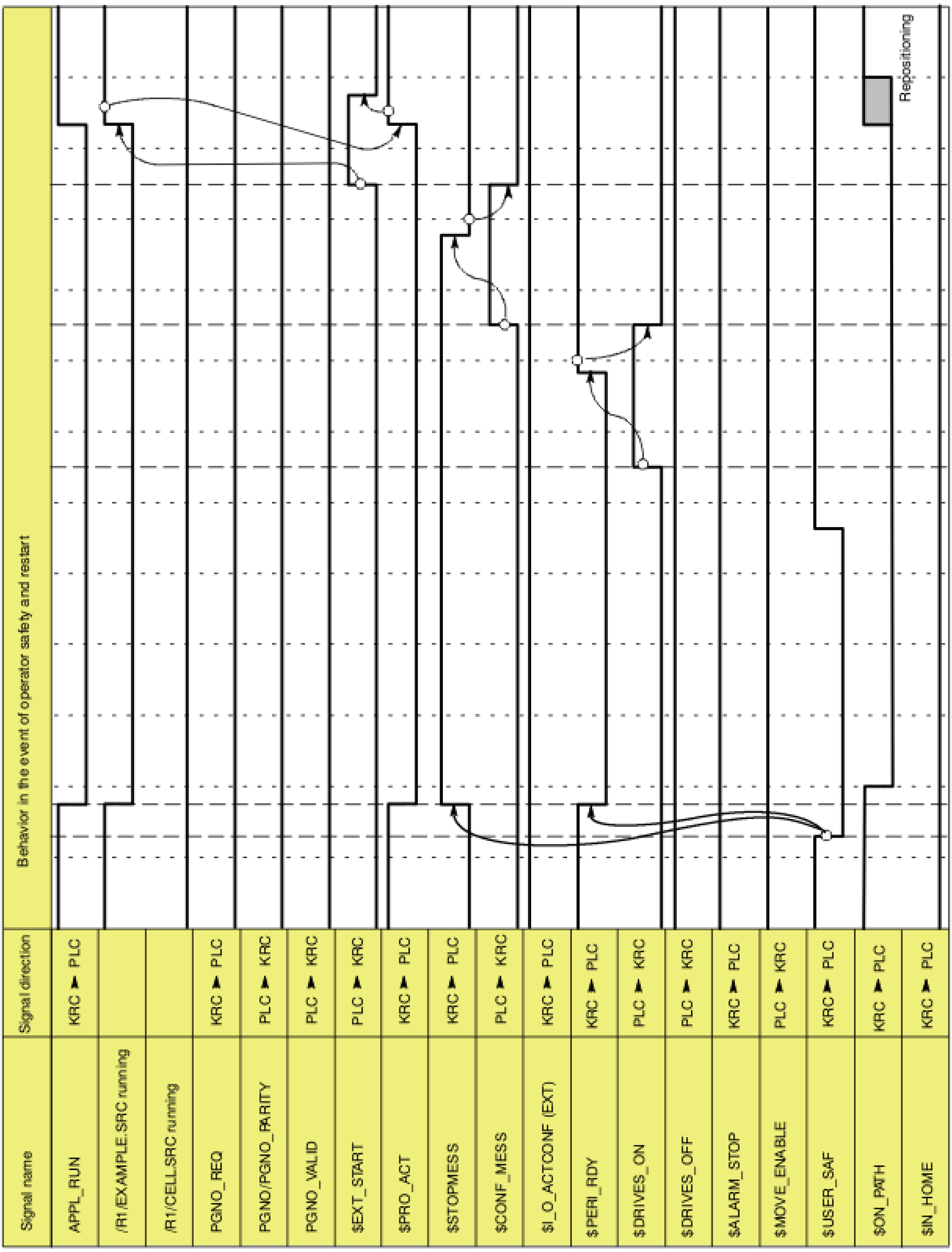


Fig. 6-32: Automatic system start and normal operation with program number acknowledgement by means of PGNO_VALID


Automatic system start with normal operation
Preconditions:
$PRO_I_O[]=/R1/SPS()
Program no. valid -> $EXT_START

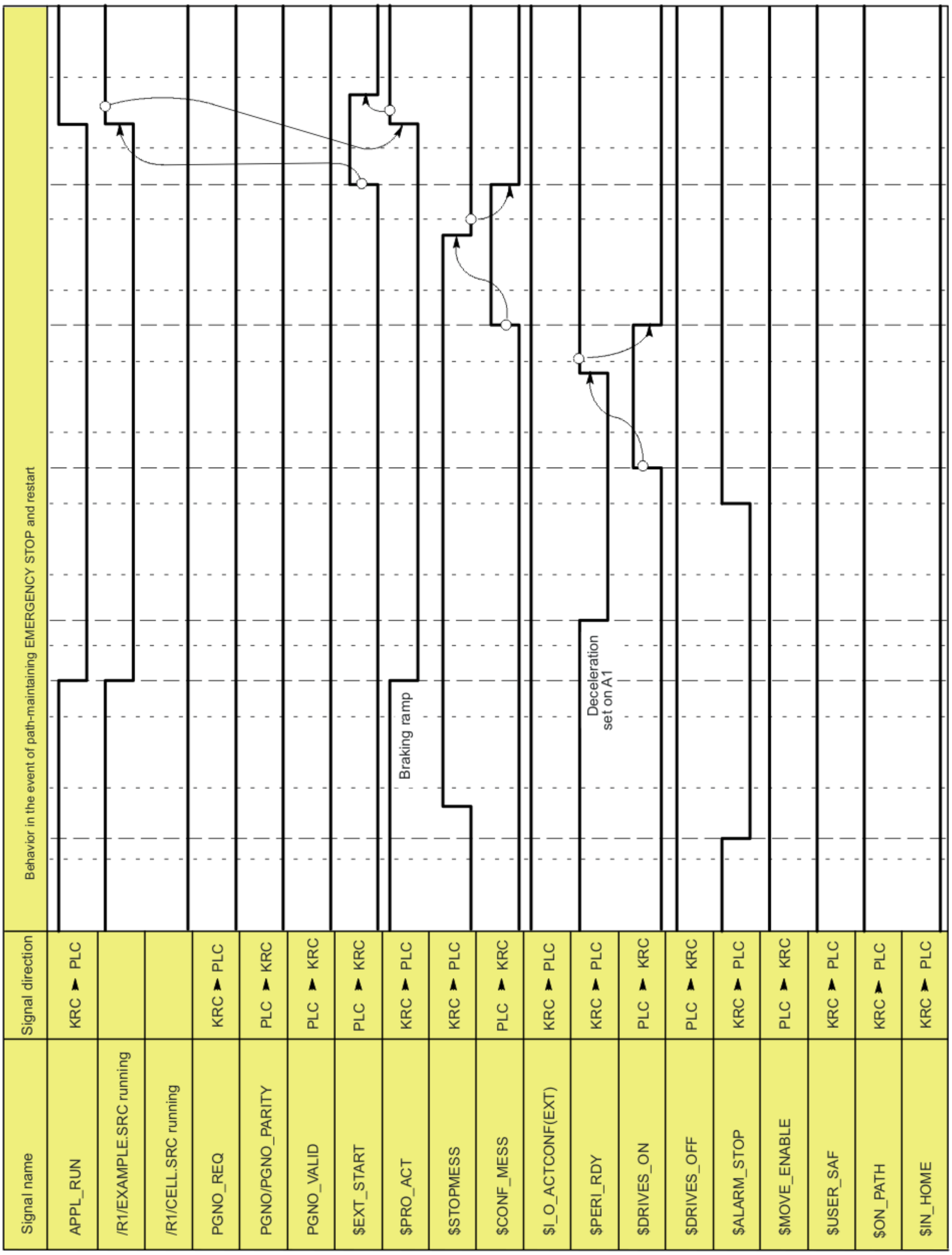
Signal name	Signal direction
APPL_RUN	KRC ▶ PLC
/R1/EXAMPLE.SRC running	
/R1/CELL.SRC running	
PGNO_REQ	KRC ▶ PLC
PGNO/PGNO_PARITY	PLC ▶ KRC
PGNO_VALID	PLC ▶ KRC
$EXT_START	PLC ▶ KRC
$PRO_ACT	KRC ▶ PLC
$STOPMESS	KRC ▶ PLC
$CONF_MESS	PLC ▶ KRC
$I_O_ACTCONF (EXT)	KRC ▶ PLC
$PERI_RDY	KRC ▶ PLC
$DRIVES_ON	PLC ▶ KRC
$DRIVES_OFF	PLC ▶ KRC
$ALARM_STOP	KRC ▶ PLC
$MOVE_ENABLE	PLC ▶ KRC
$USER_SAF	KRC ▶ PLC
$ON_PATH	KRC ▶ PLC
$IN_HOME	KRC ▶ PLC

Fig. 6-33: Automatic system start and normal operation with program number acknowledgement by means of $EXT_START



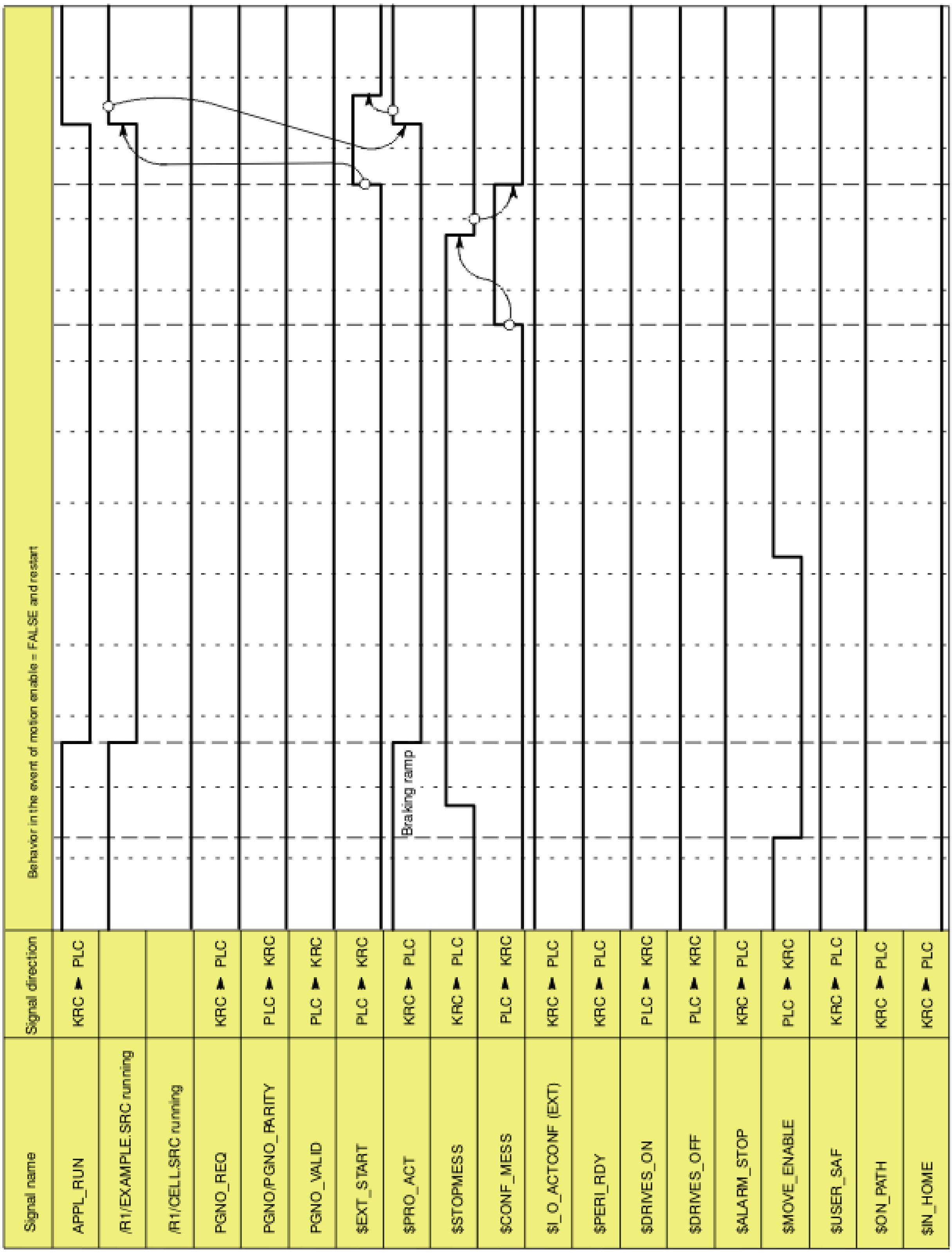
Signal name	Signal direction
APPL_RUN	KRC ➤ PLC
/R1/EXAMPLE.SRC running	
/R1/CELL.SRC running	
PGNO_REQ	KRC ➤ PLC
PGNO/PGNO_PARITY	PLC ➤ KRC
PGNO_VALID	PLC ➤ KRC
$EXT_START	PLC ➤ KRC
$PRO_ACT	KRC ➤ PLC
$STOPMESS	KRC ➤ PLC
$CONF_MESS	PLC ➤ KRC
$I_O_ACTCONF (EXT)	KRC ➤ PLC
$PERI_RDY	KRC ➤ PLC
$DRIVES_ON	PLC ➤ KRC
$DRIVES_OFF	PLC ➤ KRC
$ALARM_STOP	KRC ➤ PLC
$MOVE_ENABLE	PLC ➤ KRC
$USER_SAF	KRC ➤ PLC
$ON_PATH	KRC ➤ PLC
$IN_HOME	KRC ➤ PLC

Fig. 6-34: Restart after dynamic braking (operator safety and restart)


Behavior in the event of path-maintaining EMERGENCY STOP and restart
Braking ramp
Deceleration set on A1

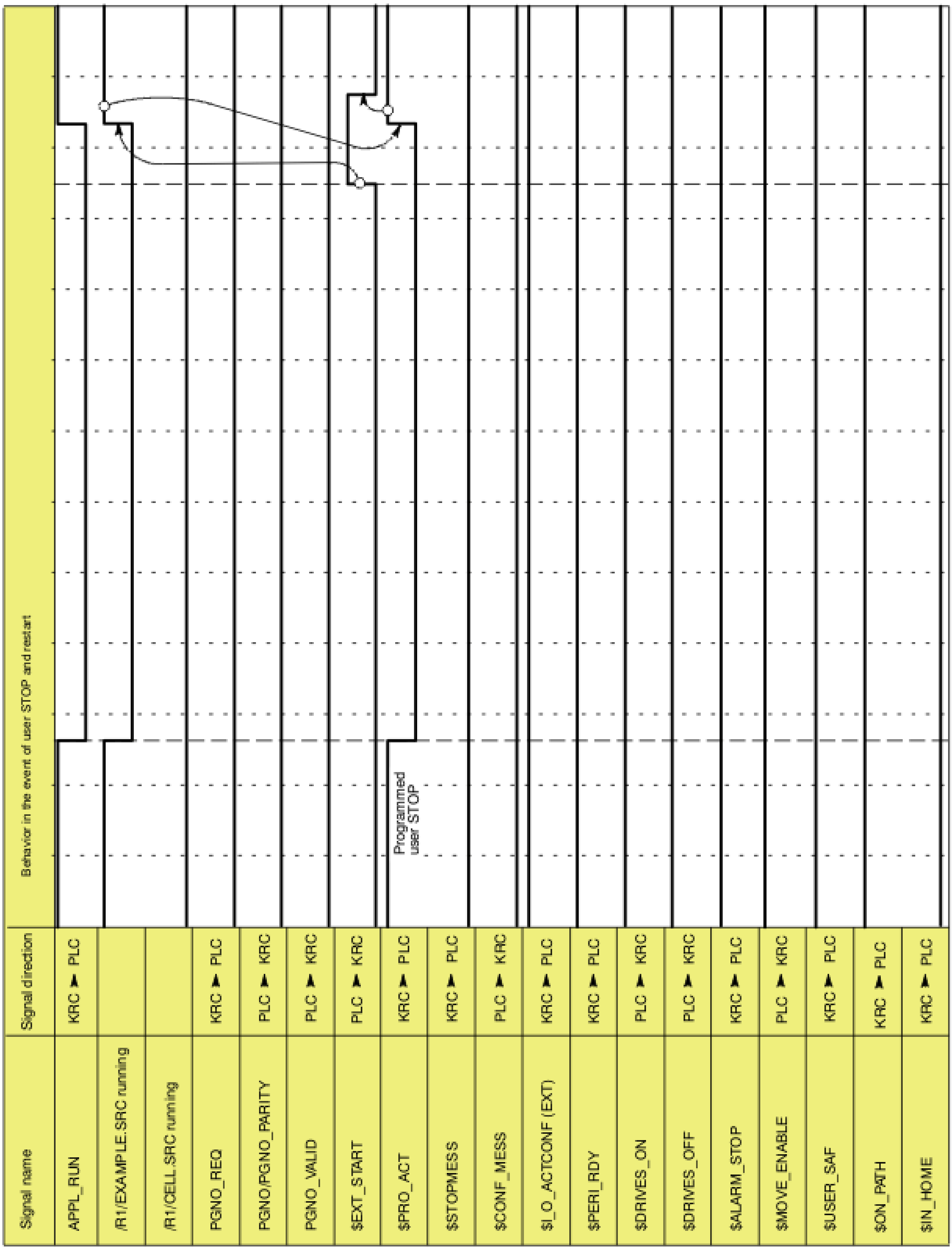
Signal name	Signal direction
APPL_RUN	KRC ➤ PLC
/R1/EXAMPLE.SRC running	
/R1/CELL.SRC running	
PGNO_REQ	KRC ➤ PLC
PGNO/PGNO_PARITY	PLC ➤ KRC
PGNO_VALID	PLC ➤ KRC
$EXT_START	PLC ➤ KRC
$PRO_ACT	KRC ➤ PLC
$STOPMESS	KRC ➤ PLC
$CONF_MESS	PLC ➤ KRC
$I_O_ACTCONF(EXT)	KRC ➤ PLC
$PERI_RDY	KRC ➤ PLC
$DRIVES_ON	PLC ➤ KRC
$DRIVES_OFF	PLC ➤ KRC
$ALARM_STOP	KRC ➤ PLC
$MOVE_ENABLE	PLC ➤ KRC
$USER_SAF	KRC ➤ PLC
$ON_PATH	KRC ➤ PLC
$IN_HOME	KRC ➤ PLC

Fig. 6-35: Restart after path-maintaining EMERGENCY STOP



Signal name	Signal direction
APPL_RUN	KRC ➤ PLC
/R1/EXAMPLE.SRC running	
/R1/CELL.SRC running	
PGNO_REQ	KRC ➤ PLC
PGNO/PGNO_PARITY	PLC ➤ KRC
PGNO_VALID	PLC ➤ KRC
$EXT_START	PLC ➤ KRC
$PRO_ACT	KRC ➤ PLC
$STOPMESS	KRC ➤ PLC
$CONF_MESS	PLC ➤ KRC
$I_O_ACTCONF (EXT)	KRC ➤ PLC
$PERI_RDY	KRC ➤ PLC
$DRIVES_ON	PLC ➤ KRC
$DRIVES_OFF	PLC ➤ KRC
$ALARM_STOP	KRC ➤ PLC
$MOVE_ENABLE	PLC ➤ KRC
$USER_SAF	KRC ➤ PLC
$ON_PATH	KRC ➤ PLC
$IN_HOME	KRC ➤ PLC

Fig. 6-36: Restart after motion enable



Signal name	Signal direction	Behavior in the event of user STOP and restart
APPL_RUN	KRC ► PLC	
/R1/EXAMPLE.SRC running		
/R1/CELL.SRC running		
PGNO_REQ	KRC ► PLC	
PGNO/PGNO_PARITY	PLC ► KRC	
PGNO_VALID	PLC ► KRC	
$EXT_START	PLC ► KRC	
$PRO_ACT	KRC ► PLC	
$STOPMESS	KRC ► PLC	
$CONF_MESS	PLC ► KRC	
$I_O_ACTCONF (EXT)	KRC ► PLC	
$PERI_RDY	KRC ► PLC	
$DRIVES_ON	PLC ► KRC	
$DRIVES_OFF	PLC ► KRC	
$ALARM_STOP	KRC ► PLC	
$MOVE_ENABLE	PLC ► KRC	
$USER_SAF	KRC ► PLC	
$ON_PATH	KRC ► PLC	
$IN_HOME	KRC ► PLC	

Fig. 6-37: Restart after user STOP

6.24 Event planner

This function can be used for time-related or action-related control of the data comparison between the kernel system and the hard drive. During data comparison, the kernel system data are written to the hard drive.

6.24.1 Configuring a data comparison

Procedure

1. Select the menu sequence **Configure** > **Tools** > **Event planner**.
2. Open the tree structure in the left-hand part of the window.
3. Select the desired action:
   - **T1 and T2 Consistency**: Compare data in operating modes T1 and T2 at regular intervals.
   - **AUT and EXT Consistency**: Compare data in operating modes Automatic and Automatic External at regular intervals.
   - **Logic Consistency**: Compare data after a change of operating mode or after online optimizing.

     Online optimizing is the modification of the parameters of a program during operation.
4. Make the desired settings in the right-hand part of the window.

   (>>> 6.24.2 "Configuring T1 and T2 Consistency, AUT and EXT Consistency" Page 188)

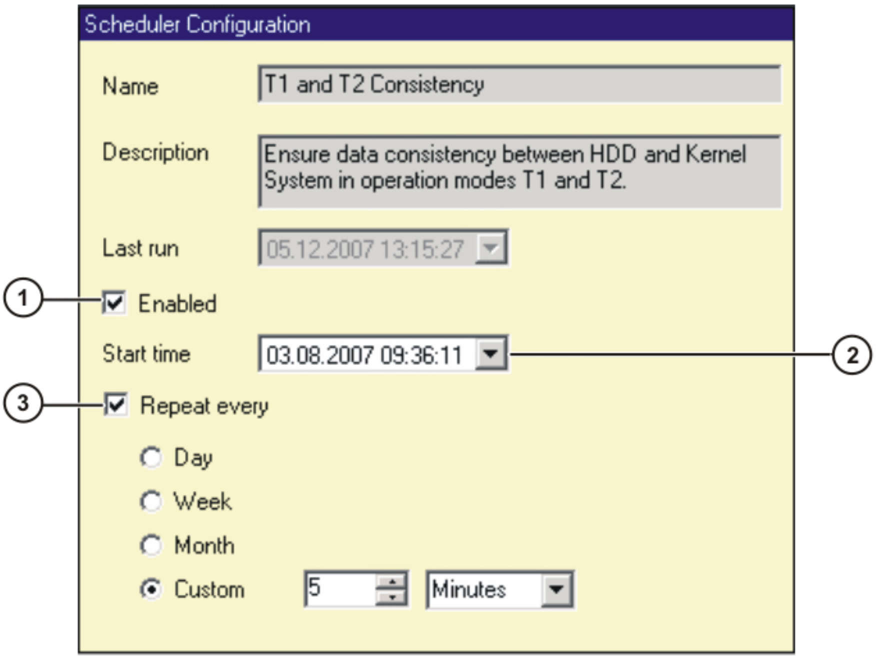
   (>>> 6.24.3 "Configuring Logic Consistency" Page 189)
5. Save the configuration with the **Apply** softkey.

6.24.2 Configuring T1 and T2 Consistency, AUT and EXT Consistency

The following settings are possible here:

- Definition of the date and time that a comparison will first be made between the data in the kernel system and those on the hard drive.
- Definition of the interval at which this operation is to be repeated.

Description



Scheduler Configuration

Name: T1 and T2 Consistency

Description: Ensure data consistency between HDD and Kernel System in operation modes T1 and T2.

Last run: 05.12.2007 13:15:27

☑ Enabled

Start time: 03.08.2007 09:36:11

☑ Repeat every

○ Day

○ Week

○ Month

● Custom 5 Minutes

Fig. 6-38: Configuring T1 and T2 Consistency

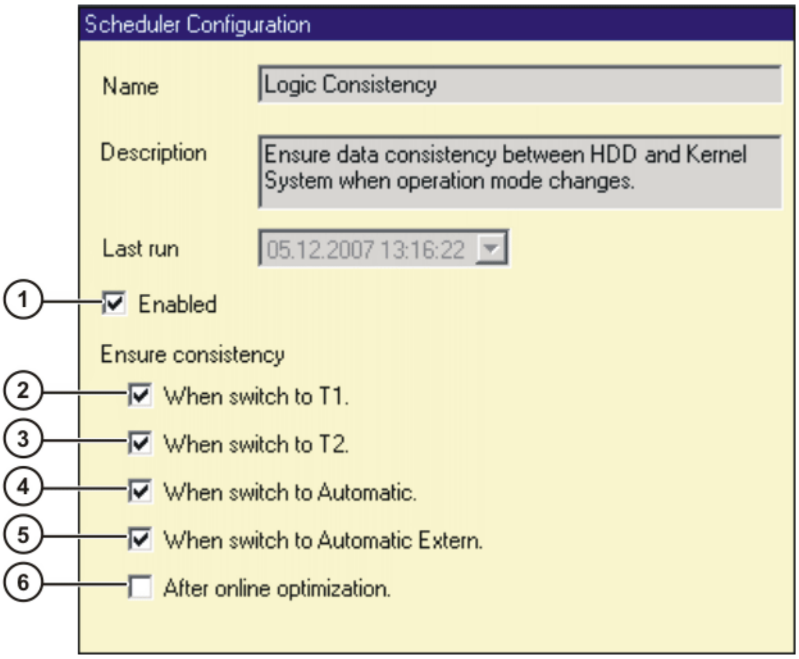
Item	Description
1	■ **Check box active:** data comparison is activated. ■ **Check box not active:** data comparison is deactivated.
2	Enter date and time in the format indicated.
3	■ **Check box active:** interval is activated. ■ **Check box not active:** interval is deactivated.

NOTICE If the intervals selected are too small, this can result in damage to the hard drive. An interval of several minutes is recommended.

6.24.3 Configuring Logic Consistency

Description

The following settings are possible here:



Scheduler Configuration

Name: Logic Consistency

Description: Ensure data consistency between HDD and Kernel System when operation mode changes.

Last run: 05.12.2007 13:16:22

☑ Enabled

Ensure consistency

☑ When switch to T1.

☑ When switch to T2.

☑ When switch to Automatic.

☑ When switch to Automatic Extern.

□ After online optimization.

Fig. 6-39: Configuring Logic Consistency

Item	Description
1	■ Check box activated: data comparison is activated. ■ Check box not active: data comparison is deactivated.
2	■ Check box activated: data are compared when operating mode is switched to T1. ■ Check box not active: data comparison is deactivated.
3	■ Check box activated: data are compared when operating mode is switched to T2. ■ Check box not active: data comparison is deactivated.
4	■ Check box activated: data are compared when operating mode is switched to Automatic. ■ Check box not active: data comparison is deactivated.
5	■ Check box activated: data are compared when operating mode is switched to Automatic External. ■ Check box not active: data comparison is deactivated.
6	■ Check box activated: data are compared after online optimizing. ■ Check box not active: data comparison is deactivated.

6.25 KRC Configurator

The KRC Configurator can be used to configure the Navigator and various other functions. Most functions can be specially configured for each user group.

> The settings in the KRC Configurator are not checked for validity or plausibility.

Precondition

- Windows interface (CTRL+ESC)

Procedure

1. Double-click on the file KrcConfigurator.exe in the directory C:\KRC\UTIL\KRCCONFIGURATOR. The KRC Configurator opens.
2. Edit the desired tab and save it by pressing **Apply**.

3. If necessary, repeat step 2 for other tabs.
4. Exit the KRC Configurator by pressing **Exit**.

6.25.1 Operating the KRC Configurator

The following functions are available in each of the tabs:

Select tab

1. Activate the menu bar by pressing the ALT key.
2. Using the arrow keys, select the **Tabs** menu and press the Enter key.
   The menu is opened.
3. Using the arrow keys, select the desired entry and press the Enter key.
   The desired tab is displayed.
4. Deactivate the menu bar by pressing the ALT key.

Edit tab

1. Press the TAB key to jump to the desired element on the user interface.
   Precondition: The NUM function must be deactivated.
2. List boxes: Select the desired entry by means of the arrow keys.
   Check boxes: Activate or deactivate by pressing the space bar.
   Buttons: Press the Enter key.

The following buttons are available in each of the tabs:

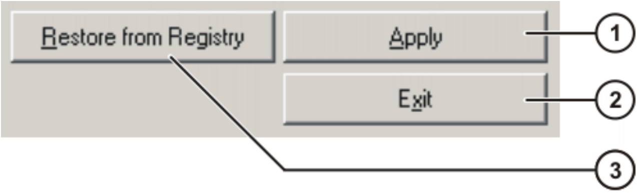


Fig. 6-40

Item	Description
1	Saves the changes in the current tab.
2	Closes the KRC Configurator. Changes that have not been applied with **Apply** are not saved.
3	Restores the KRC Configurator to the state it had when the program was started or the last time changes were saved with **Apply**.

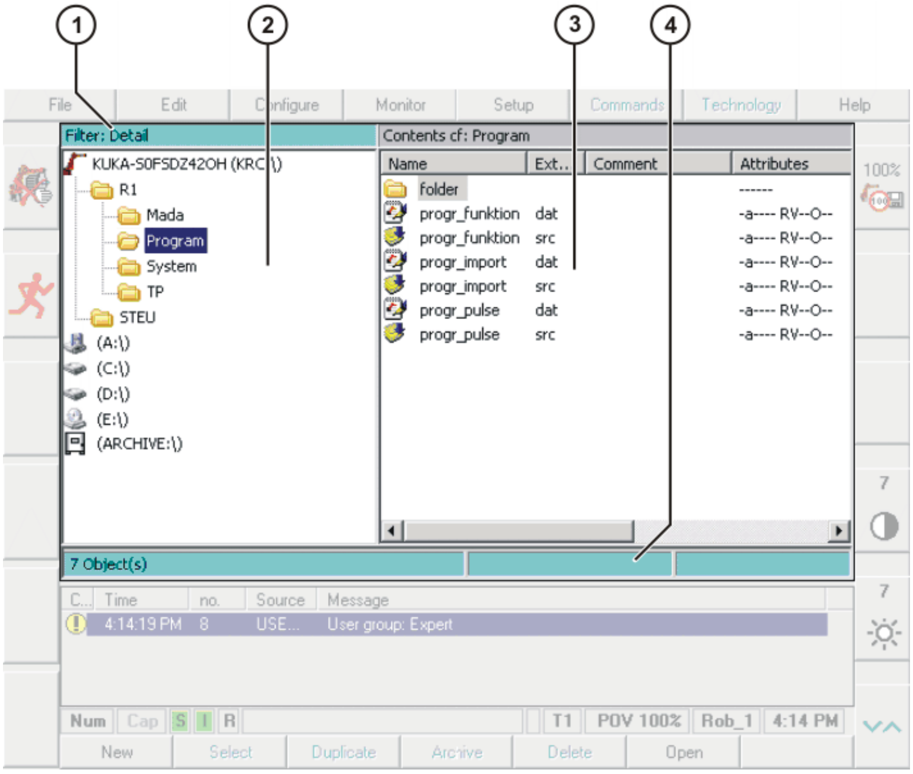
6.25.2 Display tab

Overview

The following Navigator properties can be configured here:

- Appearance of the directory structure
- Number of columns in the file list
- Number of columns in the file list

The Navigator can be specially configured for each user group.



Name	Ext..	Comment	Attributes
folder			------
progr_funktion	dat		-a---- RV--O--
progr_funktion	src		-a---- RV--O--
progr_import	dat		-a---- RV--O--
progr_import	src		-a---- RV--O--
progr_pulse	dat		-a---- RV--O--
progr_pulse	src		-a---- RV--O--

C...	Time	no.	Source	Message
!	4:14:19 PM	8	USE...	User group: Expert

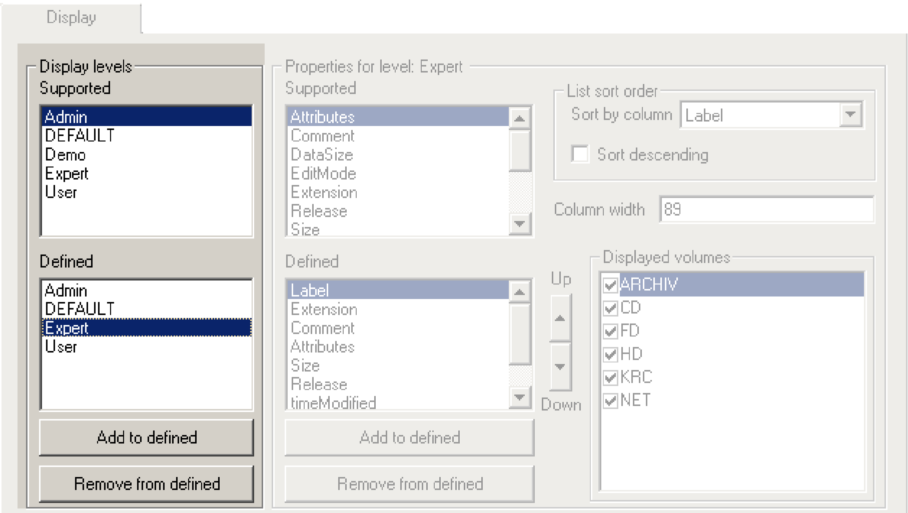
Fig. 6-41: Navigator

1	Header	3	File list
2	Directory structure	4	Status bar

Description:

The user group for which the Navigator is to be configured is selected here.

Display levels



Sort by column: Label

□ Sort descending

Column width: 89

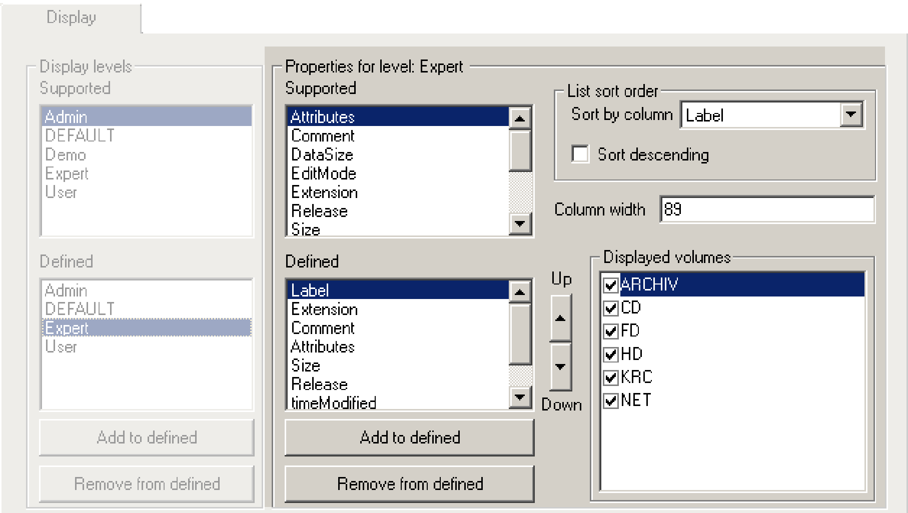
☑ ARCHIV
☑ CD
☑ FD
☑ HD
☑ KRC
☑ NET

Fig. 6-42: Display tab, Display levels

Element	Description
Supported	Available user groups (cannot be changed)
Defined	User groups set up in the system. The user group for which the Navigator is to be configured must be selected.
Add to defined	Adds the user group selected in **Supported** to **Defined**.
Remove from defined	Removes the user group selected in **Defined**.

Description: Properties for level

The Navigator is configured for a user group here.



Display levels, Supported: Admin, DEFAULT, Demo, Expert, User

Display levels, Defined: Admin, DEFAULT, Expert, User

Properties for level: Expert

Supported: Attributes, Comment, DataSize, EditMode, Extension, Release, Size

Defined: Label, Extension, Comment, Attributes, Size, Release, timeModified

List sort order

Sort by column: Label

☐ Sort descending

Column width: 89

Displayed volumes: ☑ ARCHIV, ☑ CD, ☑ FD, ☑ HD, ☑ KRC, ☑ NET

Fig. 6-43: Display tab, Properties for level

Element	Description
Properties for level:	The user group selected in **Display levels**, **Defined**, is displayed.
Supported	Columns that can be displayed in the file list (cannot be changed).
Defined	Columns that are displayed in the file list.
Add to defined	Adds the column selected in **Supported** to the file list.
Remove from defined	Removes the column selected in **Defined**.
Up, Down	Changes the order of the columns displayed in the file list. (Exception: **Label** cannot be moved.)
Sort by column	Defines the column to be used for sorting the file list.
Sort descending	Check box active: inverted sorting order
Column width	Column width of the column selected in **Defined**.
Displayed volumes	Drives that are displayed in the directory structure.

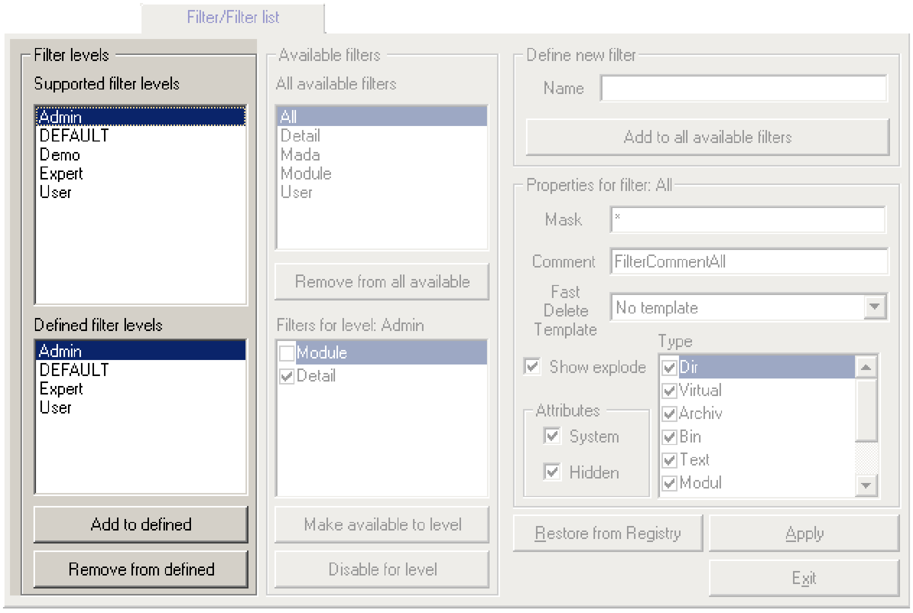
6.25.3 Filter tab

Overview

The filters available in the Navigator are configured here. The filters can be specially configured for each user group.

Description: Filter levels

The user group for which the filters are to be configured is selected here.



Mask: *

Comment: FilterCommentAll

Fast Delete Template: No template

☐ Module
☑ Detail

☑ Show explode

☑ System
☑ Hidden

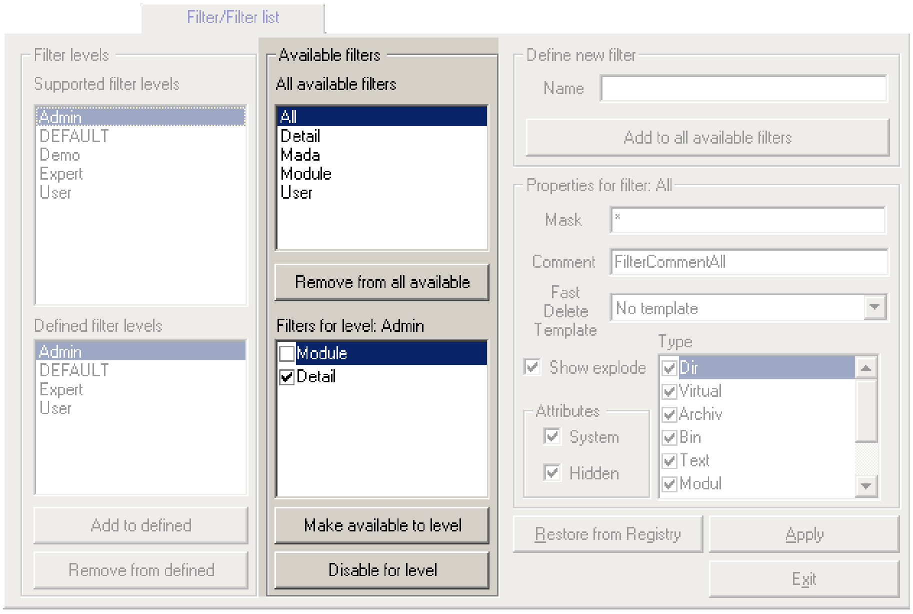
☑ Dir
☑ Virtual
☑ Archiv
☑ Bin
☑ Text
☑ Modul

Fig. 6-44: Filter tab, Filter levels

Element	Description
Supported filter levels	Available user groups (cannot be changed)
Defined filter levels	User groups set up in the system. The user group for which the filters are to be configured must be selected.
Add to defined	Adds the user group selected in **Supported filter levels** to **Defined filter levels**.
Remove from defined	Removes the user group selected in **Defined filter levels**.

Description: Available filters

Filters are assigned here to a user group.



Mask: *

Comment: FilterCommentAll

Fast Delete Template: No template

☐ Module
☑ Detail

☑ Show explode

☑ System
☑ Hidden

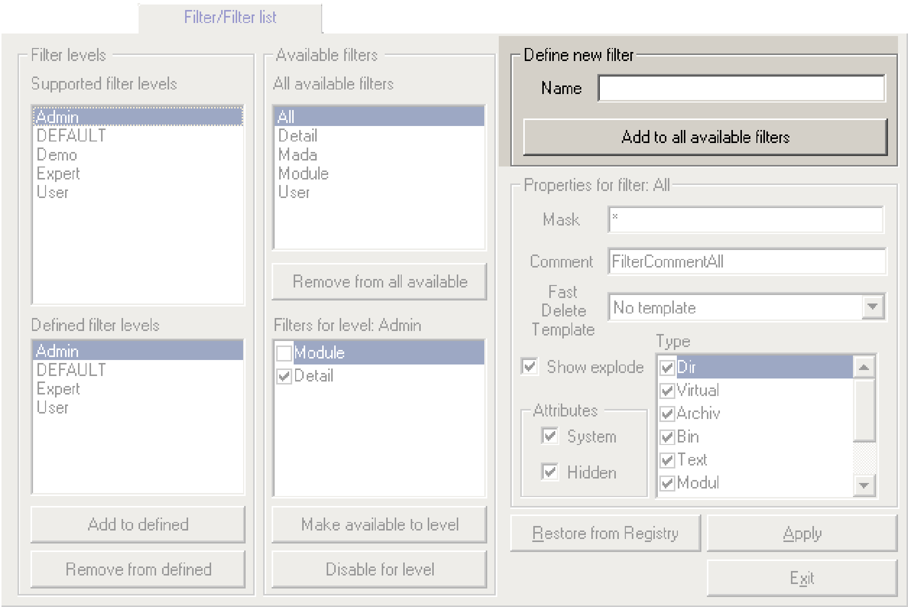
☑ Dir
☑ Virtual
☑ Archiv
☑ Bin
☑ Text
☑ Modul

Fig. 6-45: Filter tab, Available filters

Element	Description
All available filters	Available filters
Remove from all available	Removes a filter from the list of available filters.
Filters for level:	Filters assigned to the user group selected in **Defined filter levels**. Check box active: This filter is used by default with this user group.
Make available to level	Adds the filter selected in **All available Filters** to **Filters for level**.
Disable for level	Removes the filter selected in **Filters for level**.

Description: Define new filter

New filters can be defined here.



Supported filter levels: Admin, DEFAULT, Demo, Expert, User

Defined filter levels: Admin, DEFAULT, Expert, User

All available filters: All, Detail, Mada, Module, User

Filters for level: Admin: □ Module, ☑ Detail

Define new filter — Name:

Properties for filter: All

Mask: *

Comment: FilterCommentAll

Fast Delete Template: No template

☑ Show explode

Attributes: ☑ System, ☑ Hidden

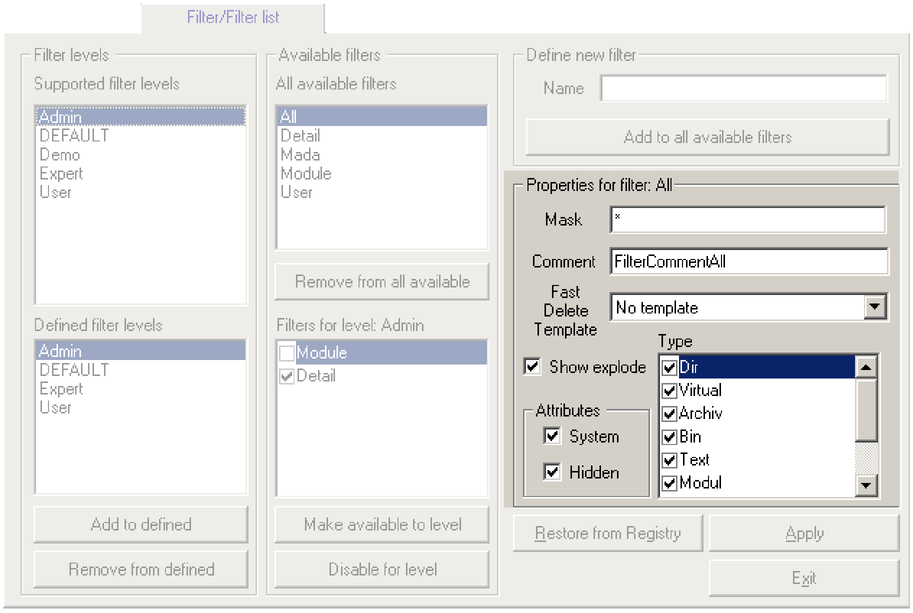
Type: ☑ Dir, ☑ Virtual, ☑ Archiv, ☑ Bin, ☑ Text, ☑ Modul

Fig. 6-46: Filter tab, Define new filter

Element	Description
Name	Name of the new filter
Add to all available filters	Adds the filter to **All available filters**.

Description: Properties for filter

The properties for a filter are defined here.



Supported filter levels: Admin, DEFAULT, Demo, Expert, User

Defined filter levels: Admin, DEFAULT, Expert, User

All available filters: All, Detail, Mada, Module, User

Filters for level: Admin: □ Module, ☑ Detail

Name:

Properties for filter: All

Mask: *

Comment: FilterCommentAll

Fast Delete Template: No template

☑ Show explode

Attributes: ☑ System, ☑ Hidden

Type: ☑ Dir, ☑ Virtual, ☑ Archiv, ☑ Bin, ☑ Text, ☑ Modul

Fig. 6-47: Filter tab, Properties for filter

Element	Description
Properties for filter:	The filter selected in **All available filters** is displayed.
Mask	Defines which files are displayed in the Navigator. Example: *.src = all SRC files are displayed.
Comment	Comment relating to the filter. When a filter is selected, the comment is displayed in the Navigator alongside the name of the filter. If the filter that is defined by default in the KSS is selected, this character string is not displayed; instead, the translation contained in the KUKA language database is displayed.
Fast Delete Template	This function is not currently supported.
Show explode	Check box active: SRC and DAT files are displayed separately in the Navigator.
Attributes	Check box active: Files with the attribute **System** or **Hidden** are displayed in the Navigator.
Type	Objects that are displayed in the Navigator. The following objects are available: ■ **Dir**: directories ■ **Virtual**: virtual directories (if available) ■ **Archiv**: archive files ■ **Bin**: binary files ■ **Text**: text files ■ **Modul**: modules ■ **Raw**: all other file types ■ **Ibgn file**: IBGN files ■ **Protected files**: encrypted and/or signed files

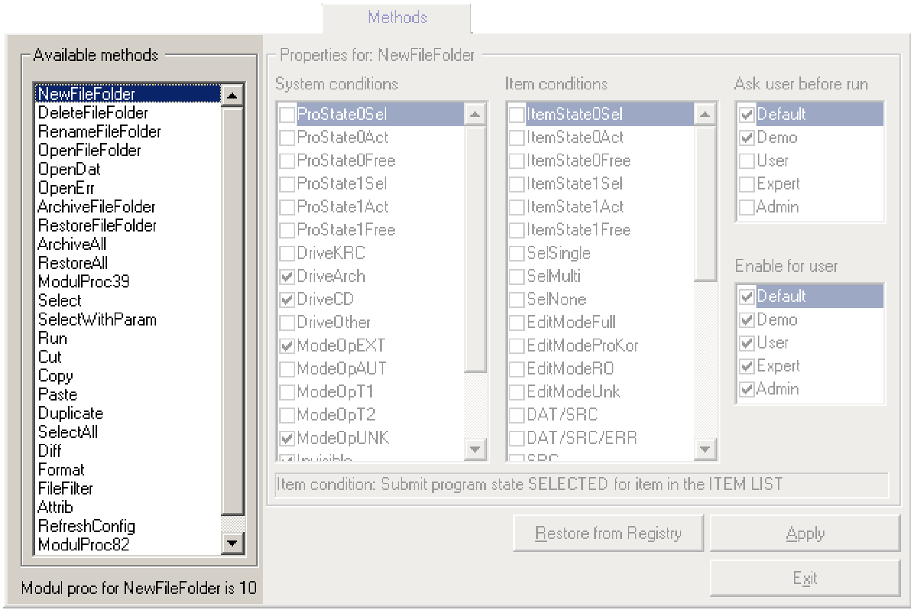
6.25.4 Methods tab

Overview

Here the user defines which commands are allowed in the Navigator, subject to certain system states and object states. Furthermore, the user also defines which commands are available for which user group.

Description: Available methods

The Navigator command whose properties are to be defined is selected here.



Available methods: NewFileFolder, DeleteFileFolder, RenameFileFolder, OpenFileFolder, OpenDat, OpenErr, ArchiveFileFolder, RestoreFileFolder, ArchiveAll, RestoreAll, ModulProc39, Select, SelectWithParam, Run, Cut, Copy, Paste, Duplicate, SelectAll, Diff, Format, FileFilter, Attrib, RefreshConfig, ModulProc82

Modul proc for NewFileFolder is 10

Properties for: NewFileFolder

System conditions: □ ProState0Sel, □ ProState0Act, □ ProState0Free, □ ProState1Sel, □ ProState1Act, □ ProState1Free, □ DriveKRC, ☑ DriveArch, ☑ DriveCD, □ DriveOther, ☑ ModeOpEXT, □ ModeOpAUT, □ ModeOpT1, □ ModeOpT2, ☑ ModeOpUNK, ☑ Invisible

Item conditions: □ ItemState0Sel, □ ItemState0Act, □ ItemState0Free, □ ItemState1Sel, □ ItemState1Act, □ ItemState1Free, □ SelSingle, □ SelMulti, □ SelNone, □ EditModeFull, □ EditModeProKor, □ EditModeRO, □ EditModeUnk, □ DAT/SRC, □ DAT/SRC/ERR, □ SRC

Ask user before run: ☑ Default, ☑ Demo, □ User, □ Expert, □ Admin

Enable for user: ☑ Default, ☑ Demo, ☑ User, ☑ Expert, ☑ Admin

Item condition: Submit program state SELECTED for item in the ITEM LIST

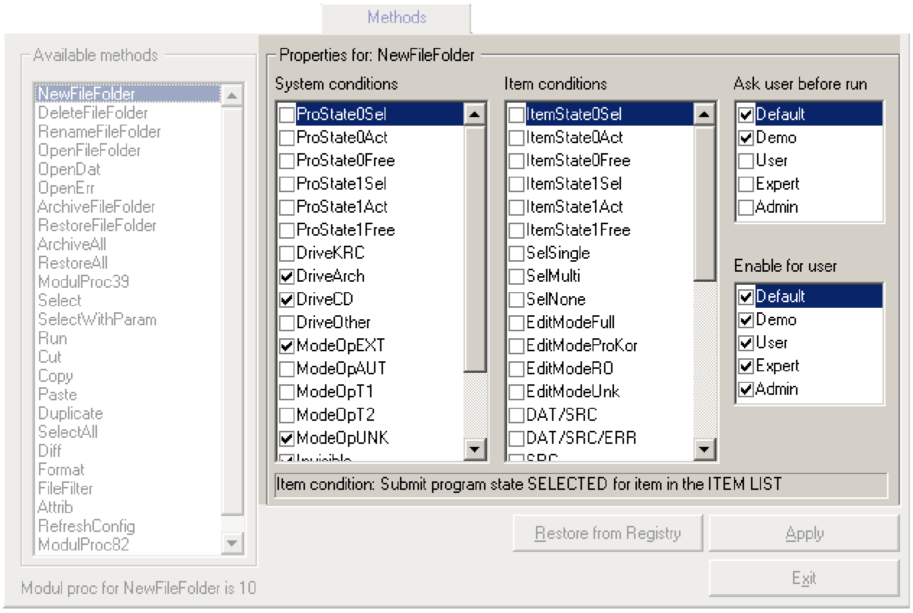
Fig. 6-48: Methods tab, Available methods

Available methods	Navigator command
NewFileFolder	Create directory or file
DeleteFileFolder	Delete directory or file
RenameFileFolder	Rename directory or file
OpenFileFolder	Open directory or file
OpenDat	Open DAT file
OpenErr	Open ERR file
ArchiveFileFolder	Archive directory or file
RestoreFileFolder	Restore directory or file
ArchiveAll	Archive all
RestoreAll	Restore all
ModulProc39	Internal identifier
Select	Select
SelectWithParam	Not currently supported
Run	Start
Cut	Cut
Copy	Copy
Paste	Paste
Duplicate	Duplicate
SelectAll	Select all
Format	Format disk
FileFilter	Filter
Attrib	Attribute
RefreshConfig	Reinitialize GUI
ModulProc82	Internal identifier

Each Navigator command corresponds to an internal identifier (e.g. “ArchiveAll” corresponds to “ModulProc32”). When a module accesses the Navigator, only the internal identifier with the corresponding parameters is transferred.

Description: Properties

The properties for the Navigator command are defined here.



Properties for: NewFileFolder

System conditions

☐ ProState0Sel
☐ ProState0Act
☐ ProState0Free
☐ ProState1Sel
☐ ProState1Act
☐ ProState1Free
☐ DriveKRC
☑ DriveArch
☑ DriveCD
☐ DriveOther
☑ ModeOpEXT
☐ ModeOpAUT
☐ ModeOpT1
☐ ModeOpT2
☑ ModeOpUNK

Item conditions

☐ ItemState0Sel
☐ ItemState0Act
☐ ItemState0Free
☐ ItemState1Sel
☐ ItemState1Act
☐ ItemState1Free
☐ SelSingle
☐ SelMulti
☐ SelNone
☐ EditModeFull
☐ EditModeProKor
☐ EditModeRO
☐ EditModeUnk
☐ DAT/SRC
☐ DAT/SRC/ERR

Ask user before run

☑ Default
☑ Demo
☐ User
☐ Expert
☐ Admin

Enable for user

☑ Default
☑ Demo
☑ User
☑ Expert
☑ Admin

Item condition: Submit program state SELECTED for item in the ITEM LIST

Modul proc for NewFileFolder is 10

Fig. 6-49: Methods tab, Properties

Element	Description
Properties for:	The method/Navigator command selected in **Available methods** is displayed.
Ask user before run	Check box active: Before the command is executed in this user group, a request for confirmation is displayed.
Enable for user	Check box active: This command is permissible for this user group.

The settings in the KRC Configurator are not checked for validity or plausibility.

It is possible, for example, to activate the object states **SelSingle**, **SelMulti** and **SelNone** for a Navigator command. In this case the Navigator command would be inactive, as at least one of these object states is active at any given time.

System conditions

Check box active: The Navigator command is **not** available if this system state is active.

Example:

- Available method: **NewFileFolder**
- System condition: check box **DriveArch** active

  This means: no directories or files can be created in the Navigator for the ARCHIVE drive.

System conditions	Description
ProState0Sel	Submit program with “SELECTED” state
ProState0Act	Submit program with “ACTIVE” state
ProState0Free	Submit program with “FREE” state
ProState1Sel	Robot program with “SELECTED” state
ProState1Act	Robot program with “ACTIVE” state
ProState1Free	Robot program with “FREE” state
DriveKRC	“KRC:\” drive
DriveArch	“ARCHIVE:\” drive
DriveCD	CD-ROM drive
DriveOther	Hard drive
ModeOpEXT	“Automatic External” mode
ModeOpAUT	“Automatic” mode
ModeOpT1	“T1” mode
ModeOpT2	“T2” mode
ModeOpUNK	“Unknown” mode
Disable	The Navigator is disabled as long as the selected command is active
SelTreeWindow	Focus in the directory structure
SelListWindow	Focus in the file list

Item conditions

Check box active: The Navigator command is **not** available if this object state is active.

Item conditions	Description
ItemState0Sel	Submit program in the file list with “SELECTED” state
ItemState0Act	Submit program in the file list with “ACTIVE” state
ItemState0Free	Submit program in the file list with “FREE” state
ItemState1Sel	Robot program in the file list with “SELECTED” state
ItemState1Act	Robot program in the file list with “ACTIVE” state
ItemState1Free	Robot program in the file list with “FREE” state
SelSingle	Only one object selected
SelMulti	Multiple objects selected
SelNone	No objects selected
EditModeFull	Edit mode “FULL”
EditModeProKor	Edit mode “PROKOR”
EditModeRO	Edit mode “READONLY”
EditModeUNK	Edit mode “UNKNOWN”
DAT / SRC	Object is a DAT or SRC file
DAT / SRC / ERR	Object is a DAT or SRC file containing errors
SRC	Object is an SRC file

Item conditions	Description
SRC / ERR	Object is an SRC file containing errors
DAT / SUB	Object is a DAT or SUB file
DAT / SUB / ERR	Object is a DAT or SUB file containing errors
SUB	Object is a SUB file
SUB / ERR	Object is a SUB file containing errors
DAT / ERR	Object is a DAT file containing errors
DAT	Object is a DAT file
TXT	Object is a text file
BIN	Object is a binary file
Arch	Object is an archive
Virt	Object is a virtual directory
Folder	Object is a directory
IBGN File	Object is an IBGN file
Protected File	Object is an encrypted and/or signed file

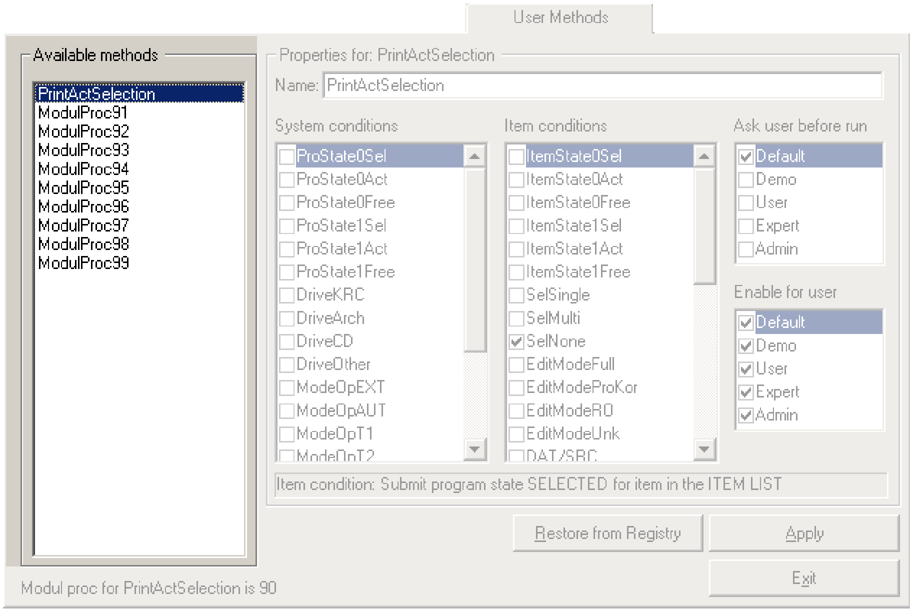
6.25.5 User Methods tab

Overview

The layout and function of this tab are largely identical to those of the **Methods** tab (>>> 6.25.4 "Methods tab" Page 197).

Description: Available methods

The user-specific Navigator command whose properties are to be defined is selected here. Precondition: The Navigator command must have been defined in the file MenueKeyKuka.ini.



User Methods

Available methods

PrintActSelection
ModulProc91
ModulProc92
ModulProc93
ModulProc94
ModulProc95
ModulProc96
ModulProc97
ModulProc98
ModulProc99

Properties for: PrintActSelection

Name: PrintActSelection

System conditions

□ ProState0Sel
□ ProState0Act
□ ProState0Free
□ ProState1Sel
□ ProState1Act
□ ProState1Free
□ DriveKRC
□ DriveArch
□ DriveCD
□ DriveOther
□ ModeOpEXT
□ ModeOpAUT
□ ModeOpT1
□ ModeOpT2

Item conditions

□ ItemState0Sel
□ ItemState0Act
□ ItemState0Free
□ ItemState1Sel
□ ItemState1Act
□ ItemState1Free
□ SelSingle
□ SelMulti
☑ SelNone
□ EditModeFull
□ EditModeProKor
□ EditModeRO
□ EditModeUnk
□ DAT/SRC

Ask user before run

☑ Default
□ Demo
□ User
□ Expert
□ Admin

Enable for user

☑ Default
☑ Demo
☑ User
☑ Expert
☑ Admin

Item condition: Submit program state SELECTED for item in the ITEM LIST

Modul proc for PrintActSelection is 90

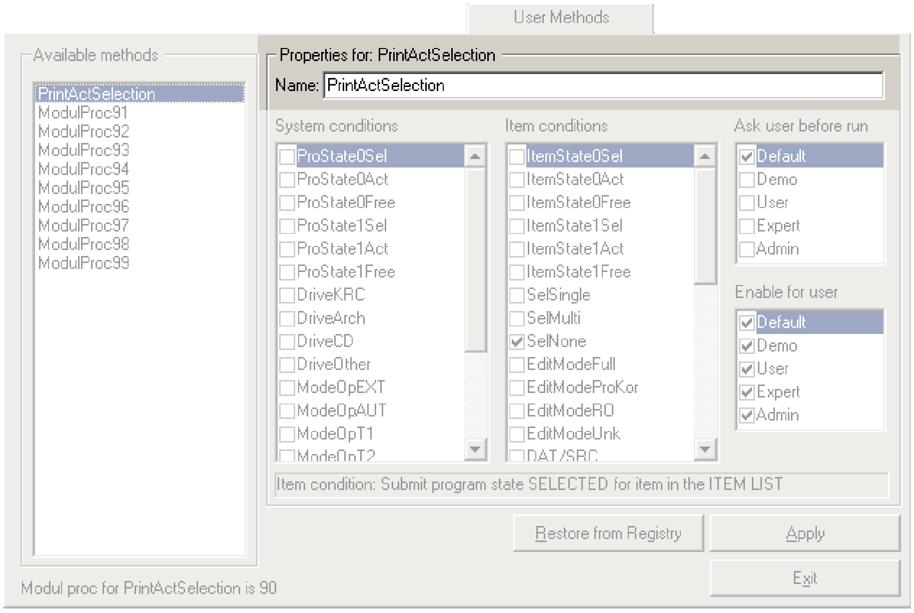
Fig. 6-50: User Methods tab, Available methods

Available methods	Description
PrintActSelection	Print the current selection.
ModulProc91	Internal identifier
ModulProc92	Internal identifier

Available methods	Description
ModulProc93	Internal identifier
ModulProc94	Internal identifier
ModulProc95	Internal identifier
ModulProc96	Internal identifier
ModulProc97	Internal identifier
ModulProc98	Internal identifier
ModulProc99	Internal identifier

Description: Properties

The properties for the Navigator command are defined here. A name can be assigned to the command.



Properties for: PrintActSelection

Name: PrintActSelection

System conditions: ☐ ProState0Sel, ☐ ProState0Act, ☐ ProState0Free, ☐ ProState1Sel, ☐ ProState1Act, ☐ ProState1Free, ☐ DriveKRC, ☐ DriveArch, ☐ DriveCD, ☐ DriveOther, ☐ ModeOpEXT, ☐ ModeOpAUT, ☐ ModeOpT1, ☐ ModeOpT2

Item conditions: ☐ ItemState0Sel, ☐ ItemState0Act, ☐ ItemState0Free, ☐ ItemState1Sel, ☐ ItemState1Act, ☐ ItemState1Free, ☐ SelSingle, ☐ SelMulti, ☑ SelNone, ☐ EditModeFull, ☐ EditModeProKor, ☐ EditModeRO, ☐ EditModeUnk, ☐ DAT/SRC

Ask user before run: ☑ Default, ☐ Demo, ☐ User, ☐ Expert, ☐ Admin

Enable for user: ☑ Default, ☑ Demo, ☑ User, ☑ Expert, ☑ Admin

Item condition: Submit program state SELECTED for item in the ITEM LIST

Modul proc for PrintActSelection is 90

Fig. 6-51: User Methods tab, Properties

Element	Description
Properties for:	The method/Navigator command selected in **Available methods** is displayed.
Name	Designation for the method/Navigator command

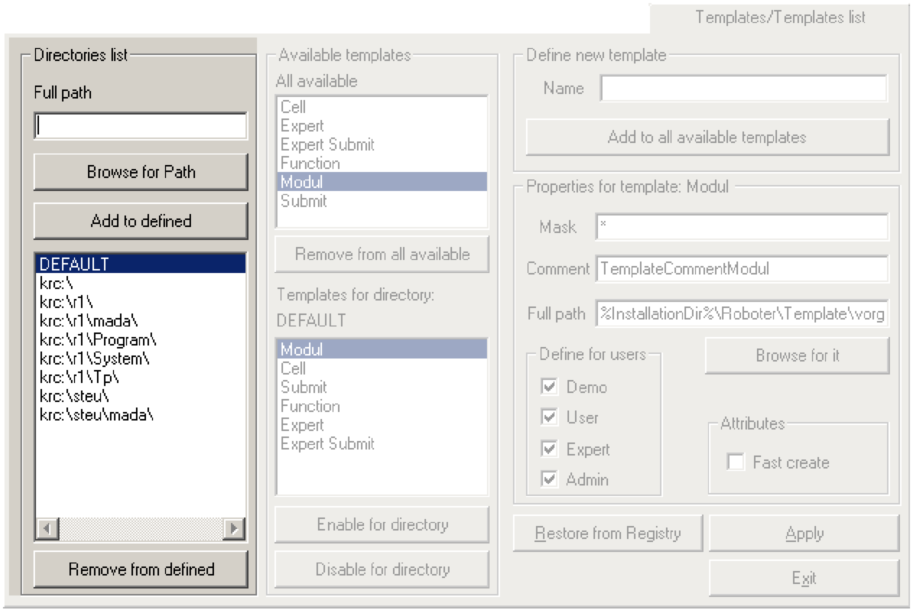
6.25.6 Templates/Templates list tab

Overview

The templates used for creating a new object (e.g. module, cell, …) can be configured here.

Description: Directories list

The paths for which specific templates are to be available are specified here.



Path list: DEFAULT, krc:\, krc:\r1\, krc:\r1\mada\, krc:\r1\Program\, krc:\r1\System\, krc:\r1\Tp\, krc:\steu\, krc:\steu\mada\

All available: Cell, Expert, Expert Submit, Function, Modul, Submit

Templates for directory: DEFAULT — Modul, Cell, Submit, Function, Expert, Expert Submit

Mask: *

Comment: TemplateCommentModul

Full path: %InstallationDir%\Roboter\Template\vorg

☑ Demo ☑ User ☑ Expert ☑ Admin

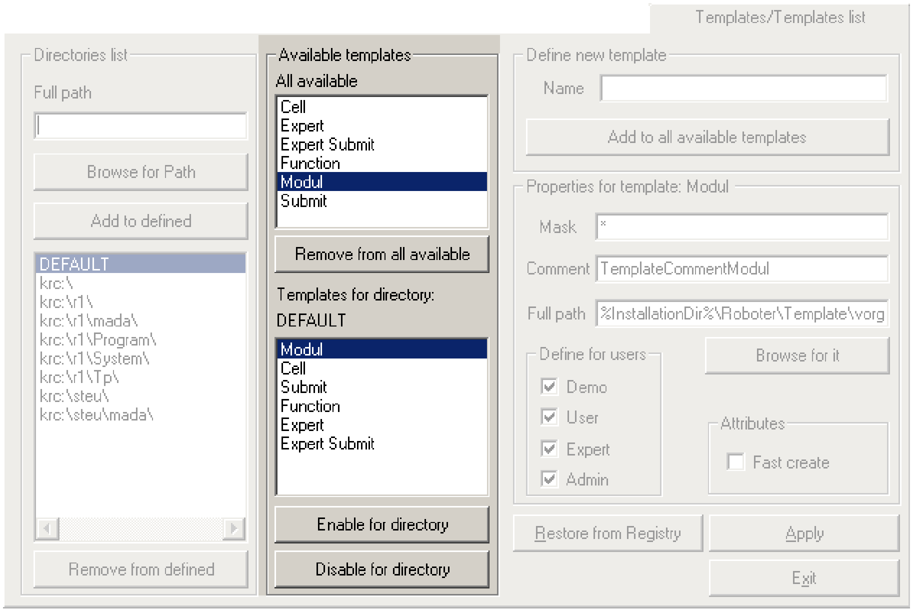
□ Fast create

Fig. 6-52: Templates tab, Directories list

Element	Description
Full path	Path for which templates are to be used. The path can be entered manually or selected via **Browse for Path**.
Add to defined	Adds the path from the **Full path** box to the path list.
Path list	The path to which templates are to be assigned must be selected in this list. **DEFAULT**: covers all paths that are not specifically included in the list.
Remove from defined	Removes the selected path from the path list.

Description: Available templates

Templates are assigned here to a path.



Path list: DEFAULT, krc:\, krc:\r1\, krc:\r1\mada\, krc:\r1\Program\, krc:\r1\System\, krc:\r1\Tp\, krc:\steu\, krc:\steu\mada\

All available: Cell, Expert, Expert Submit, Function, Modul, Submit

Templates for directory: DEFAULT — Modul, Cell, Submit, Function, Expert, Expert Submit

Mask: *

Comment: TemplateCommentModul

Full path: %InstallationDir%\Roboter\Template\vorg

☑ Demo ☑ User ☑ Expert ☑ Admin

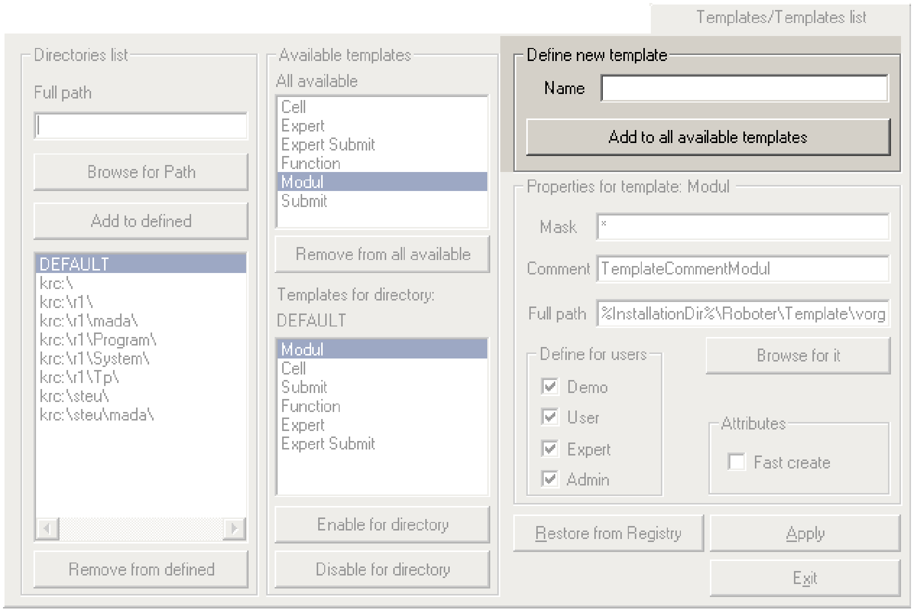
□ Fast create

Fig. 6-53: Templates tab, Available templates

Element	Description
All available	Available templates
Remove from all available	Removes the selected template from **All available**.
Templates for directory:	Templates assigned to the path selected in the path list. If there is no entry here, the **New** softkey is not available.
Enable for directory	Adds the template selected in **All available** to **Templates for directory**.
Disable for directory	Removes the template selected in **Templates for directory**.

Description: Define new template

New templates can be defined here.



Directories list: DEFAULT, krc:\, krc:\r1\, krc:\r1\mada\, krc:\r1\Program\, krc:\r1\System\, krc:\r1\Tp\, krc:\steu\, krc:\steu\mada\

All available: Cell, Expert, Expert Submit, Function, Modul, Submit

Templates for directory: DEFAULT: Modul, Cell, Submit, Function, Expert, Expert Submit

Mask: *

Comment: TemplateCommentModul

Full path: %InstallationDir%\Roboter\Template\vorg

Define for users: ☑ Demo ☑ User ☑ Expert ☑ Admin

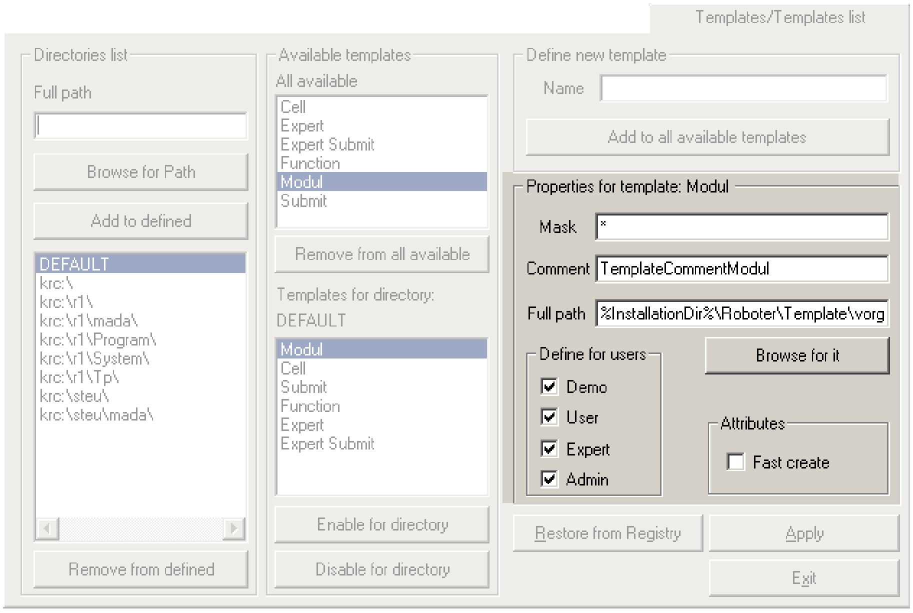
Attributes: ☐ Fast create

Fig. 6-54: Templates tab, Define new template

Element	Description
Name	Name of the new template
Add to all available templates	Adds the template to **All available**.

Description: Properties for template

The properties for a template are defined here.



Full path:

- DEFAULT
- krc:\
- krc:\r1\
- krc:\r1\mada\
- krc:\r1\Program\
- krc:\r1\System\
- krc:\r1\Tp\
- krc:\steu\
- krc:\steu\mada\

All available:

- Cell
- Expert
- Expert Submit
- Function
- Modul
- Submit

Templates for directory: DEFAULT

- Modul
- Cell
- Submit
- Function
- Expert
- Expert Submit

Name:

Properties for template: Modul

Mask: *

Comment: TemplateCommentModul

Full path: %InstallationDir%\Roboter\Template\vorg

Define for users: ☑ Demo ☑ User ☑ Expert ☑ Admin

Attributes: ☐ Fast create

Fig. 6-55: Templates tab, Properties for template

Element	Description
Properties for template:	The template selected in **All available** is displayed.
Mask	Defines which character string is permissible as the name for the template. Examples: *: all characters are permissible. *Temp99[1-99]*: only the names *Temp1* to *Temp99* are permissible.
Comment	Database key for the template User-specific templates: When a template is selected in the Navigator, the database key is displayed as a comment (alongside the name of the template). Templates that are defined by default in the KSS: When a template is selected in the Navigator, the translation of the database key is displayed as a comment.
Full path	File path of the template. The path can be entered manually or selected via **Browse for it**.
Define for users	User group for which the template is to be available.
Attributes	This function is not currently supported.

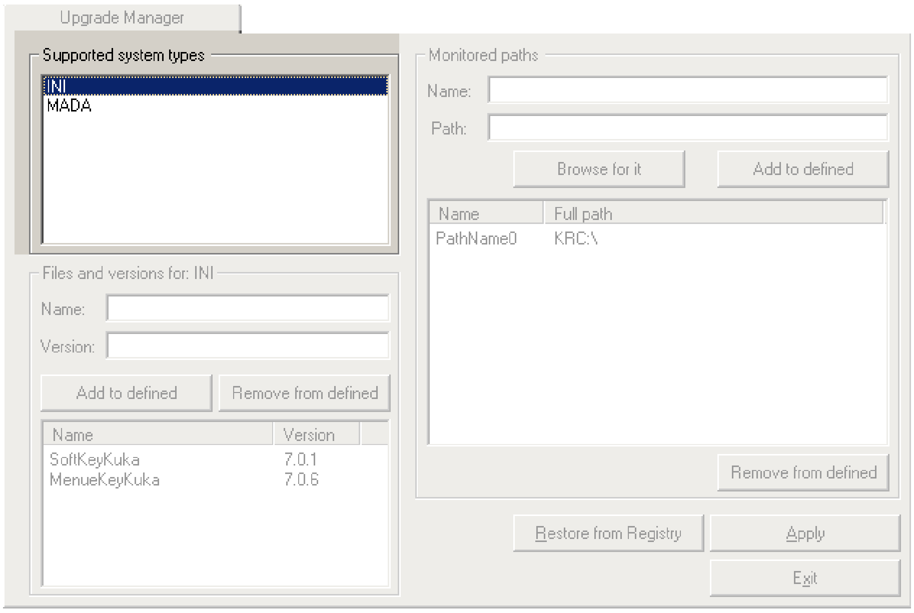
6.25.7 Upgrade Manager tab

Overview

Here the user can define monitoring functions that check, when files are added, whether it is permissible to add files to the path in question and verify the file version.

> The monitoring functions do **not** refer to a regular KSS upgrade. During this procedure they are not active.
> The monitoring functions are active during the system runtime and serve to monitor the manual addition of files.

Description: Supported system types



Upgrade Manager

Supported system types: INI, MADA

Files and versions for: INI

Name:

Version:

Name	Version
SoftKeyKuka	7.0.1
MenueKeyKuka	7.0.6

Monitored paths

Name:

Path:

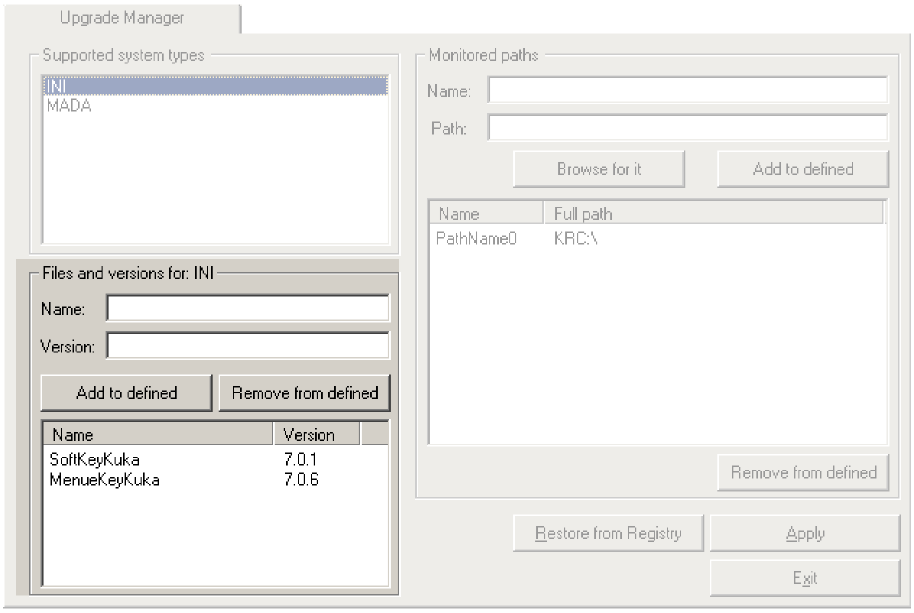
Name	Full path
PathName0	KRC:\

Fig. 6-56: Upgrade Manager tab, Supported system types

Element	Description
Supported system types	List of the file types that can be monitored. The following types are available: ■ **INI**: INI files ■ **MADA**: machine data ■ **Techpack**: files belonging to technology packages

Description: Files and versions

Adds a selected file to the current file type or removes it.



Upgrade Manager

Supported system types: INI, MADA

Files and versions for: INI

Name:

Version:

Name	Version
SoftKeyKuka	7.0.1
MenueKeyKuka	7.0.6

Monitored paths

Name:

Path:

Name	Full path
PathName0	KRC:\

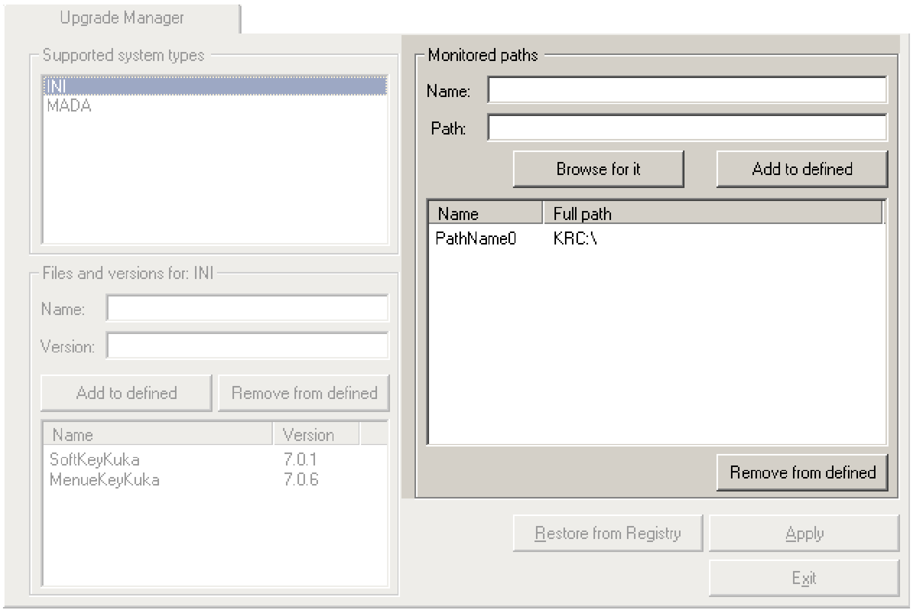
Fig. 6-57: Upgrade Manager tab, Files and versions

Element	Description
Files and versions for:	The file type selected in **Supported system types** is displayed.
Name	Name of the file to be monitored (without file extension)

Element	Description
Version	Version identifier of the file
Add to defined	Adds the file with the corresponding version identifier to the list of files to be monitored.
Remove from defined	Removes the selected file from the list.
File list	List of the files to be monitored in the current file type.

Description: Monitored paths

The paths to be monitored by the Upgrade Manager are defined here.



Name	Version
SoftKeyKuka	7.0.1
MenueKeyKuka	7.0.6

Name	Full path
PathName0	KRC:\

Fig. 6-58: Upgrade Manager tab, Monitored paths

Element	Description
Name	Symbolic name of the path to be monitored
Path	Path to be monitored. The path can be entered manually or selected via **Browse for it**.
Add to defined	Adds the path to the list of monitored paths
Path list	List of monitored paths
Remove from defined	Removes the selected path from the list

6.25.8 Archive Manager tab

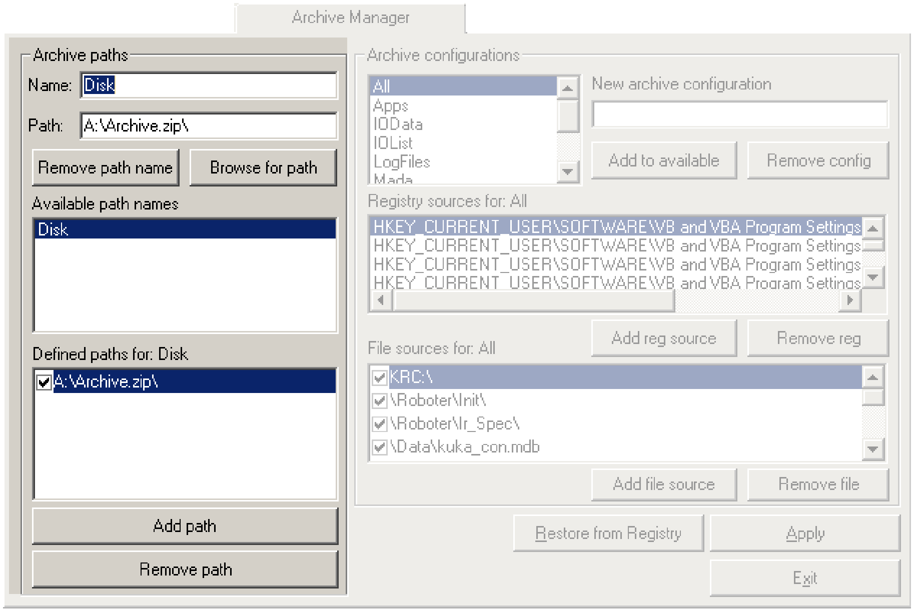
Overview

The settings for archiving and restoring files are defined here.

> Additional configuration options for archiving and restoring files can be found in the **History Info** tab (>>> 6.25.9 "History Info tab" Page 209).

Description: Archive paths

A symbolic name is defined here for the path to which files are to be archived. Symbolic names are displayed in the Navigator. Multiple paths can be grouped together under a single symbolic name.



Name: Disk

Path: A:\Archive.zip\

Available path names

- Disk

Defined paths for: Disk

- ☑ A:\Archive.zip\

Archive configurations

- All
- Apps
- IOData
- IOList
- LogFiles
- Mada

Registry sources for: All

- HKEY_CURRENT_USER\SOFTWARE\VB and VBA Program Settings
- HKEY_CURRENT_USER\SOFTWARE\VB and VBA Program Settings
- HKEY_CURRENT_USER\SOFTWARE\VB and VBA Program Settings
- HKEY_CURRENT_USER\SOFTWARE\VB and VBA Program Settings

File sources for: All

- ☑ KRC:\
- ☑ \Roboter\Init\
- ☑ \Roboter\Ir_Spec\
- ☑ \Data\kuka_con.mdb

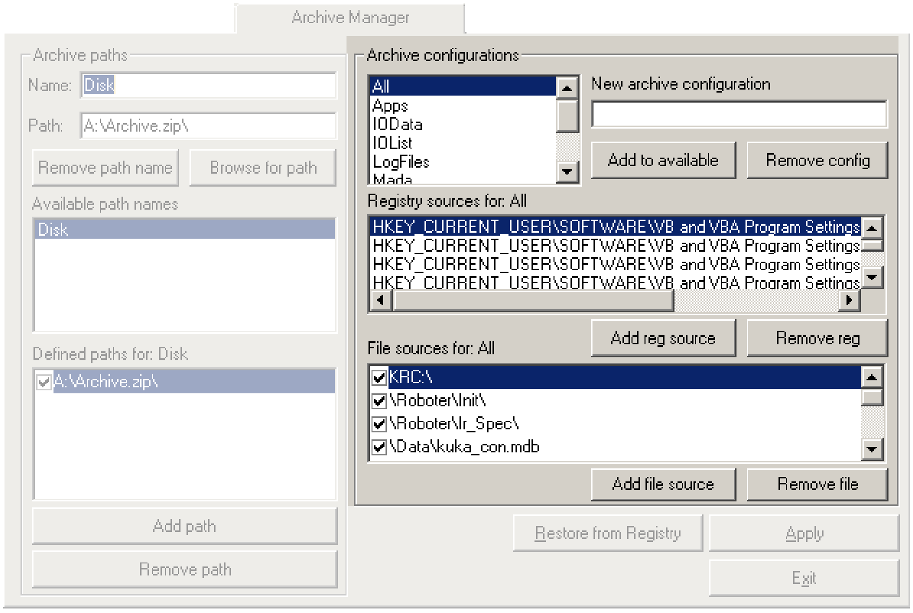
Fig. 6-59: Archive Manager tab, Archive paths

Element	Description
Name	Defined a symbolic name (symbolic names can be used, for example, in the file MenueKey.ini). A symbolic name can stand for one or more paths.
Path	Path that is assigned to this symbolic name. The path can be entered manually or selected via **Browse for path**.
Remove path name	Removes the symbolic name selected in **Available path names**.
Available path names	List of symbolic names
Defined paths for:	Path that is assigned to this symbolic name. Multiple paths can be assigned to a single symbolic name. In this case, files are archived to each of these paths. Check box active: This path is the default path. The default path must be physically present when archiving is carried out. If not, archiving is not carried out, not even to the other paths. If a non-default path is not physically present, this has no effect on archiving to the other paths. When archived files are restored, the system accesses the default path. Recommendation: even if only one path is specified, define it as a default path.
Add path	Adds the symbolic name entered in **Name** to **Available path names**. Simultaneously adds the path entered in **Path** to **Defined paths for**. In this way, multiple paths can be assigned to a single symbolic name.
Remove path	Removes the path selected in **Defined paths for**.

The file extension ".zip" in the path name (e.g. "Archive.zip") generates a Zip file. Otherwise, the files are simply copied to the defined paths.

Description: Archive configurations

Here the user defines which data are to be archived for the individual archive configurations. The archive configurations correspond to the menu items under **File** > **Archive**.



Name: Disk

Path: A:\Archive.zip\

Available path names: Disk

Defined paths for: Disk

☑ A:\Archive.zip\

Archive configurations: All, Apps, IOData, IOList, LogFiles, Mada

New archive configuration:

Registry sources for: All

HKEY_CURRENT_USER\SOFTWARE\VB and VBA Program Settings
HKEY_CURRENT_USER\SOFTWARE\VB and VBA Program Settings
HKEY_CURRENT_USER\SOFTWARE\VB and VBA Program Settings
HKEY_CURRENT_USER\SOFTWARE\VB and VBA Program Settings

File sources for: All

☑ KRC:\
☑ \Roboter\Init\
☑ \Roboter\Ir_Spec\
☑ \Data\kuka_con.mdb

Fig. 6-60: Archive Manager tab, Archive configurations

Element	Description
Archive configurations	Existing archive configurations
New archive configuration	Name for a new archive configuration
Add to available	Adds a new archive configuration to the existing ones.
Remove config	Deletes the configuration selected under **Archive configurations**.
Registry sources for:	Registry database branches to be archived/ restored
Add reg source	Adds a branch to **Registry sources for.** A window opens in which a branch can be selected or entered manually. Only registry database branches that are relevant to the KUKA System Software can be added. This is checked by the KRC Configurator. For other branches, archiving and restoring would be too time-consuming.
Remove reg source	Removes the selected entry from **Registry sources for**.

Element	Description
File sources for:	Check box active: The file or directory is archived. Check box not active: The file or directory is not archived. **Note:** If a deactivated check box refers to a subset of an active check box, this results in an exclusion. Thus: archive set A (= active check box), but without subset B (= deactivated check box).
Add file source	Adds a directory to **File sources for**. A window opens in which a path can be selected or entered manually. In the case of manual entry: ■ The following conventions apply: Example for individual files: "\Data\KUKA_CON.MDB" Example for complete directories, including subdirectories: "\ROBOTER\INIT\" Example for files with a filter: "Hugo*.*" ■ The user can define search filters, e.g. "myFiles*.bkp".
Remove file	Removes the directory selected in **File sources for**.

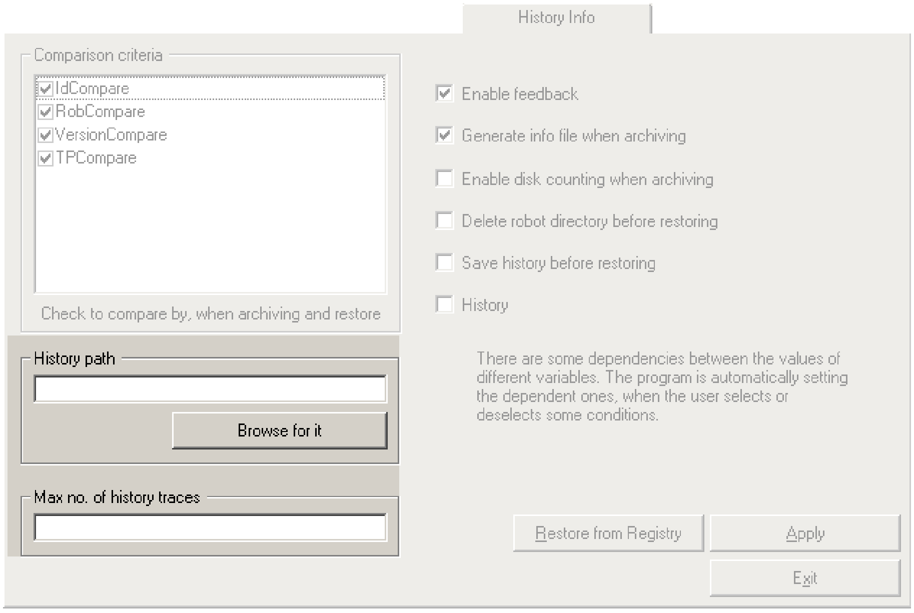
6.25.9 History Info tab

Overview

This tab is used to define whether, during archiving, the data should also be backed up on the hard drive (history trace). In this way, the data can still be restored if the archive is lost or damaged.

Other archiving settings can also be defined.

Description: History



☑ IdCompare
☑ RobCompare
☑ VersionCompare
☑ TPCompare

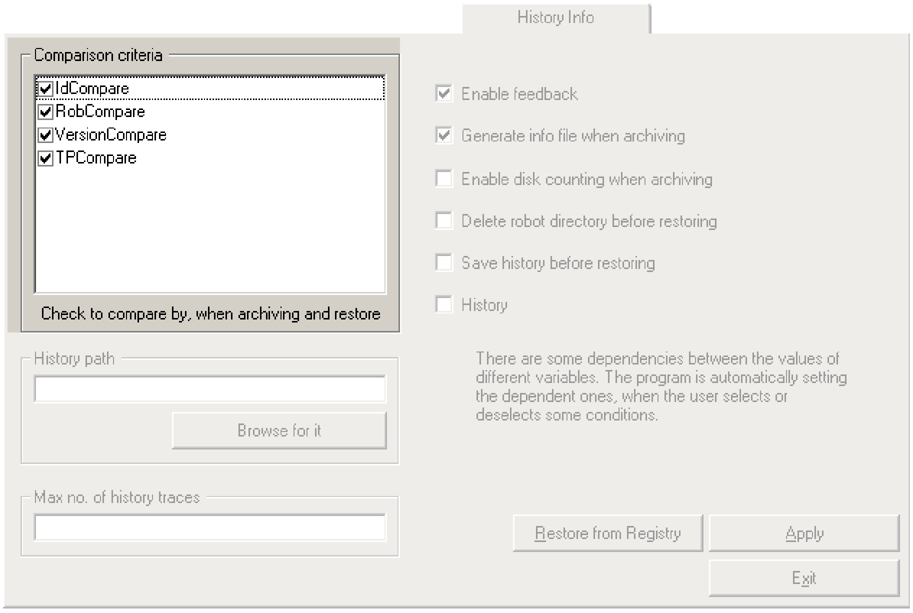
☑ Enable feedback
☑ Generate info file when archiving
□ Enable disk counting when archiving
□ Delete robot directory before restoring
□ Save history before restoring
□ History

There are some dependencies between the values of different variables. The program is automatically setting the dependent ones, when the user selects or deselects some conditions.

Fig. 6-61: History Info tab, History

Element	Description
History path	Path on the hard drive to which the history data are archived. The path can be entered manually or selected via **Browse for it**.
Max no. of history traces	Maximum number of history traces on the hard drive. Each subsequent trace overwrites the oldest existing trace.

Description: Comparison criteria



History Info

Comparison criteria

☑ IdCompare
☑ RobCompare
☑ VersionCompare
☑ TPCompare

Check to compare by, when archiving and restore

History path

Max no. of history traces

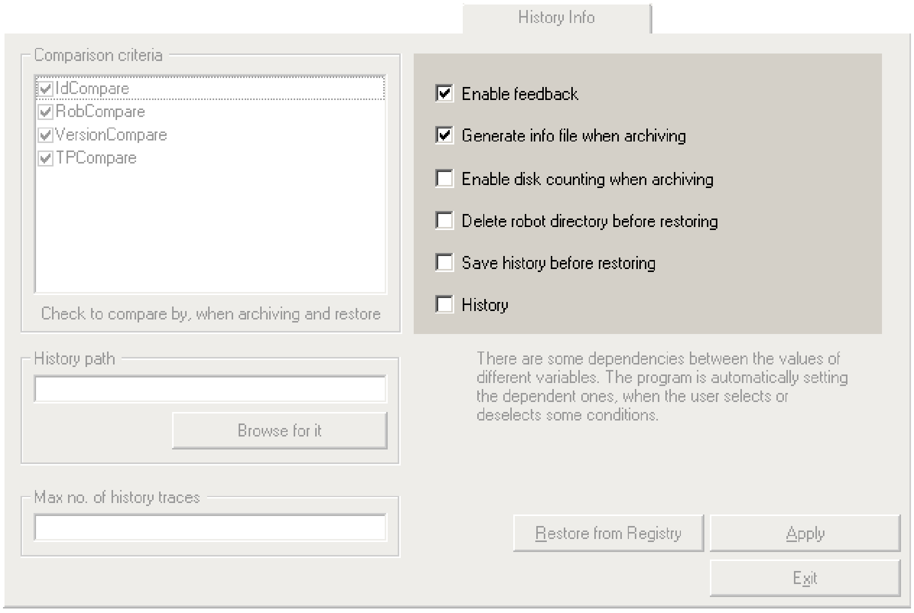
☑ Enable feedback
☑ Generate info file when archiving
□ Enable disk counting when archiving
□ Delete robot directory before restoring
□ Save history before restoring
□ History

There are some dependencies between the values of different variables. The program is automatically setting the dependent ones, when the user selects or deselects some conditions.

Fig. 6-62: History Info tab, Comparison criteria

Element	Description
Comparison criteria	Comparison criteria to be checked when restoring data. Check box active: In the case of a deviation from this criterion when restoring data, a request for confirmation is displayed. The following criteria are available: ■ **IdCompare**: archive version ■ **RobCompare**: robot name and serial number ■ **VersionCompare**: KSS version ■ **TPCompare**: technology packages

Description of further settings



☑ IdCompare
☑ RobCompare
☑ VersionCompare
☑ TPCompare

☑ Enable feedback
☑ Generate info file when archiving
□ Enable disk counting when archiving
□ Delete robot directory before restoring
□ Save history before restoring
□ History

There are some dependencies between the values of different variables. The program is automatically setting the dependent ones, when the user selects or deselects some conditions.

Fig. 6-63: History Info tab, further settings

Element	Description
Enable Feedback	Check box active: The Archive Manager generates dialog messages. In the case of an unfulfilled comparison criterion, for example, the is asked whether the process should nonetheless be continued. Otherwise, the process is terminated without a message.
Generate info file when archiving	Check box active: The file AMI.INI is created during archiving. The file contains information about the robot name, archive ID, etc. This setting is also required if data are to be archived to multiple storage media.
Enable disk counting when archiving	Check box active: This setting is required if data are to be archived to multiple storage media.
Delete robot directory before restoring	Before the data are restored, the directory KRC:\R1 and all its subdirectories are deleted. Automatically activates the settings **Save history before restoring** and **History**.
Save history before restoring	Archives the current data before restoring archived data. Automatically activates the setting **History**.
History	During archiving, a copy is saved to the specified history path.

6.25.10 General/Folder Layout tab

Overview

Here the user defines the directories in the Navigator in which subdirectories may be created or deleted.

Description: Predefined tree

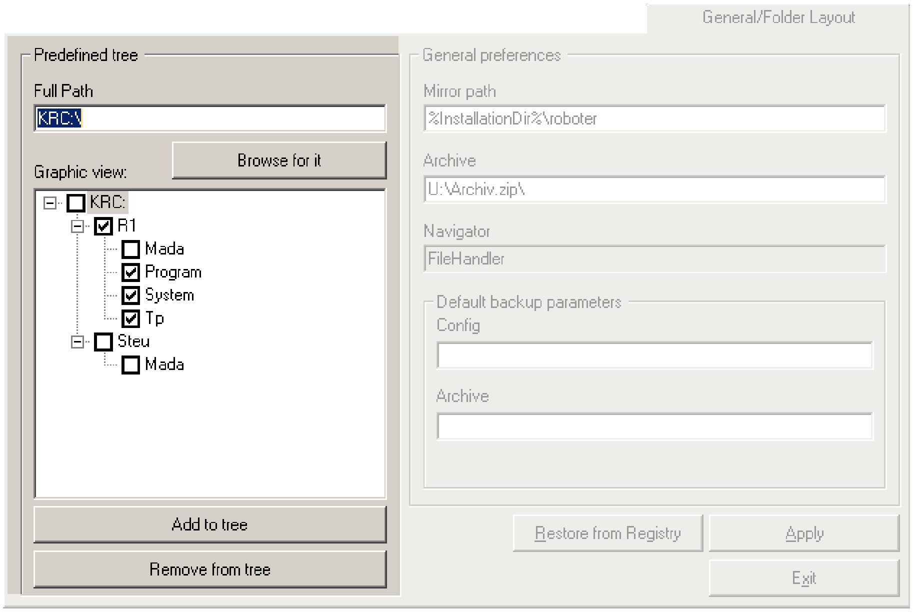
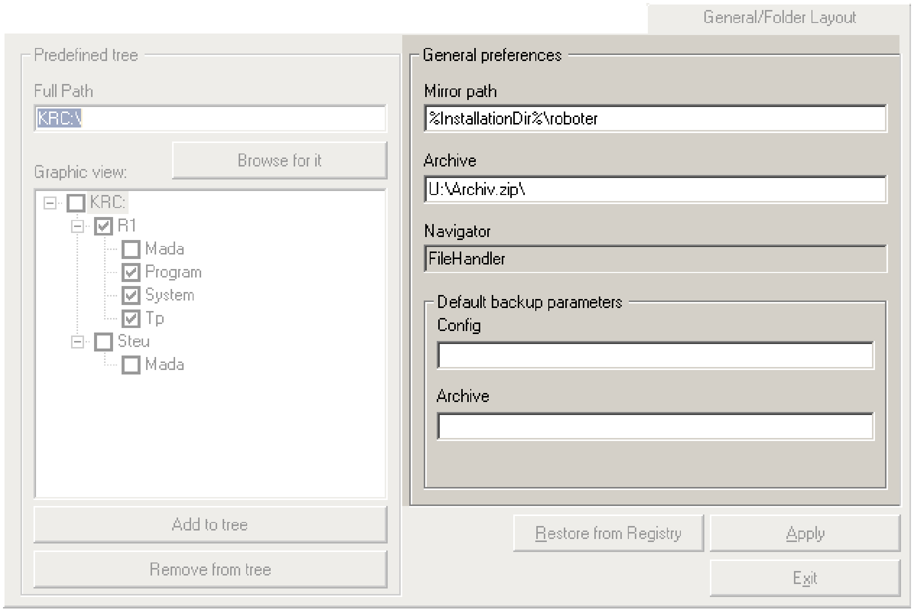


Fig. 6-64: General/Folder Layout tab, Predefined tree

Element	Description
Full path	Directory to be displayed in **Graphic view**. The directory can be entered manually or selected via **Browse for it**. The path KRC:\ represents %INSTALLATION-DIR%\KRC\Roboter\KRC.
Graphic view	Directory structure as in the Navigator Check box active: Subdirectories may be created and deleted under this node
Add to tree	Displays the directory entered in **Full Path** in **Graphic view**.
Remove from tree	Removes the directory selected in **Graphic view** from the display. A directory that is not displayed is not monitored. The creation and deletion of subdirectories is thus possible. The directory is only removed from this display. It is not removed from the directory structure in the Navigator itself.

Description: General preferences

This function is not currently supported.



Full Path: KRC:\

Graphic view:

- □ KRC:
  - ☑ R1
    - □ Mada
    - ☑ Program
    - ☑ System
    - ☑ Tp
  - □ Steu
    - □ Mada

Mirror path: %InstallationDir%\roboter

Archive: U:\Archiv.zip\

Navigator: FileHandler

Config:

Archive:

Fig. 6-65: General/Folder Layout tab, General preferences

7 System variables

7.1 Variables

These variables can be monitored in the KRL program or viewed via the variable display.

Position variables

Variable	Description
$POS_ACT	Type: E6POS ■ Robot under position control: variable indicates the programmed Cartesian position. ■ Robot with stiffness control or gravitation compensation activated: variable indicates the currently measured Cartesian position.
$POS_ACT_MES	Type: E6POS Variable always indicates the currently measured Cartesian position.
$POS_ACT_CMD	Type: E6POS Variable always indicates the programmed Cartesian position.
$POS_DELTA_ACT	Type: E6POS Cartesian difference between programmed and measured Cartesian position: $POS_DELTA_ACT = $POS_ACT_CMD - $POS_ACT_MES Only relevant if $STIFFNESS.FRAMETYPE=#TOOL
$AXIS_ACT	Type: E6AXIS ■ Robot under position control: variable indicates the programmed axis position. ■ Robot with stiffness control or gravitation compensation activated: variable indicates the currently measured axis position.
$AXIS_ACT_MES	Type: E6AXIS Variable always indicates the currently measured axis position.
$AXIS_ACT_CMD	Type: E6AXIS Variable always indicates the programmed axis position.

Torque variables

Variable	Description
$TORQUE_AXIS_ACT	Type: E6POS Variable indicates the current axis-specific torque.
$TORQUE_TCP_EST	(>>> 6.4 "Back estimation of the force measurement" Page 126)

Variable	Description
$TORQUE_AXIS_EST	Type: E6POS Variable indicates the current external axis-specific torque, i.e. the torque caused by an external force pulse.
$TORQUE_AXIS_RATIO	Type: REAL Variable indicates the current percentage value of the external axis-specific torque. ■ **0.0 to 1.0** (1.0 = 100%) The percentage value refers to the maximum torque of the axes. If the percentage value exceeds the value set for TRIGBYCONTACT, TRIGBYCONTACT is activated.

Controller variables

Variable	Description
$STIFF_STRAT_ACT	Type: INT Variable indicates the current controller strategy. This variable can be used to program an interrupt for TRIGBYCONTACT, e.g. to stop the motion to the contact point. ■ **10**: Position controller ■ **20**: Cartesian stiffness controller ■ **30**: Axis-specific stiffness controller ■ **101**: Gravity compensation
$STIFF_TYPE_ACT	Type: ENUM Variable indicates the currently set controller type. ■ **#POSITION**: Position controller ■ **#CART_IMP**: Cartesian stiffness controller ■ **#JNT_IMP**: Axis-specific stiffness controller ■ **#GRAVCOMP**: Gravity compensation

7.2 I/Os for signaling states

The following signals can be used for programming interrupts or in the Submit interpreter (SPS.SUB).

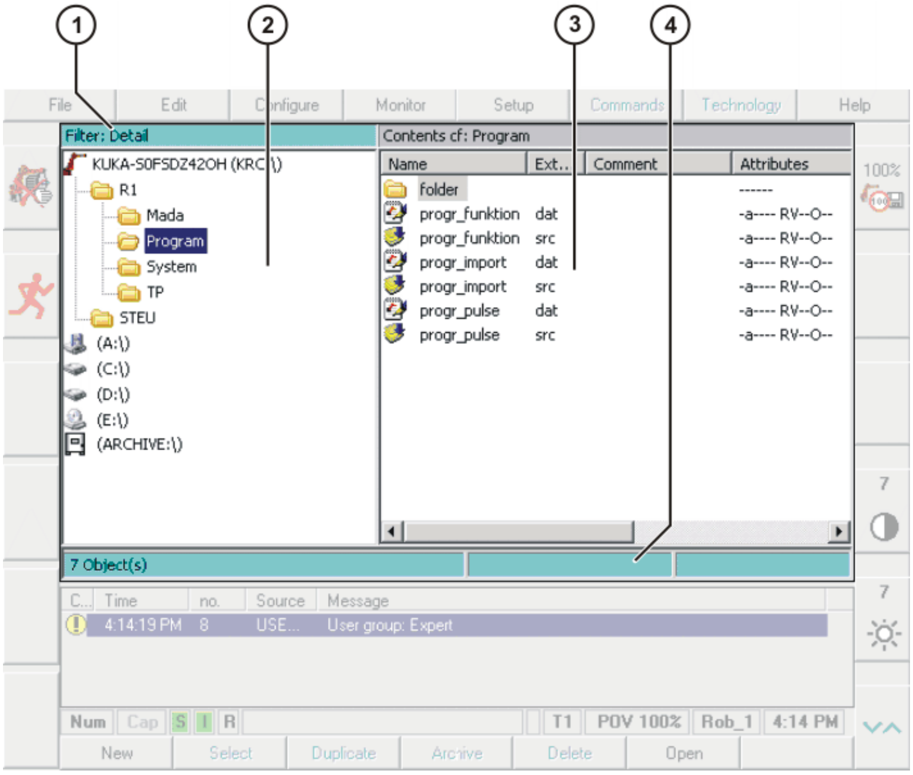
I/O	Description
$OUT[303]	Cartesian velocity of 250 mm/s has been exceeded.
$OUT[304]	Limitation of the Cartesian deviation from the path is active ($STIFFNESS.CPMAXDELTA).
$OUT[305]	Force limitation of the Cartesian stiffness controller is active ($STIFFNESS.MAXFORCE).
$OUT[306]	Limitation of the axis-specific deviation is active ($STIFFNESS.AXISMAXDELTA).
$OUT[307]	Force limitation of the axis-specific stiffness controller is active ($STIFFNESS.AXISMAXDELTATRQ).
$OUT[308]	Force activation with DESIREDFORCE is active

I/O	Description
$OUT[309]	Maximum deflection with DESIREDFORCE has been reached.
$OUT[310]	Maximum velocity with DESIREDFORCE has been reached.
$OUT[311]	Gravitation vector does not match current torque measurement, i.e. the load data $LOAD or the gravitation vector $GRAVITATION[] have not been entered correctly.

8 Program management

8.1 Navigator file manager

Overview



Name	Ext..	Comment	Attributes
folder			------
progr_funktion	dat		-a---- RV--O--
progr_funktion	src		-a---- RV--O--
progr_import	dat		-a---- RV--O--
progr_import	src		-a---- RV--O--
progr_pulse	dat		-a---- RV--O--
progr_pulse	src		-a---- RV--O--

C...	Time	no.	Source	Message
!	4:14:19 PM	8	USE...	User group: Expert

Fig. 8-1: Navigator

1	Header	3	File list
2	Directory structure	4	Status bar

Description

In the Navigator, the user manages programs and system-specific files.

Header

- Left-hand area: the selected filter is displayed.
  (>>> 8.1.1 "Selecting filters" Page 220)
- Right-hand area: the directory or drive selected in the directory structure is displayed.

Directory structure

Overview of directories and drives. Exactly which directories and drives are displayed depends on the user group and configuration.

File list

The contents of the directory or drive selected in the directory structure are displayed. The manner in which programs are displayed depends on the selected filter.

The file list has the following columns:

Column	Description
Name	Directory or file name
Extension	File extension This column is not displayed in the user group “User”.
Comment	Comment

Column	Description
Attributes	Attributes of the operating system and kernel system This column is not displayed in the user group “User”.
Size	File size in kilobytes This column is not displayed in the user group “User”.
#	Number of changes made to the file
Changed	Date and time of the last change
Created	Date and time of file creation This column is not displayed in the user group “User”.

The user can scroll left and right in the file list using the keyboard shortcuts SHIFT+RIGHT ARROW or SHIFT+LEFT ARROW.

Pop-up menus are available for the objects in the file list. Calling the pop-up menu: select object(s) and press the RIGHT ARROW key.

Status bar

The status bar can display the following information:

- Selected objects
- Action in progress
- User dialogs
- User entry prompts
- Requests for confirmation

8.1.1 Selecting filters

Description

This function is not available in the user group “User”.

The filter defines how programs are displayed in the file list. The following filters are available:

- **Detail**
  Programs are displayed as SRC and DAT files. (Default setting)
- **Modules**
  Programs are displayed as modules.

Procedure

1. Select the menu sequence **File** > **Filter**.
2. Select the desired filter in the left-hand section of the Navigator.
3. Confirm with **OK**.

8.1.2 Displaying or modifying file properties

Precondition

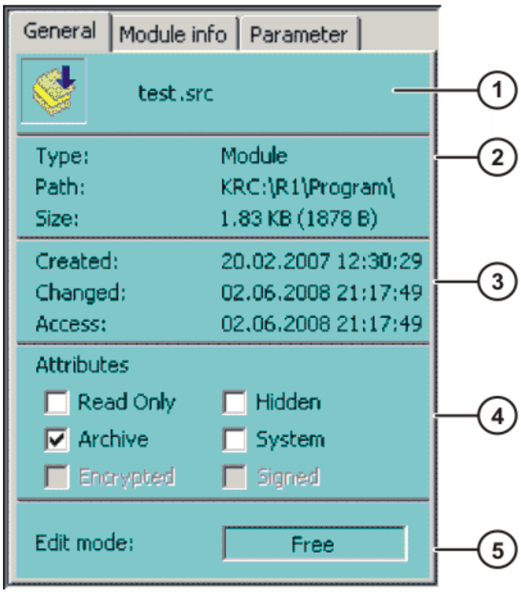
- To change properties: user group “Expert”.

Procedure

1. Select the object in the directory structure or in the file list.
2. Select the menu sequence **File** > **Attributes**.
3. Change the properties and save the change by pressing the **OK** softkey.

Description

General



test.src

Type: Module
Path: KRC:\R1\Program\
Size: 1.83 KB (1878 B)

Created: 20.02.2007 12:30:29
Changed: 02.06.2008 21:17:49
Access: 02.06.2008 21:17:49

Attributes

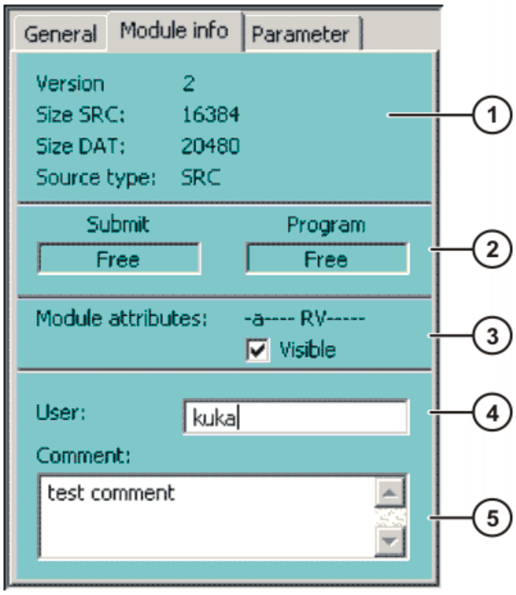
□ Read Only □ Hidden
☑ Archive □ System
□ Encrypted □ Signed

Edit mode: Free

Fig. 8-2: File properties, “General” tab

Item	Description
1	Name of the selected file
2	File type, path and file size. File types: ■ **Module**: module ■ **Dir**: directory ■ **Archiv**: archive file ■ **Bin**: binary file ■ **Text**: text file ■ **VirtualDir**: virtual directory ■ **Unknown**: all other file types
3	Windows file properties
4	Windows file properties. The properties can be modified in the user group “Expert”.
5	**Free**: the file is not selected in the KUKA.HMI and is not open. **Full**: the file is open in the KUKA.HMI. **ProKor**: the file is selected in the KUKA.HMI.

Description Module info



Version 2
Size SRC: 16384
Size DAT: 20480
Source type: SRC

Submit: Free
Program: Free

Module attributes: -a---- RV-----
☑ Visible

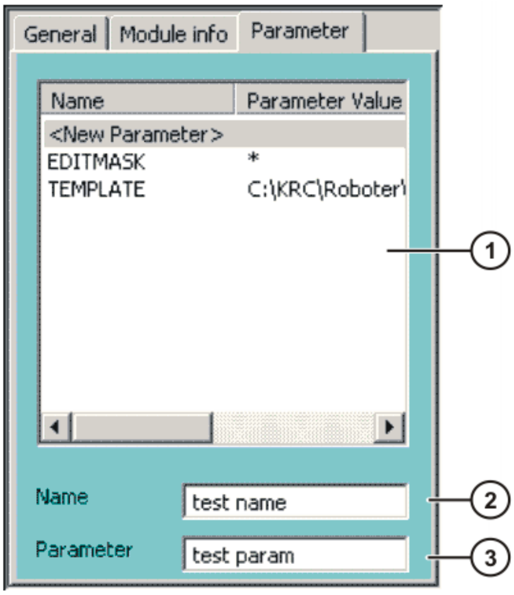
User: kuka

Comment:
test comment

Fig. 8-3: File properties, "Module info" tab

Item	Description
1	**Version**: internal version number of the file. After creation, the file does not yet have a number. After the first change, the file receives the number 1. The number is incremented after every change. **Size SRC**: size of the SRC file **Size DAT**: size of the DAT file **Source type**: source type ■ **SRC**: SRC file ■ **SubmitSub**: SUB file ■ **None**: all other file types, e.g. DAT file
2	Status of the module in the Submit interpreter and in the robot interpreter **Free**: program is not selected. **Selected**: program is selected. **Active**: only relevant for the **Submit** box. This program is currently being used by the Submit interpreter.
3	Check box active: if this program is called as a subprogram, it is displayed in the Editor. Check box not active: if this program is called as a subprogram, it is not displayed in the Editor. This program cannot be selected manually.
4	The user can enter his name here (max. 30 characters).
5	The user can enter a comment for the module here. It is possible to scroll up and down in the comment using the UP ARROW and DOWN ARROW keys. The comment is displayed in the **Comment** column in the Navigator.

Description Parameter



Name	Parameter Value
<New Parameter>	
EDITMASK	*
TEMPLATE	C:\KRC\Roboter\

Name: test name

Parameter: test param

Fig. 8-4: File properties, “Parameter” tab

Any desired information can be stored in KRL modules.

Item	Description
1	The existing information is shown here. Information can be deleted by selecting the line, deleting the contents of the **Parameter** box and pressing the **OK** softkey.
2	The user can enter a name here for a new piece of information.
3	The user can enter information here.

Program in the editor

If a SRC or DAT file is opened in an editor (e.g. WordPad) in Windows, a number of the file properties are displayed above the DEF line.

In KUKA.HMI, the file properties are not visible here.

```
1  &ACCESS RV
2  &REL 2
3  &COMMENT test comment
4  &USER kuka
5  &PARAM test name = test param
6  &PARAM TEMPLATE = C:\KRC\Roboter\Template\vorgabe
7  &PARAM EDITMASK = *
8  DEF test( )
…
```

Line	Description
1	**Module info** tab, check box **Visible** - `&ACCESS RV` = check box active - `&ACCESS R` = check box inactive
2	**Module info** tab, **Version** box
3	**Module info** tab, **Comment** box
4	**Module info** tab, **User** box
5	**Parameter** tab, **Name** and **Parameter** boxes

8.1.3 Icons in the Navigator

Drives:

Icon	Description	Default path
🦾	Robot	KRC:\
💾	Floppy disk	A:\
🖴	Hard disk	e.g. "KUKADISK (C:\)" or "KUKADATA (D:\)"
💿	CD-ROM	E:\
🖧	Network drive	F:\, G:\, …
🖫	Backup drive	Archive:\

Directories and files:

Icon	Description
📁	Directory
📂	Open directory
A	Archive in ZIP format
🕒	The contents of a directory are being read.
M	Module
✗	Module containing errors
⬇	SRC file
✗	SRC file containing errors
📝	DAT file
✗	DAT file containing errors
📄	ASCII file. Can be read using any editor.
📄	Binary file. Cannot be read in the text editor.

8.1.4 Creating a new folder

Precondition

- The Navigator is displayed.

Procedure

1. In the directory structure, use the UP and DOWN arrow keys to select the folder in which the new folder is to be created.

   Closed folders can be opened by pressing the Enter key.
2. Press the **NEW** softkey.
3. Enter a name for the folder and press **OK**.

8.1.5 Creating a new program

Precondition

- The Navigator is displayed.

Procedure

1. In the directory structure, use the UP and DOWN arrow keys to select the folder in which the program is to be created.

   Closed folders can be opened by pressing the Enter key.
2. Move to the file list by pressing the RIGHT arrow button.
3. Press the **New** softkey.

   The **Template selection** window is opened.
4. Select the desired template and press **OK**.
5. Enter a name for the program and press the softkey **OK**.

It is not possible to select a template in the user group “User”. By default, a program of type “Module” is created.

8.1.6 Renaming a file

Precondition

- The Navigator is displayed.

Procedure

1. In the directory structure, use the UP and DOWN arrow keys to select the folder in which the file is located.

   Closed folders can be opened by pressing the Enter key.
2. Move to the file list by pressing the RIGHT arrow button. Select the desired file.
3. Select the menu sequence **File** > **Rename**.
4. Overwrite the file name with a new name and press **OK**.

8.1.7 Encrypted files

Precondition

In order to display, select or execute signed or encrypted files on the robot controller:

- User group “Expert”

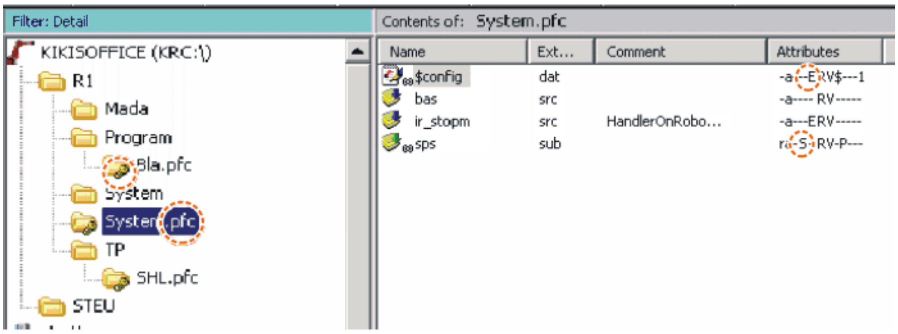
Description

SRC, DAT and SUB files can be encrypted or signed using the KUKA.Encryption software. This prevents unauthorized persons from viewing or modifying programs. Encrypted and signed files are grouped together in so-called PFC files (Protected File Container).

Further information about KUKA.Encryption can be found in the software documentation.

Display of the files in the Navigator:

When the robot controller is started, it recognizes all PFC files contained in the directory KRC:\ and loads the files contained in them.



Filter: Detail

KIKISOFFICE (KRC:\)
- R1
  - Mada
  - Program
    - Bla.pfc
  - System
  - System.pfc
  - TP
    - SHL.pfc
- STEU

Contents of: System.pfc

Name	Ext...	Comment	Attributes
$config	dat		-a--ERV$---1
bas	src		-a---- RV-----
ir_stopm	src	HandlerOnRobo...	-a---ERV-----
sps	sub		ra-S-RV-P---

Fig. 8-5: Encrypted files in the Navigator

PFC files are displayed in the directory structure with a key icon.

The files contained in the PFC file are displayed in the file list in the customary manner as SRC, DAT and SUB files. The **Attributes** column indicates whether the file is encrypted or signed.

- **E**: File is encrypted. Encrypted files cannot be read by the user, but can be edited (e.g. using the variable correction function).
- **S**: File is signed. Signed files can be read by the user, but cannot be edited.

An encrypted or signed program can be highlighted and selected using the **Select** softkey in the same way as other programs. The selected program can be executed, reset and deselected in the usual manner.

8.2 Selecting or opening a program

Overview

- A program can be selected or opened. Instead of the Navigator, an editor is then displayed with the program.

  (>>> 8.2.1 "Selecting a program" Page 227)

  (>>> 8.2.2 "Opening a program" Page 228)
- It is possible to toggle backwards and forwards between the program display and the Navigator.

  (>>> 8.2.3 "Toggling between the Navigator and the program" Page 228)
- A program cannot be both opened and selected at the same time. It is possible, however, for one program to be opened while another one is selected.

  (>>> 8.2.4 "Selecting one program and opening another program" Page 229)

Differences

Program is selected:

- The block pointer is displayed.
- The program can be started.
- The program can be edited to a certain extent.

  Selected programs are particularly suitable for editing in the user group “User”.

  KRL instructions covering several lines (e.g. LOOP … ENDLOOP) are not permissible.
- When the program is deselected, modifications are accepted without a request for confirmation. If impermissible modifications are programmed, an error message is displayed.

Program is opened:

- The program cannot be started.

- The program can be edited.

  Opened programs are particularly suitable for editing in the user group “Expert”.
- A request for confirmation is generated when the program is closed. Modifications can be accepted or rejected.

8.2.1 Selecting a program

> If a selected program is edited in the user group “Expert”, the cursor must then be removed from the edited line and positioned in any other line!
>
> Only in this way is it certain that the editing will be applied when the program is deselected again.

Precondition

- T1, T2 or AUT mode

Procedure

1. Select the program in the Navigator and press the **Select** softkey.

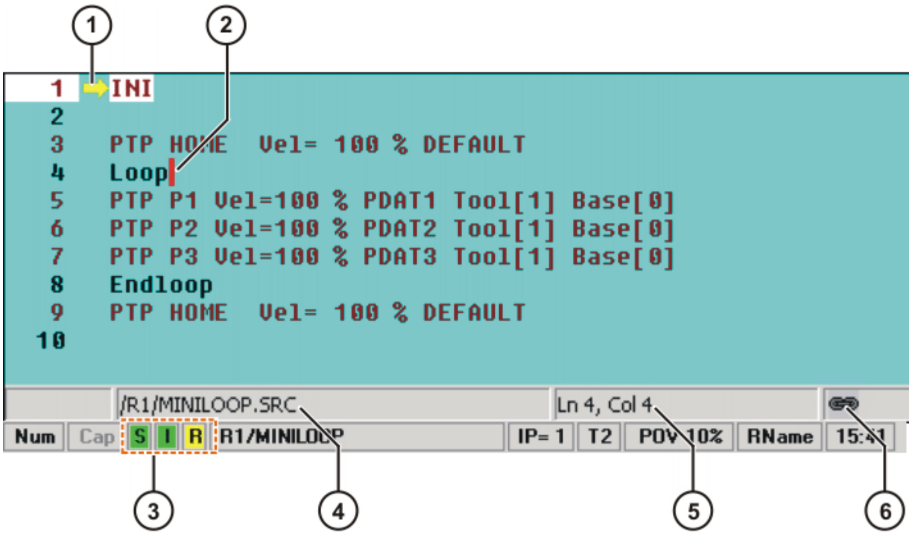
   Or: Double-click on the program in the Navigator.

   The program is displayed in the editor. It is irrelevant whether a module, an SRC file or a DAT file is selected. It is always the SRC file that is displayed in the editor.
2. Start or edit the program.
3. Deselect the program again by selecting the menu sequence **Edit** > **Cancel program**.

> When the program is deselected, modifications are accepted without a request for confirmation!

If the program is running, it must be stopped before it can be deselected.

Description



```
INI

PTP HOME  Vel= 100 % DEFAULT
Loop
PTP P1 Vel=100 % PDAT1 Tool[1] Base[0]
PTP P2 Vel=100 % PDAT2 Tool[1] Base[0]
PTP P3 Vel=100 % PDAT3 Tool[1] Base[0]
Endloop
PTP HOME  Vel= 100 % DEFAULT

```

/R1/MINILOOP.SRC | Ln 4, Col 4

Num | Cap | S | I | R | R1/MINILOOP | IP= 1 | T2 | POV 10% | RName | 15:41

Fig. 8-6: Program is selected

1 Block pointer

2 Cursor

3

(>>> 4.2.4 "Status bar" Page 46)

4 Program path and file name

5 Position of the cursor

6 The icon indicates that the program is selected.

8.2.2 Opening a program

Precondition

- T1, T2 or AUT mode

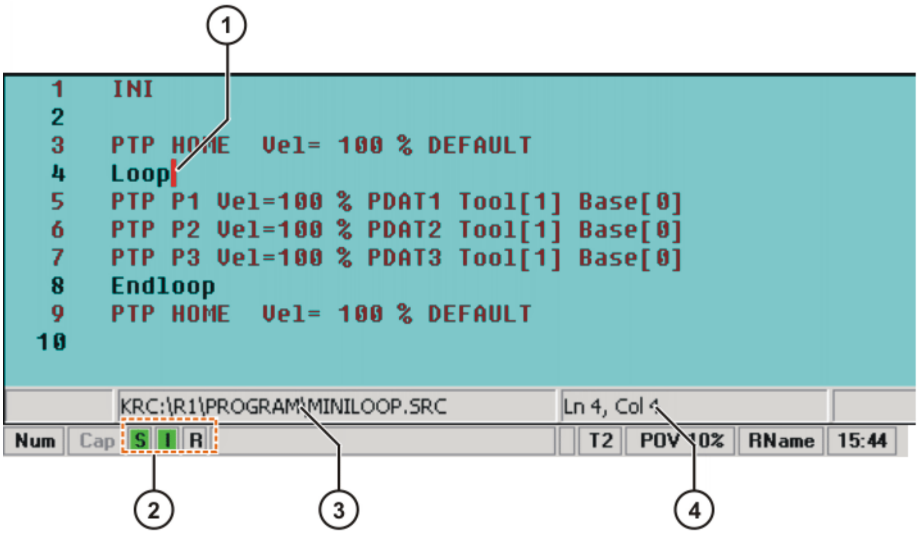
A program can be opened in AUT EXT mode, but not edited.

Procedure

1. Select the program in the Navigator and press the **Open** softkey. The program is displayed in the editor.

   If a module has been selected, the SRC file is displayed in the editor. If an SRC file or DAT file has been selected, the corresponding file is displayed in the editor.
2. Edit the program.
3. Close the program: press the **Close** softkey.
4. To accept the changes, answer the request for confirmation with **Yes**.

Description



```
INI

PTP HOME  Vel= 100 % DEFAULT
Loop
PTP P1 Vel=100 % PDAT1 Tool[1] Base[0]
PTP P2 Vel=100 % PDAT2 Tool[1] Base[0]
PTP P3 Vel=100 % PDAT3 Tool[1] Base[0]
Endloop
PTP HOME  Vel= 100 % DEFAULT
```

KRC:\R1\PROGRAM\MINILOOP.SRC

Ln 4, Col 4

Fig. 8-7: Program is open

1 Cursor

2 (>>> 4.2.4 "Status bar" Page 46)

3 Program path and file name

4 Position of the cursor

8.2.3 Toggling between the Navigator and the program

Description

If a program is selected or open, it is possible to display the Navigator again without having to deselect or close the program. The user can then return to the program.

Procedure

Program is selected:

- Toggling from the program to the Navigator: press the **NAVIGATOR** softkey.
- Toggling from the Navigator to the program: press the **PROGRAM** softkey.

A program is open:

- Toggling from the program to the Navigator: press the **NAVIGATOR** softkey.
- Toggling from the Navigator to the program: press the **EDITOR** softkey.

8.2.4 Selecting one program and opening another program

Description

A program cannot be both opened and selected at the same time. It is possible, however, for one program to be opened while another one is selected.

Procedure

1. Select a program.
2. Return to the Navigator by pressing the **NAVIGATOR** softkey.
3. Select another program then select the menu sequence **File** > **Open** > **File/Folder**.
4. You can return to the Navigator by pressing the **NAVIGATOR** softkey.
   Here you have the following options:
   - Display the selected program again by pressing the **EDITOR** softkey.
   - Display the opened program again by pressing the **PROGRAM** softkey.

8.3 Structure of a KRL program

```
1  DEF my_program( )
2  INI
3
4  PTP HOME  Vel= 100 % DEFAULT
   ...
8  LIN point_5 CONT Vel= 2 m/s CPDAT1 Tool[3] Base[4]
   ...
14 PTP point_1 CONT Vel= 100 % PDAT1 Tool[3] Base[4]
   ...
20 PTP HOME  Vel= 100 % DEFAULT
21
22 END
```

Line	Description
1	The DEF line indicates the name of the program. If the program is a function, the DEF line begins with "DEFFCT" and contains additional information. The DEF line can be displayed or hidden. Select the menu sequence **Configure** > **Tools** > **Editor** > **Def-line**. This function is not available in the user group "User".
2	The INI line contains initializations for internal variables and parameters.
4	HOME position
8	LIN motion (>>> 10.2.3 "Programming a LIN motion" Page 274)
14	PTP motion (>>> 10.2.1 "Programming a PTP motion" Page 273)
20	HOME position
22	The END line is the last line in any program. If the program is a function, the wording of the END line is "ENDFCT". The END line must not be deleted!

The first motion instruction in a KRL program must define an unambiguous starting position. The HOME position, which is stored by default in the robot controller, ensures that this is the case.

If the first motion instruction is not the default HOME position, or if this position has been changed, one of the following statements must be used:

- Complete PTP instruction of type POS or E6POS
- Complete PTP instruction of type AXIS or E6AXIS

“Complete” means that all components of the end point must be specified.

⚠ **WARNING** If the HOME position is modified, this affects all programs in which it is used. Physical injuries or damage to property may result.

In programs that are used exclusively as subprograms, different statements can be used as the first motion instruction.

8.3.1 HOME position

The HOME position is not program-specific. It is generally used as the first and last position in the program as it is uniquely defined and uncritical.

The HOME position is stored by default with the following values in the robot controller:

Axis	A1	A2	A3	A4	A5	A6
Position	0°	90°	0°	0°	0°	0°

Additional HOME positions can be taught. A HOME position must meet the following conditions:

- Good starting position for program execution
- Good standstill position. For example, the stationary robot must not be an obstacle.

⚠ **WARNING** If the HOME position is modified, this affects all programs in which it is used. Physical injuries or damage to property may result.

8.4 Displaying/hiding program sections

8.4.1 Displaying/hiding the DEF line

Description

By default, the DEF line is hidden. Declarations can only be made in a program if the DEF line is visible.

The DEF line is displayed and hidden separately for opened and selected programs. If detail view (ASCII mode) is activated, the DEF line is visible and does not need to be activated separately.

Precondition

- User group “Expert”
- Program is selected or open.

Procedure

- Select the menu sequence **Configure** > **Misc.** > **Editor** > **Def-line**.

  Check mark activated in menu: DEF line is displayed.

  Check mark not activated in menu: DEF line is hidden.

8.4.2 Activating detail view (ASCII mode)

Description

Detail view (ASCII mode) is deactivated by default to keep the program transparent. If detail view is activated, hidden program lines, such as the FOLD and ENDFOLD lines and the DEF line, are displayed.

Detail view is activated and deactivated separately for opened and selected programs.

Precondition

- User group “Expert”
- Program is selected or open.

Procedure

- Select the menu sequence **Configure** > **Misc.** > **Editor** > **ASCII Mode**.

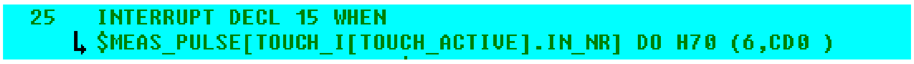
  Check mark activated in menu: ASCII mode is activated.

  Check mark not activated in menu: ASCII mode is deactivated.

8.4.3 Activating/deactivating the line break function

Description

If a line is wider than the program window, the line is broken by default. The part of the line after the break has no line number and is marked with a black, L-shaped arrow. The line break function can be deactivated.



```
25   INTERRUPT DECL 15 WHEN
   ↳ $MEAS_PULSE[TOUCH_I[TOUCH_ACTIVE].IN_NR] DO H70 (6,CD0 )
```

Fig. 8-8: Line break

The line break function is activated and deactivated separately for opened and selected programs.

Precondition

- User group “Expert”
- Program is selected or open.

Procedure

- Select the menu sequence **Configure** > **Miscellaneous** > **Editor** > **Linebreak**.

  Check mark activated in menu: line break function is activated.

  Check mark not activated in menu: line break function is deactivated.

8.4.4 Displaying Folds

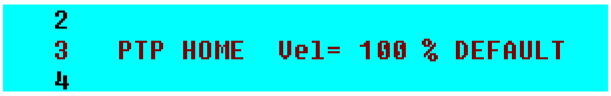
Description

Folds are used to hide sections of the program. In this way, Folds make programs more transparent. The hidden program sections are processed during program execution in exactly the same way as normal program sections.

- In the user group “User”, Folds are always closed. In other words, the contents of the Folds are not visible and cannot be edited.
- In the user group “Expert”, Folds are closed by default. They can be opened and edited. New Folds can be created.

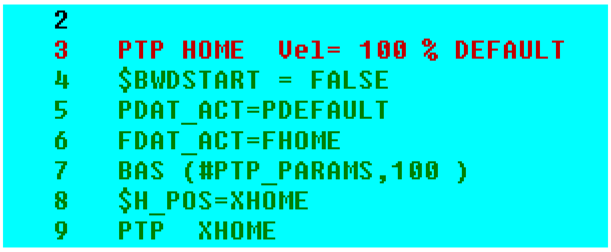
  (>>> 8.6.3 "Creating Folds" Page 239)

If a program is deselected, all Folds are automatically closed.



```
2
3   PTP HOME  Vel= 100 % DEFAULT
4
```

Fig. 8-9: Example of a closed Fold



```

PTP HOME  Vel= 100 % DEFAULT
$BWDSTART = FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS (#PTP_PARAMS,100 )
$H_POS=XHOME
PTP  XHOME
```

Fig. 8-10: Example of an open Fold

Color coding of Folds:

Color	Description
Dark red	Closed fold
Light red	Opened fold
Dark blue	Closed sub-Fold
Light blue	Opened sub-Fold
Green	Contents of the Fold

Precondition

- User group “Expert”
- Program is selected or open.

Procedure

1. Position the cursor in the line containing the Fold.
2. Select the menu sequence **Program** > **FOLD** > **Current FOLD open/ close**. The Fold then opens.
3. To close the Fold, select the same menu sequence as for opening it.

Alternatively, select the menu sequence **Program** > **FOLD** > **All FOLDs open** or **All FOLDs close** to open or close all the Folds in a program at once.

8.5 Starting a program

8.5.1 Program run modes

The program run mode is selected in the left-hand status key bar.

Status key	Program run mode	Description
🏃	#GO	The program is executed through to the end without stopping.
🚶🚶	#MSTEP (Motion Step)	The program is executed with a stop after each motion block. The Start key must be pressed again for each motion block.
🚶🚶🚶	#ISTEP (Incremental Step)	The program is executed with a stop after each program line. Program lines that cannot be seen and blank lines are also taken into consideration. The Start key must be pressed again for each line. ISTEP is only available to the user group “Expert”.
🚶🚶	#BSTEP (backward motion)	This program run mode is automatically selected if the Start backwards key is pressed.

In #MSTEP and #ISTEP modes, the program is executed without an advance run.

The following additional program run modes are available for systems integrators.

These program run modes can only be selected via the variable correction function. System variable for the program run mode: $PRO_MODE.

Status key	Program run mode	Description
	#PSTEP (Program Step)	The program is executed step by step without an advance run. Subprograms are executed completely.
	#CSTEP (Continuous Step)	Approximate positioning points are executed with advance processing, i.e. they are approximated. Exact positioning points are executed without an advance run and with a stop after the motion instruction.

8.5.2 Advance run

The advance run is the maximum number of motion blocks that the robot controller calculates and plans in advance during program execution. The actual number is dependent on the capacity of the computer.

The advance run refers to the current position of the block pointer. It is set via the system variable $ADVANCE:

- Default value: 3
- Maximum value: 5

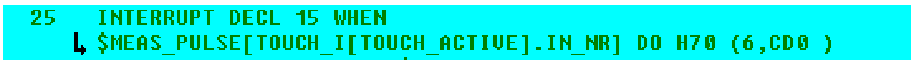
The advance run is required, for example, in order to be able to calculate approximate positioning motions. If $ADVANCE = 0 is set, approximate positioning is not possible.

Certain statements trigger an advance run stop. These include statements that influence the periphery, e.g. OUT statements.

8.5.3 Icons in the program

Line break

If a line is wider than the program window, the line is broken by default. The part of the line after the break has no line number and is marked with a black, L-shaped arrow. The line break function can be deactivated.
(>>> 8.4.3 "Activating/deactivating the line break function" Page 231)



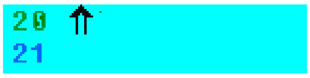
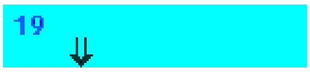
```
 25   INTERRUPT DECL 15 WHEN
    ↳ $MEAS_PULSE[TOUCH_I[TOUCH_ACTIVE].IN_NR] DO H70 (6,CD0 )
```

Fig. 8-11: Line break

Block pointer

During program execution, the block pointer indicates which motion block is currently being executed.

Icon	Description
↳	L-shaped arrow (yellow): The motion block is being executed in the forwards direction.
↳+	L-shaped arrow (yellow) with plus sign: The motion block is being executed in the forwards direction. This block pointer is not displayed in the user group "User".
➡	Normal arrow (yellow): The robot has completed the motion block in the forwards direction
➡+	Normal arrow (yellow) with plus sign: The robot has completed the motion block in the forwards direction This block pointer is not displayed in the user group "User".
↰	L-shaped arrow (red): The motion block is being executed in the backwards direction.
↰+	L-shaped arrow (red) with plus sign: The motion block is being executed in the backwards direction. This block pointer is not displayed in the user group "User".
➡	Normal arrow (red): The robot has completed the motion block in the backwards direction
➡+	Normal arrow (red) with plus sign: The robot has completed the motion block in the backwards direction This block pointer is not displayed in the user group "User".

Icon	Description
 20 ⇑ 21	The block pointer is located higher up in the program.
 19 ⇓	The block pointer is located lower down in the program.

8.5.4 Setting the program override (POV)

Description

Program override is the velocity of the robot during program execution. The program override is specified as a percentage of the programmed velocity.

> ℹ In T1 mode, the maximum velocity is 250 mm/s, irrespective of the value that is set.

Preparation

- Define the program override intervals:

  Select the menu sequence **Configure** > **Jogging** > **Program OV Steps**.

Active	Description
No	The override can be adjusted in 1% steps.
Yes	Intervals: 100%, 75%, 50%, 30%, 10%, 3%, 1%, 0%

Procedure

- Increase or reduce the override in the right-hand status key bar. The status key indicates the current override as a percentage.

8.5.5 Starting a program forwards (manual)

Precondition

- A program is selected.
- Operating mode T1 or T2
- Robot is not in gravitation compensation.

Procedure

1. Select the program run mode.
2. Hold the enabling switch down and wait until the status bar indicates I (i.e. drives ready).
3. Carry out a BCO run:

   Press Start key and hold it down until the message “Programmed path reached (BCO)” is displayed in the message window. The robot stops.

> ⚠ **WARNING** A BCO run is always executed as a PTP motion from the actual position to the target position. Observe the motion to avoid collisions. The velocity is automatically reduced during the BCO run.

4. Press Start key and hold it down.

   The program is executed with or without stops, depending on the program run mode.

> To stop a program that has been started manually, release the Start key.

8.5.6 Starting a program forwards (automatic)

Precondition

- A program is selected.
- Operating mode Automatic (not Automatic External)
- Robot is not in gravitation compensation.

Procedure

1. Select the program run mode GO in the left-hand status key bar:

2. Press **Drives ON**.
3. Carry out a BCO run:

   Press Start key and hold it down until the message “Programmed path reached (BCO)” is displayed in the message window. The robot stops.

> ⚠ **WARNING** A BCO run is always executed as a PTP motion from the actual position to the target position. Observe the motion to avoid collisions. The velocity is automatically reduced during the BCO run.

4. Press the Start key. The program is executed.

> To stop a program that has been started in Automatic mode, press the STOP key.

8.5.7 Carrying out a block selection

Description

A program can be started at any point by means of a block selection.

Precondition

- A program is selected.
- Operating mode T1 or T2
- Robot is not in gravitation compensation.

Procedure

1. Select the program run mode.
2. Position the cursor in the line containing the motion block at which the program is to be started.
3. Press the **Line Sel.** softkey. The yellow block pointer indicates the motion block.
4. Hold the enabling switch down and wait until the status bar indicates I (i.e. drives ready).
5. Carry out a BCO run: Press Start key and hold it down until the message “Programmed path reached (BCO)” is displayed in the message window. The robot stops.

> ⚠ WARNING A BCO run is always executed as a PTP motion from the actual position to the target position. Observe the motion to avoid collisions. The velocity is automatically reduced during the BCO run.

6. The program can now be started manually or automatically. It is not necessary to carry out a BCO run again.

8.5.8 Starting a program backwards

Description

In the case of backward motion, the robot stops at every point. Approximate positioning is not possible.

> Exactly how the controller responds during backward motion depends on the configuration.
> (>>> 6.19 "Backward motion" Page 160)

Precondition

- A program is selected.
- Operating mode T1 or T2
- Robot is not in gravitation compensation.

Procedure

1. Hold the enabling switch down and wait until the status bar indicates I (i.e. drives ready).
2. Carry out a BCO run:

   Press Start key and hold it down until the message “Programmed path reached (BCO)” is displayed in the message window. The robot stops.

> ⚠ WARNING A BCO run is always executed as a PTP motion from the actual position to the target position. Observe the motion to avoid collisions. The velocity is automatically reduced during the BCO run.

3. Press Start backwards key. The program run mode “Backward motion” is automatically selected:

4. Press Start backwards key again for each motion block.

8.5.9 Resetting a program

Description

In order to restart an interrupted program from the beginning, it must be reset. This returns the program to the initial state.

Precondition

- Program is selected.

Procedure

- Select the menu sequence **Program** > **Reset program**.

8.5.10 Starting Automatic External mode

> ⚠ **WARNING** There is no BCO run in Automatic External mode. This means that the robot moves to the first programmed position after the start at the programmed (not reduced) velocity and does not stop there.

Precondition

- Operating mode T1 or T2
- Inputs/outputs for Automatic External and the program CELL.SRC are configured.
- Robot is not in gravitation compensation.

Procedure

1. Select the program CELL.SRC in the Navigator. (This program is located in the folder “R1”.)
2. Set program override to 100%. (This is the recommended setting. A different value can be set if required.)
3. Carry out a BCO run:

   Hold down the enabling switch. Then press the Start key and hold it down until the message “Programmed path reached (BCO)” is displayed in the message window.

> ⚠ **WARNING** A BCO run is always executed as a PTP motion from the actual position to the target position. Observe the motion to avoid collisions. The velocity is automatically reduced during the BCO run.

4. Turn the mode selector switch to “Automatic External”.
5. Start the program from a higher-level controller (PLC).

> ℹ To stop a program that has been started in Automatic mode, press the STOP key.

8.6 Editing a program

> ℹ If a selected program is edited in the user group “Expert”, the cursor must then be removed from the edited line and positioned in any other line!
> Only in this way is it certain that the editing will be applied when the program is deselected again.

8.6.1 Inserting a comment or stamp

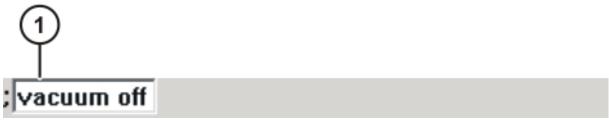
Precondition

- Program is selected or open.
- T1, T2 or AUT mode

Procedure

1. Position the cursor in the line after which the comment or stamp is to be inserted.
2. Select the menu sequence **Commands** > **Comment** > **Normal** or **Stamp**.
3. In the case of Stamp: update the system time by pressing the **New time** softkey.
4. Enter text.
5. Save by pressing the **Cmd Ok** softkey.

Description Comment



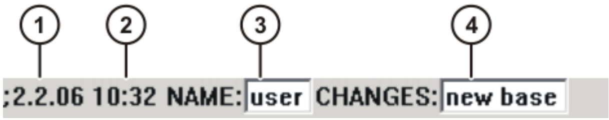
;vacuum off

Fig. 8-12: Inline form "Comment"

Item	Description
1	Any text

Description Stamp

A stamp is a comment that is extended to include the system date and time and the user ID.



;2.2.06 10:32 NAME:user CHANGES:new base

Fig. 8-13: Inline form "Stamp"

Item	Description
1	Current system date (cannot be edited)
2	Current system time
3	Name or ID of the user
4	Any text

8.6.2 Deleting program lines

Precondition

- Program is selected or open.
- T1, T2 or AUT mode

Procedure

1. Position the cursor in the line to be deleted.

   If several consecutive lines are to be deleted: Position the cursor in the first line. Then press SHIFT + DOWN ARROW until all the lines are selected.
2. Select the menu sequence **Edit** > **Delete**.
3. Confirm the request for confirmation with **Yes**.

Lines cannot be restored once they have been deleted!

If a program line containing a motion instruction is deleted, the point name and coordinates remain saved in the DAT file. The point can be used in other motion instructions and does not need to be taught again.

8.6.3 Creating Folds

Syntax

`;FOLD` *Name*

Statements

`;ENDFOLD` *<Name>*

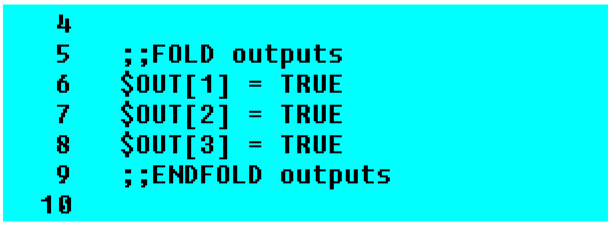
The ENDFOLD lines can be assigned more easily if the name of the Fold is entered here as well. Folds can be nested.

Precondition

- User group “Expert”
- Program is selected or open.

Procedure

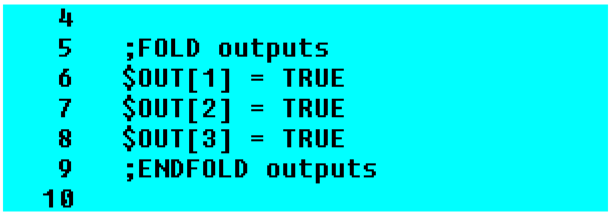
1. Enter Fold in program. A double semicolon prevents the Fold from closing when edited.



```
;;FOLD outputs
$OUT[1] = TRUE
$OUT[2] = TRUE
$OUT[3] = TRUE
;;ENDFOLD outputs
```

Fig. 8-14: Creating a sample Fold, step 1

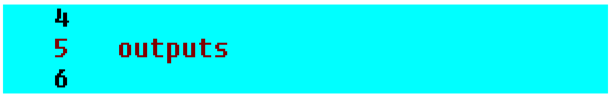
2. Delete the second semicolon.



```
;FOLD outputs
$OUT[1] = TRUE
$OUT[2] = TRUE
$OUT[3] = TRUE
;ENDFOLD outputs
```

Fig. 8-15: Creating a sample Fold, step 2

3. Position the cursor in a line outside the Fold. The Fold closes.



```
outputs
```

Fig. 8-16: Creating a sample Fold, step 3

8.6.4 Additional editing functions

The following additional program editing functions can be found in the **Program** menu:

Function	Precondition
Copy	■ Program is selected or open. ■ User group “Expert” ■ T1, T2 or AUT mode
Paste	■ Program is selected or open. ■ User group “Expert” ■ T1, T2 or AUT mode
Cut	■ Program is selected or open. ■ User group “Expert” ■ T1, T2 or AUT mode

Function	Precondition
Find	■ Program is selected or open.
Replace	■ Program has been opened. ■ User group “Expert”

8.7 Printing a program

Procedure

1. Select the program in the Navigator. Multiple program selection is also possible.
2. Select the menu sequence **File** > **Print**>**Current selection**.

8.8 Archiving

8.8.1 Destination for archiving

Overview

Archiving can be performed to the following target destinations:

- KUKA.USBData stick (default configuration)

NOTICE Only the KUKA.USB data stick may be used. Data may be lost or modified if any other USB stick is used.

- Network path or local path

  The desired path must be configured in the KRC Configurator.

8.8.2 Archiving data

Precondition

- If data are to be archived to a USB stick: KUKA.USBData stick is connected.
- If data are to be archived to a network path or a local path: the path is configured in the KRC Configurator.

Procedure

1. Select the menu sequence **File** > **Archive** and the desired menu item.
2. Confirm the request for confirmation with **Yes**. The archive is created.

   A message indicates completion of the archiving process. The file ARCHIVE.ZIP is generated by default.
3. If data were archived to a USB stick: remove the stick when the LED on the stick is no longer lit.

Description

The following menu items are available for archiving. Exactly which files are archived depends on the configuration in the KRC Configurator.
(>>> 6.25.8 "Archive Manager tab" Page 206)

Menu item	Description
All	The data that are required to restore an existing system are archived.
Applications	All user-defined KRL modules and their corresponding system files are archived.
Machine data	The machine data are archived.
Configuration > Drivers	The I/O drivers are archived. Not available in the user group “User”.
Configuration > I/O Longtexts	The long text names of the inputs/outputs are archived. Not available in the user group “User”.

Menu item	Description
Configuration > KUKA Tech-Pack	The configuration of the installed technology packages is archived. Not available in the user group “User”.
Log Data	The log files are archived.
Current selection	The files selected in the Navigator are archived.

- If archiving is carried out using the menu item **All**, an existing archive will be overwritten.
- If archiving is carried out using a menu item other than **All** and an archive is already available, the robot controller compares its robot name with that in the archive. If the names are different, a request for confirmation is generated.

8.8.3 Restoring data

⚠ WARNING Only KSS 5.6 lr archives may be loaded into KSS 5.6 lr. If other archives are loaded, the following may occur:

- Error messages
- Robot controller is not operable.
- Personal injury and damage to property.

Overview If the archive files are not the same version as the system files, an error message is generated. Similarly, if the version of the archived technology packages does not match the installed version, an error message is generated.

8.8.3.1 Restoring data via the menu

Description With the exception of **Log Data**, the menu items available for restoring data are the same as those available for archiving.

When restoring data, the robot controller accesses the path configured for archiving. For example, if the default configuration (USB path) has been left, data will also be restored from the USB stick.

Precondition

- If data are to be restored from the USB stick: A KUKA.USB data stick with the archive is connected.

Procedure

1. Select the menu sequence **File** > **Restore** and the desired menu item.
2. Confirm the request for confirmation with **Yes**. Archived files are restored to the robot controller. A message indicates completion of the restoration process.
3. When data have been restored from the USB stick: remove the stick when the LED on the stick is no longer lit.
4. Reboot the robot controller.

8.8.3.2 Restoring data via softkey

Description When restoring data using the softkey, a different path can be selected from that to which the data were archived.

Precondition

- If data are to be restored from the USB stick: A KUKA.USB data stick with the archive is connected.
- If data are to be restored from a network path or a local path: the path is configured in the KRC Configurator.

Procedure

1. Select the drive **(ARCHIVE:\)** in the Navigator. The directories belonging to **(ARCHIVE:\)** are displayed.
2. Select the desired object in a directory.
3. Press the **Restore** softkey.

> The **Restr. All** softkey ignores which objects are selected in the Navigator and accesses the path that is configured for archiving.

4. Confirm the request for confirmation with **Yes**. Archived files are restored to the robot controller. A message indicates completion of the restoration process.
5. When data have been restored from the USB stick: remove the stick when the LED on the stick is no longer lit.
6. Reboot the robot controller.

9 Basic principles of motion programming

9.1 Overview of motion types

The following motion types can be programmed:

- Point-to-point motions (PTP)
  (>>> 9.2 "Motion type PTP" Page 243)
- Linear motions (LIN)
  (>>> 9.3 "Motion type LIN" Page 243)
- Circular motions (CIRC)
  (>>> 9.4 "Motion type CIRC" Page 244)
- Spline motions
  (>>> 9.7 "Motion type “Spline”" Page 251)

LIN, CIRC and spline motions are also known as CP (“Continuous Path”) motions.

The start point of a motion is always the end point of the previous motion.

9.2 Motion type PTP

The robot guides the TCP along the fastest path to the end point. The fastest path is generally not the shortest path and is thus not a straight line. As the motions of the robot axes are rotational, curved paths can be executed faster than straight paths.

The exact path of the motion cannot be predicted.

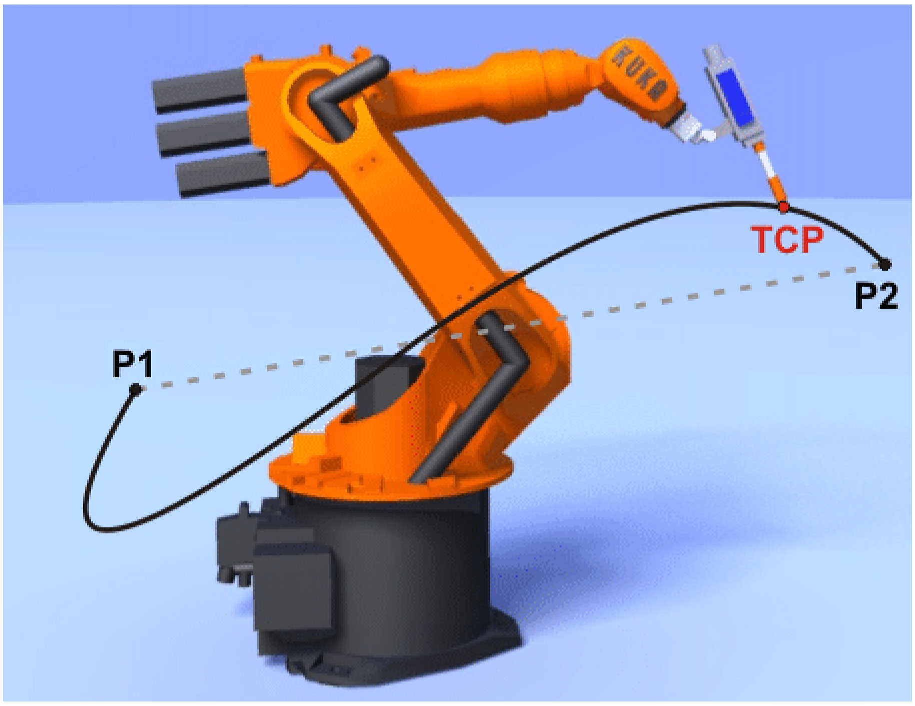


Fig. 9-1: PTP motion

9.3 Motion type LIN

The robot guides the TCP at a defined velocity along a straight path to the end point.

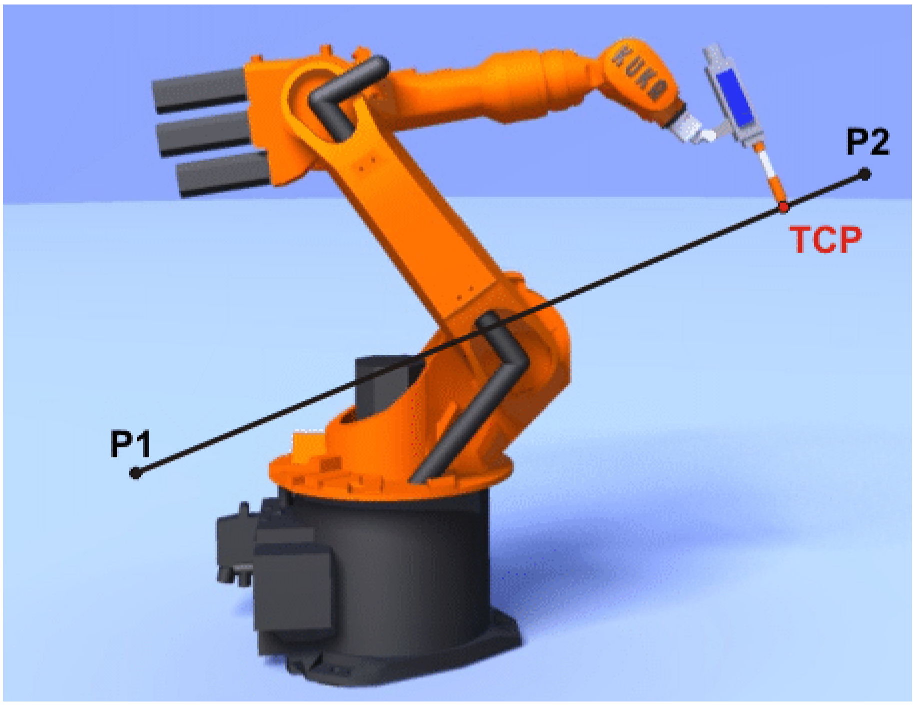


Fig. 9-2: LIN motion

9.4 Motion type CIRC

The robot guides the TCP at a defined velocity along a circular path to the end point. The circular path is defined by a start point, auxiliary point and end point.

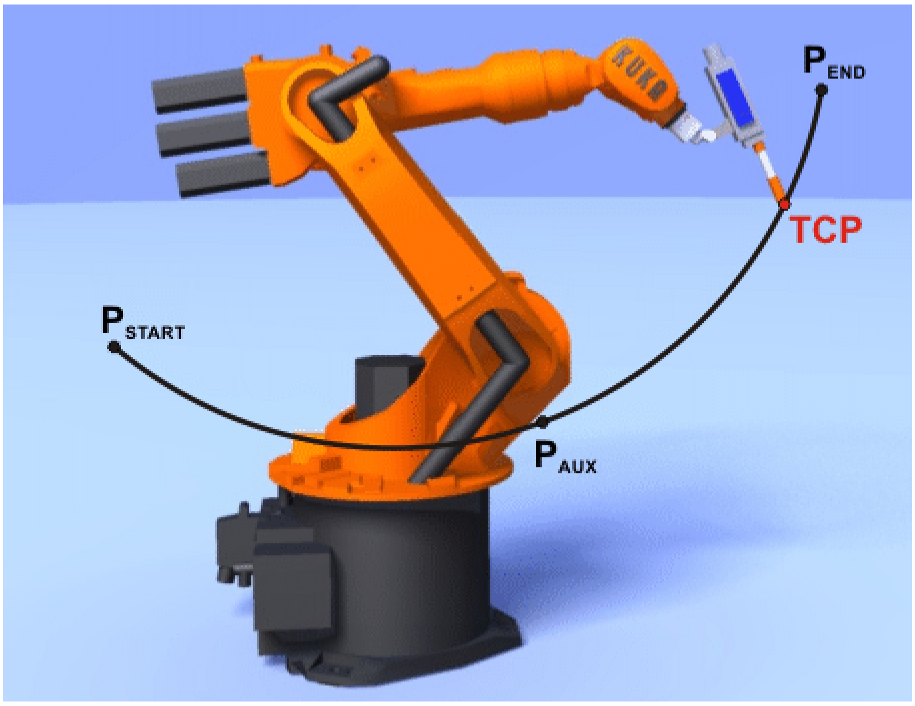


Fig. 9-3: CIRC motion

9.5 Approximate positioning

Approximate positioning means that the motion does not stop exactly at the programmed point. Approximate positioning is an option that can be selected during motion programming.

> Approximate positioning is not possible if the motion instruction is followed by an instruction that triggers an advance run stop.

PTP motion

The TCP leaves the path that would lead directly to the end point and moves along a faster path. During programming of the motion, the maximum distance from the end point at which the TCP may deviate from its original path is defined.

The path of an approximated PTP motion cannot be predicted. It is also not possible to predict which side of the approximated point the path will run.

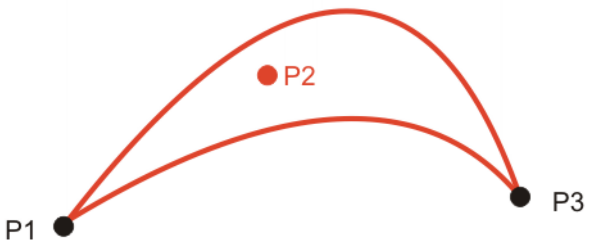


Fig. 9-4: PTP motion, P2 is approximated

LIN motion

The TCP leaves the path that would lead directly to the end point and moves along a shorter path. During programming of the motion, the maximum distance from the end point at which the TCP may deviate from its original path is defined.

The path in the approximate positioning range is **not** an arc.

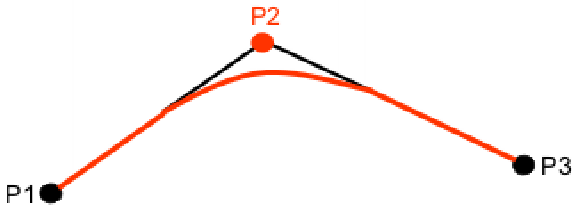


Fig. 9-5: LIN motion, P2 is approximated

CIRC motion

The TCP leaves the path that would lead directly to the end point and moves along a shorter path. During programming of the motion, the maximum distance from the end point at which the TCP may deviate from its original path is defined.

The motion always stops exactly at the auxiliary point.

The path in the approximate positioning range is **not** an arc.

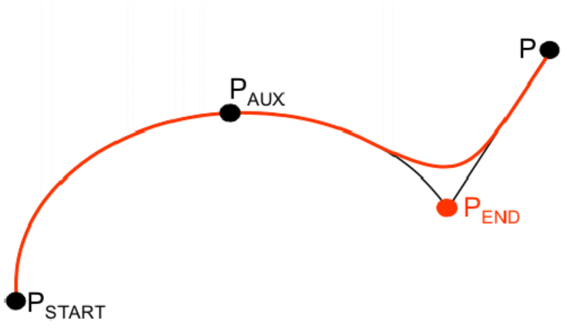


Fig. 9-6: CIRC motion, P_{END} is approximated

9.6 Orientation control LIN, CIRC

Description

The orientation of the TCP can be different at the start point and end point of a motion. There are several different types of transition from the start orientation to the end orientation. A type must be selected when a CP motion is programmed.

The orientation control for LIN and CIRC motions is defined as follows:

- In the option window **Motion parameter**

  (>>> 10.2.9 "Option window “Motion parameter” (LIN, CIRC)" Page 279)
- Or via the system variable $ORI_TYPE

LIN motion

Orientation control	Description
■ Option window: **Constant orientation** ■ $ORI_TYPE = #CONSTANT	The orientation of the TCP remains constant during the motion. The programmed orientation is disregarded for the end point and that of the start point is retained.
■ Option window: **Standard** ■ $ORI_TYPE = #VAR	The orientation of the TCP changes continuously during the motion.
■ Option window: **Wrist PTP** ■ $ORI_TYPE = #JOINT	The orientation control **Wrist PTP** (#JOINT) must not be used. Selecting **Wrist PTP** causes an error during program execution.

If a wrist axis singularity occurs with **Standard**, the following remedy is recommended:
Re-teach start and/or end point. Select orientations that prevent a wrist axis singularity from occurring and allow the path to be executed with **Standard**.

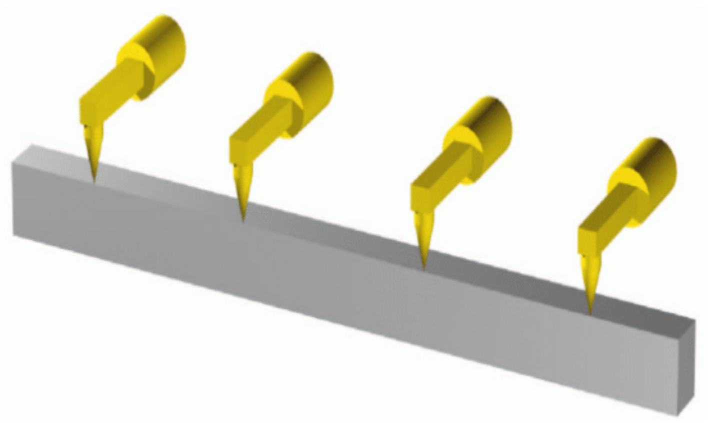


Fig. 9-7: Orientation control - Constant

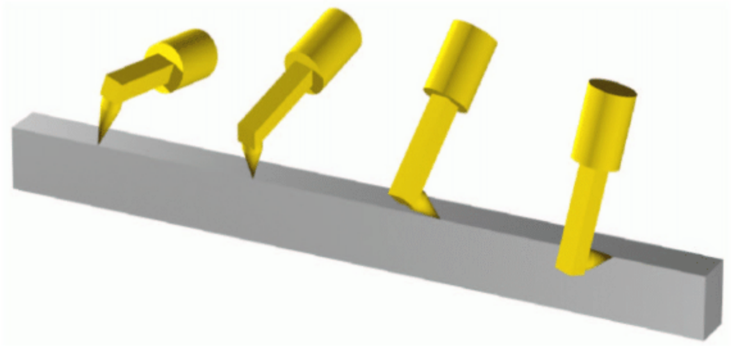


Fig. 9-8: Standard

CIRC motion

During CIRC motions, the robot controller only takes the programmed orientation of the end point into consideration. The programmed orientation of the auxiliary point is disregarded.

The same orientation control options are available for selection for CIRC motions as for LIN motions.

It is also possible to define for CIRC motions whether the orientation control is to be base-related or path-related. This is defined via the system variable $CIRC_TYPE.

Orientation control	Description
$CIRC_TYPE = #BASE	Base-related orientation control during the circular motion
$CIRC_TYPE = #PATH	Path-related orientation control during the circular motion

Combinations of $ORI_TYPE and $CIRC_TYPE:

- Constant orientation, path-related:

  $ORI_TYPE = #CONSTANT, $CIRC_TYPE = #PATH

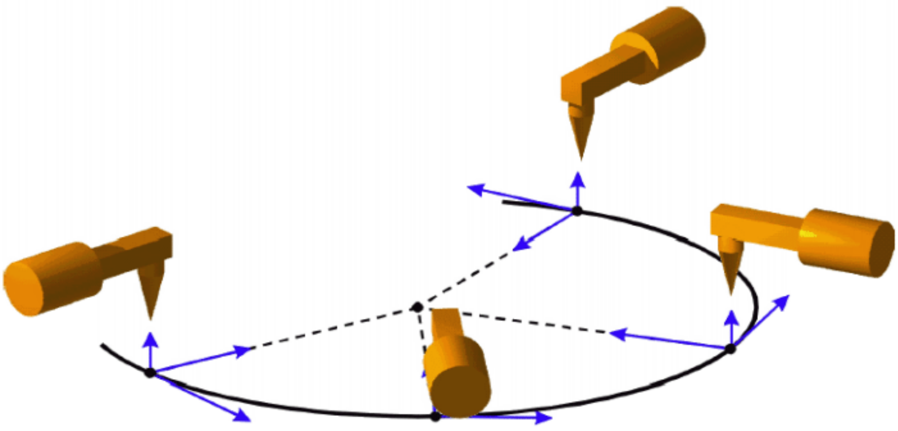


Fig. 9-9: Constant orientation, path-related

- Variable orientation, path-related:
  $ORI_TYPE = #VAR, $CIRC_TYPE = #PATH

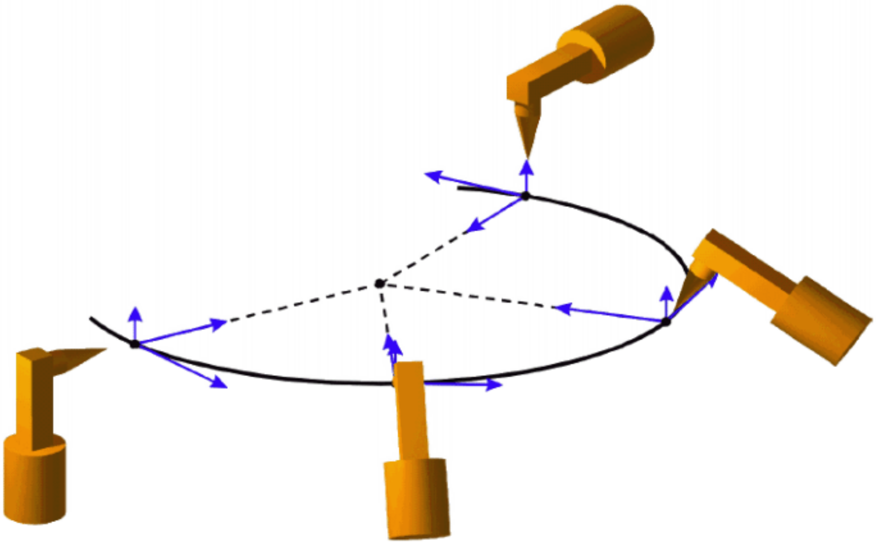


Fig. 9-10: Variable orientation, path-related

- Constant orientation, base-related:
  $ORI_TYPE = #CONSTANT, $CIRC_TYPE = #BASE

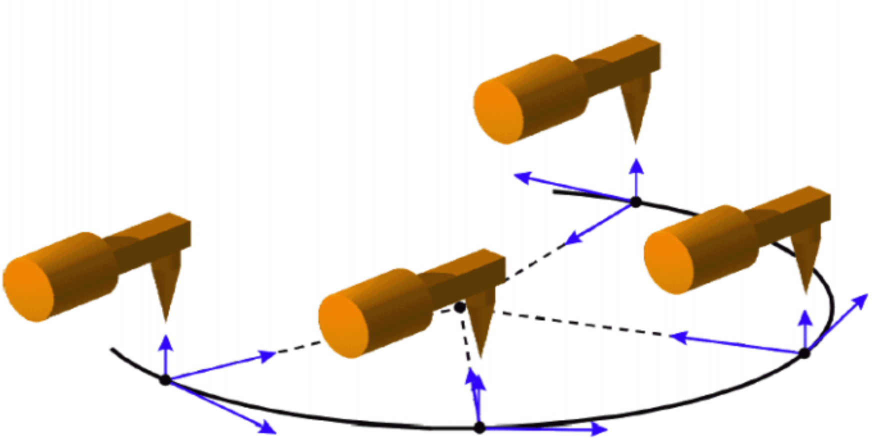


Fig. 9-11: Constant orientation, base-related

- Variable orientation, base-related:
  $ORI_TYPE = #VAR, $CIRC_TYPE = #BASE

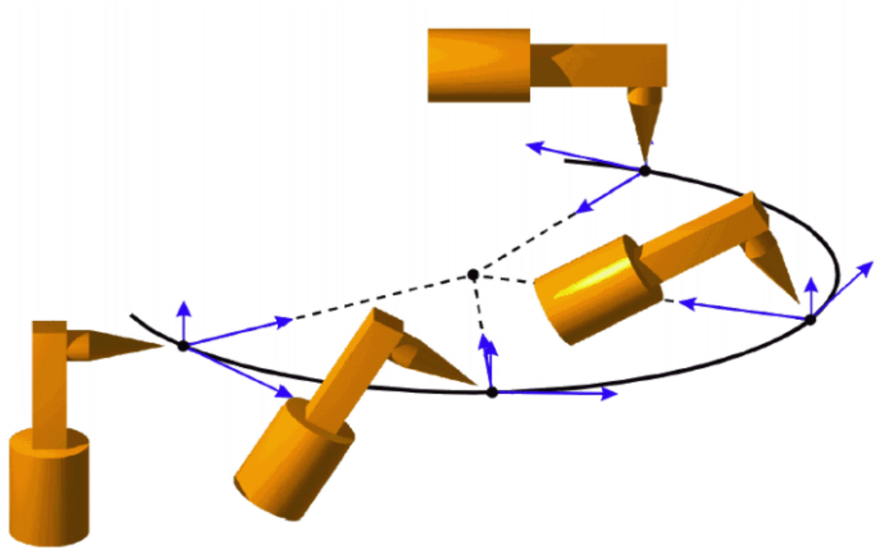


Fig. 9-12: Variable orientation, base-related

9.6.1 Combinations of $ORI_TYPE and $CIRC_TYPE

$ORI_TYPE = #CONSTANT, $CIRC_TYPE = #PATH:

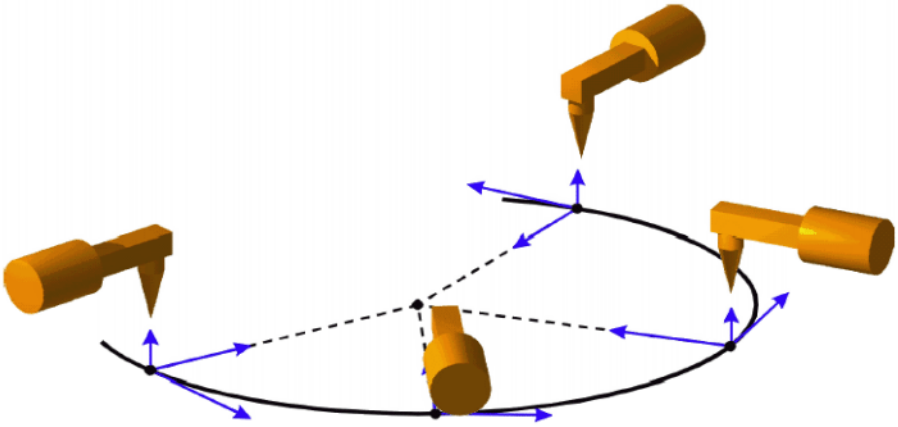


Fig. 9-13: Constant orientation, path-related

$ORI_TYPE = #VAR, $CIRC_TYPE = #PATH:

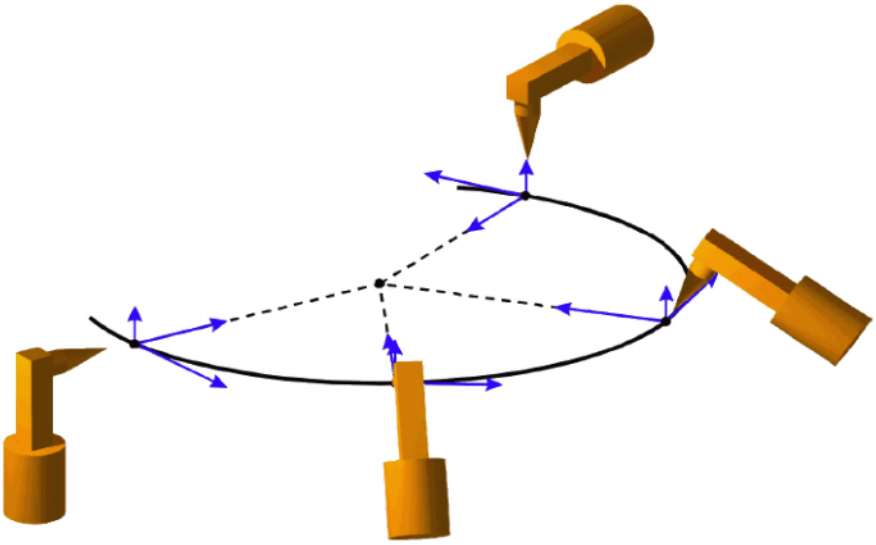


Fig. 9-14: Variable orientation, path-related

$ORI_TYPE = #CONSTANT, $CIRC_TYPE = #BASE:

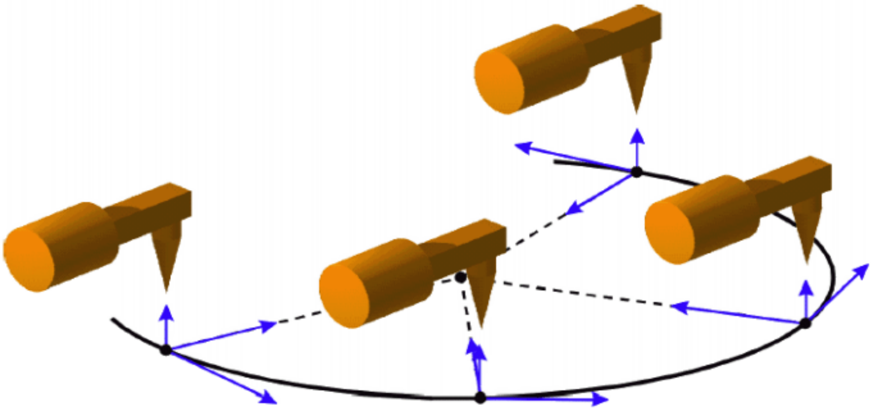


Fig. 9-15: Constant orientation, base-related

$ORI_TYPE = #VAR, $CIRC_TYPE = #BASE:

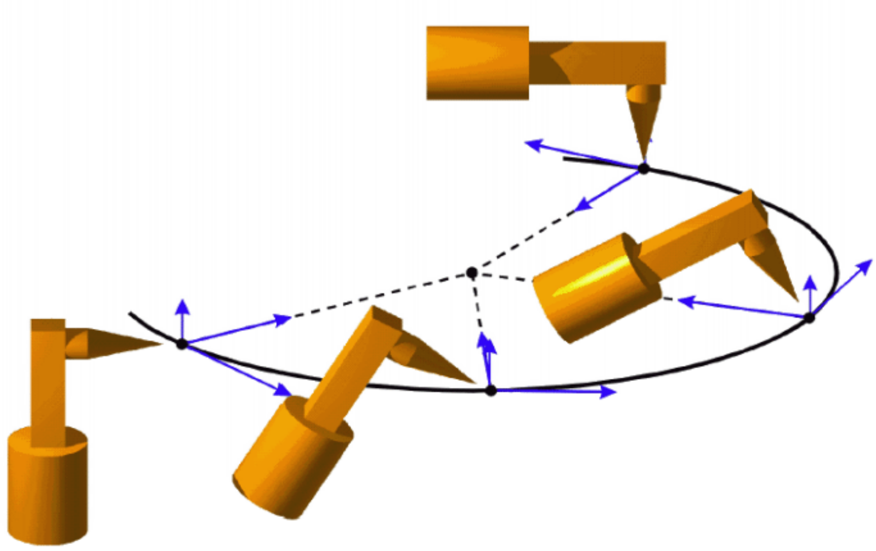


Fig. 9-16: Variable orientation, base-related

9.7 Motion type “Spline”

“Spline” is a Cartesian motion type that is suitable for particularly complex, curved paths. Such paths can generally also be generated using approximated LIN and CIRC motions, but Spline nonetheless has advantages.

Disadvantages of approximated LIN and CIRC motions:

- The path is defined by means of approximated points that are not located on the path. The approximate positioning ranges are difficult to predict. Generating the desired path is complicated and time-consuming.
- In many cases, the velocity may be reduced in a manner that is difficult to predict, e.g. in the approximate positioning ranges and near points that are situated close together.
- The path changes if approximate positioning is not possible, e.g. for time reasons.
- The path changes in accordance with the override setting, velocity or acceleration.

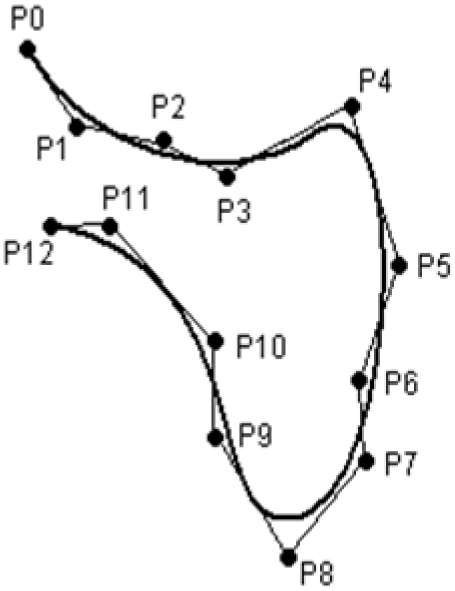


Fig. 9-17: Curved path with LIN

Advantages of Spline:

- The path is defined by means of points that are located on the path. The desired path can be generated easily.
- The programmed velocity is maintained. There are few cases in which the velocity is reduced.

  (>>> 9.7.1 "Velocity profile for spline motions" Page 253)
- The path always remains the same, irrespective of the override setting, velocity or acceleration.
- Circles and tight radii are executed with great precision.

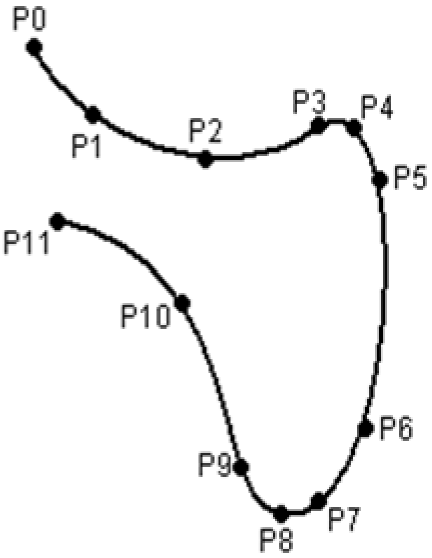


Fig. 9-18: Curved path with spline block

A spline motion can consist of several individual motions: spline segments. These are taught separately. The segments are grouped together to form the overall motion in a so-called spline block. A spline block is planned and executed by the robot controller as a single motion block.

Furthermore, individual SLIN and SCIRC motions are possible (without spline block).

Further characteristics of all spline motions:

- If all points are situated on a plane, then the path is also situated in this plane.
- If all points are situated on a straight line, then the path is also a straight line.

9.7.1 Velocity profile for spline motions

The path always remains the same, irrespective of the override setting, velocity or acceleration. Only dynamic effects can cause deviations at different velocities.

The programmed acceleration is valid not only for the direction along the path, but also perpendicular to the path. The same applies to the jerk limitation. Effects include the following:

- In the case of circles, the centrifugal acceleration is taken into consideration. The velocity that can be achieved thus also depends on the programmed acceleration and the radius of the circle.
- In the case of curves, the maximum permissible velocity is derived from the radius of the curve, the acceleration and the jerk limitation.

Reduction of the velocity

In the case of spline motions, the velocity may, under certain circumstances, fall below the programmed velocity. This occurs particularly in the case of:

- Tight corners
- Major reorientation
- Large motions of the external axes

If the points are close together, the velocity is not reduced.

Reduction of the velocity to 0

This is the case for:

- Successive points with the same Cartesian coordinates.
- Successive SLIN and/or SCIRC segments. Cause: inconstant velocity direction.

  In the case of SLIN-SCIRC transitions, the velocity is also reduced to 0 if the straight line is a tangent of the circle, as the circle, unlike the straight line, is curved.

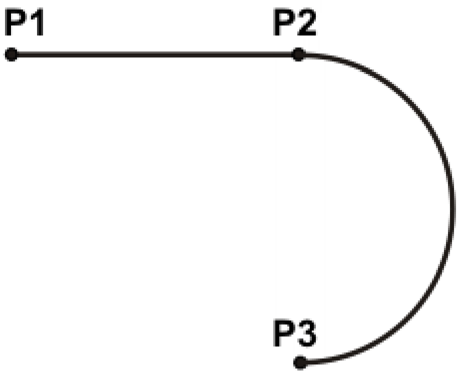


Fig. 9-19: Exact positioning at P2

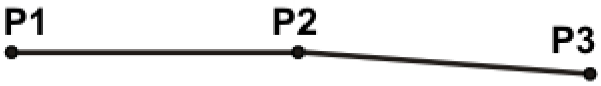


Fig. 9-20: Exact positioning at P2

Exceptions:

- In the case of successive SLIN segments that result in a straight line and in which the orientations change uniformly, the velocity is not reduced.

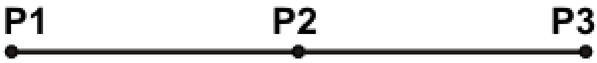


Fig. 9-21: P2 is executed without exact positioning.

- In the case of a SCIRC-SCIRC transition, the velocity is not reduced if both circles have the same center point and the same radius and if the orientations change uniformly. (This is difficult to teach, so calculate and program points.)

> Circles with the same center point and the same radius are sometimes programmed to obtain circles ≥ 360°. A simpler method is to program a circular angle.

9.7.2 Block selection with spline motions

Spline block

A spline block is planned and executed by the robot controller as a single motion block. Block selection to the spline segments is nonetheless possible. The BCO run is executed as a LIN motion. This is indicated by means of a message that must be acknowledged.

If the second segment in the spline block is an SPL segment, a modified path is executed in the following cases:

- Block selection to the first segment in the spline block
- Block selection to the spline block
- Block selection to a line before the spline block if this does not contain a motion instruction and if there is no motion instruction before the spline block

If the Start key is pressed after the BCO run, the modified path is indicated by means of a message that must be acknowledged.

Example:

```
1  PTP P0
2  SPLINE
3   SPL P1
4   SPL P2
5   SPL P3
6   SPL P4
7   SCIRC P5, P6
8   SPL P7
9   SLIN P8
10 ENDSPLINE
```

Line	Description
2	Start of the spline block

Line	Description
3 … 9	Spline segments
10	End of the spline block

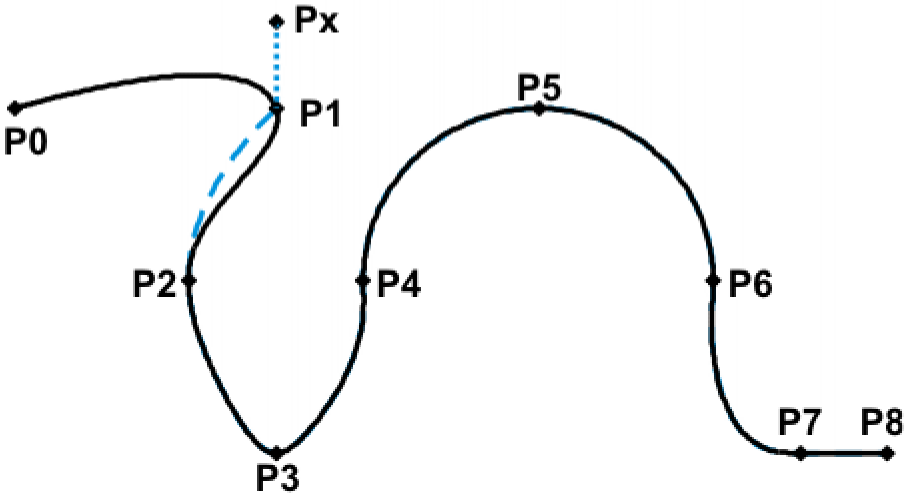


Fig. 9-22: Example: modified path in the case of block selection to P1

SCIRC

In the case of block selection to a SCIRC instruction for which a circular angle has been programmed, the motion is executed to the end point including the circular angle, provided that the robot controller knows the start point. If this is not possible, the motion is executed to the programmed end point. In this case, a message is generated, indicating that the circular angle is not being taken into consideration.

Position/type of SCIRC instruction	End point for block selection
SCIRC segment is 1st segment in the spline block	Circular angle is not taken into consideration
Other SCIRC segments in the spline block	Circular angle is taken into consideration
Individual SCIRC motions	Circular angle is not taken into consideration

9.7.3 Modifications to spline blocks

Description

- Modification of the position of the point:

  If a point within a spline block is offset, the path is modified, at most, in the 2 segments before this point and the 2 segments after it.

  Small point offsets generally result in small modifications to the path. If, however, very long segments are followed by very short segments or vice versa, small modifications can have a very great effect, as the tangents and curves can change very greatly in such cases.
- Modification of the segment type:

  If an SPL segment is changed into an SLIN segment or vice versa, the path changes in the previous segment and the next segment.

Example 1

```
PTP P0
SPLINE
 SPL P1
 SPL P2
 SPL P3
 SPL P4
 SCIRC P5, P6
 SPL P7
 SLIN P8
ENDSPLINE
```

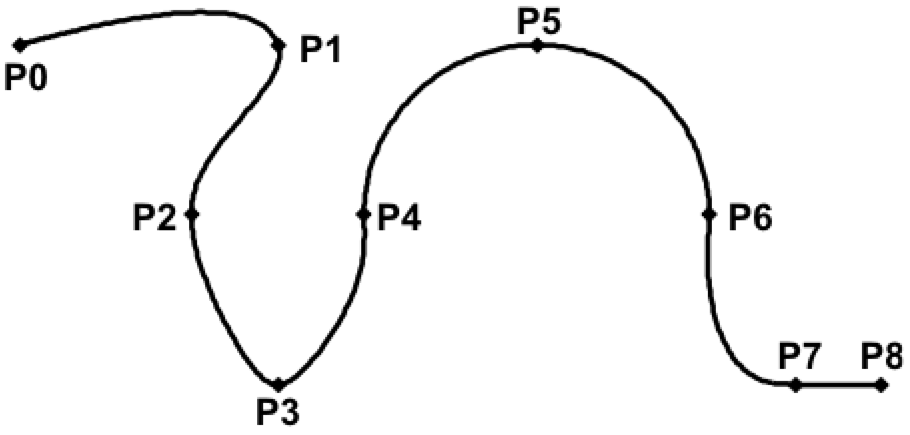


Fig. 9-23: Original path

P3 is offset. This causes the path to change in segments P1 - P2, P2 - P3 and P3 - P4. Segment P4 - P5 is not changed in this case, as it belongs to an SCIRC and a circular path is thus defined.

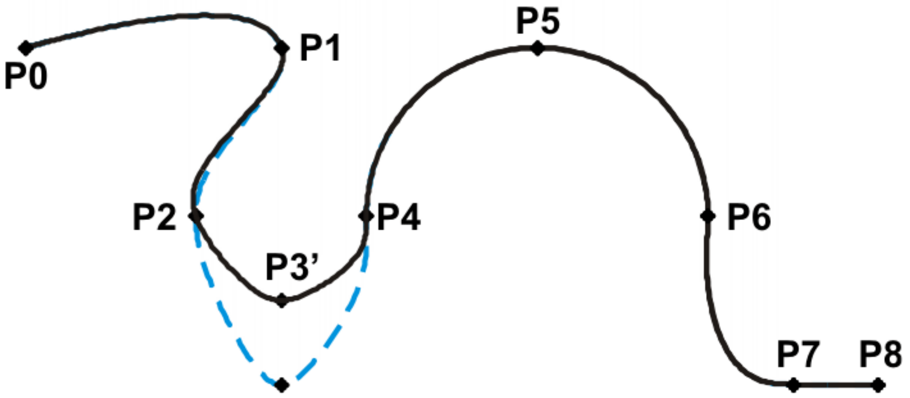


Fig. 9-24: Point has been offset

In the original path, the segment type of P2 - P3 is changed from SPL to SLIN. The path changes in segments P1 - P2, P2 - P3 and P3 - P4.

```
PTP P0
SPLINE
 SPL P1
 SPL P2
 SLIN P3
 SPL P4
 SCIRC P5, P6
 SPL P7
 SLIN P8
ENDSPLINE
```

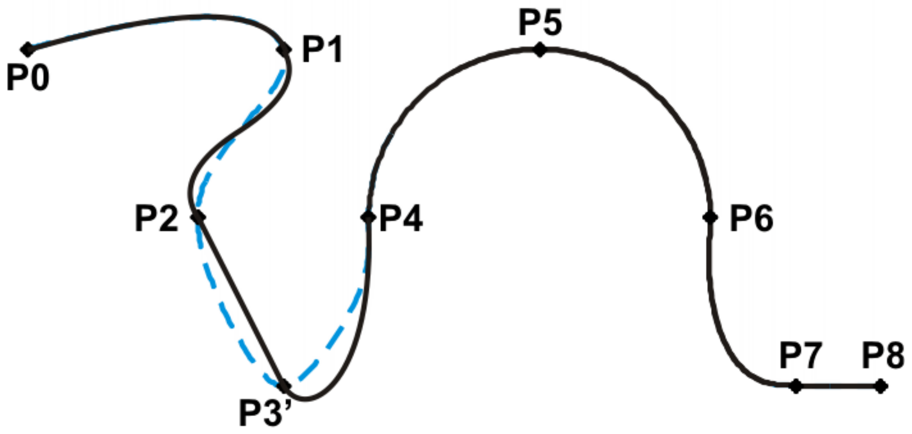


Fig. 9-25: Segment type has been changed

Example 2

```
...
SPLINE
 SPL {X 100, Y 0, ...}
 SPL {X 102, Y 0}
 SPL {X 104, Y 0}
 SPL {X 204, Y 0}
ENDSPLINE
```

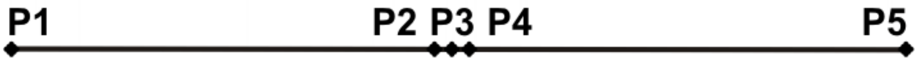


Fig. 9-26: Original path

P3 is offset. This causes the path to change in all the segments illustrated. Since P2 - P3 and P3 - P4 are very short segments and P1 - P2 and P4 - P5 are long segments, the slight offset causes the path to change greatly.

```
...
SPLINE
 SPL {X 100, Y 0, ...}
 SPL {X 102, Y 1}
 SPL {X 104, Y 0}
 SPL {X 204, Y 0}
ENDSPLINE
```

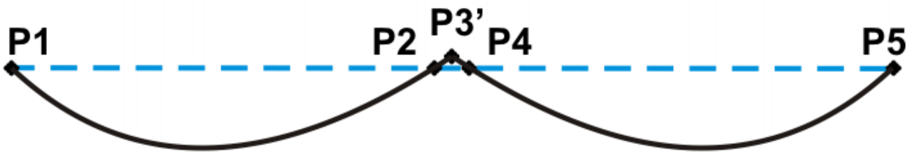


Fig. 9-27: Point has been offset

Remedy:

- Distribute the points more evenly
- Program straight lines (except very short ones) as SLIN segments

9.7.4 Approximate positioning of spline motions

Approximate positioning between spline motions (individual SLIN and SCIRC motions and spline blocks) is possible.

Approximate positioning between spline motions and LIN, CIRC or PTP is not possible.

Approximation not possible due to time or advance run stop:

If approximation is not possible for reasons of time or due to an advance run stop, the robot waits at the start of the approximate positioning arc.

- In the case of time reasons: the robot moves again as soon as it has been possible to plan the next block.
- In the case of an advance run stop: the end of the current block is reached at the start of the approximate positioning arc. This means that the advance run stop is canceled and the robot controller can plan the next block. Robot motion is resumed.

In both cases, the robot now moves along the approximate positioning arc. Approximate positioning is thus technically possible; it is merely delayed.

This response differs from that for LIN, CIRC or PTP motions. If approximate positioning is not possible for the reasons specified, the motion is executed to the end point with exact positioning.

No approximate positioning in MSTEP and ISTEP:

In the program run modes MSTEP and ISTEP, the robot stops exactly at the end point, even in the case of approximated motions.

In the case of approximate positioning from one spline block to another spline block, the result of this exact positioning is that the path is different in the last segment of the first block and in the first segment of the second block from the path in program run mode GO.

In all other segments of both spline blocks, the path is identical in MSTEP, ISTEP and GO.

9.7.5 Replacing an approximated motion with a spline block

Description

In order to replace conventional approximated motions with spline blocks, the program must be modified as follows:

- Replace LIN - LIN with SLIN - SPL - SLIN.
- Replace LIN - CIRC with SLIN - SPL - SCIRC.

  Recommendation: Allow the SPL to project a certain way into the original circle. The SCIRC thus starts later than the original CIRC.

In approximated motions, the corner point is programmed. In the spline block, the points at the start and end of the approximation are programmed instead.

> The approximate positioning arc of the approximated motions varies according to the override. For this reason, when reproducing an approximated motion, it must be ensured that it is executed with the desired override.

The following approximated motion is to be reproduced:

```
LIN P1 C_DIS
LIN P2
```

Spline motion:

```
SPLINE
 SLIN P1A
 SPL P1B
 SLIN P2
ENDSPLINE
```

P1A = start of approximation, P1B = end of approximation

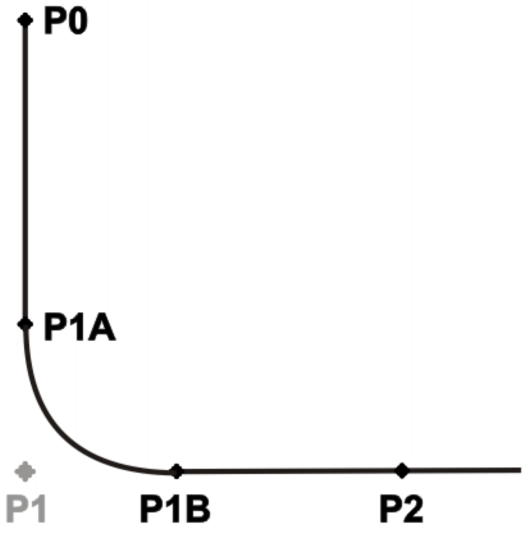


Fig. 9-28: Approximated motion - spline motion

Ways of determining P1A and P1B:

- Execute the approximated path and save the positions at the desired point by means of Trigger.
- Calculate the points in the program with KRL.
- The start of the approximation can be determined from the approximate positioning criterion. Example: If C_DIS is specified as the approximate positioning criterion, the distance from the start of the approximation to the corner point corresponds to the value of $APO.CDIS.

  The end of the approximation is dependent on the programmed velocity.

The SPL path does not correspond exactly to the approximate positioning arc, even if P1A and P1B are exactly at the start/end of the approximation. In order to recreate the exact approximate positioning arc, additional points must be inserted into the spline. Generally, one point is sufficient.

Example

The following approximated motion is to be reproduced:

```
$APO.CDIS=20
$VEL.CP=0.5
LIN {Z 10} C_DIS
LIN {Y 60}
```

Spline motion:

```
SPLINE WITH $VEL.CP=0.5
 SLIN {Z 30}
 SPL {Y 30, Z 10}
 SLIN {Y 60}
ENDSPLINE
```

The start of the approximate positioning arc has been calculated from the approximate positioning criterion.

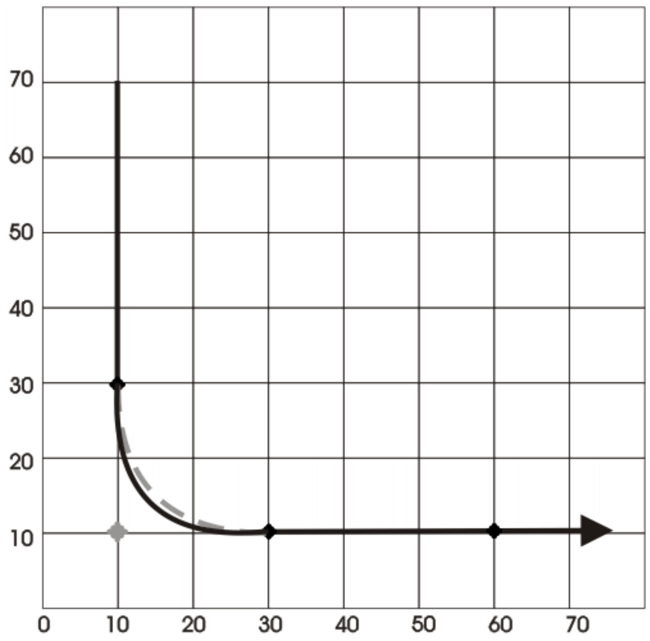


Fig. 9-29: Example: Approximated motion - spline motion 1

The SPL path does not yet correspond exactly to the approximate positioning arc. For this reason, an additional SPL segment is inserted into the spline.

```
SPLINE WITH $VEL.CP=0.5
 SLIN {Z 30}
 SPL {Y 15, Z 15}
 SPL {Y 30, Z 10}
 SLIN {Y 60}
ENDSPLINE
```

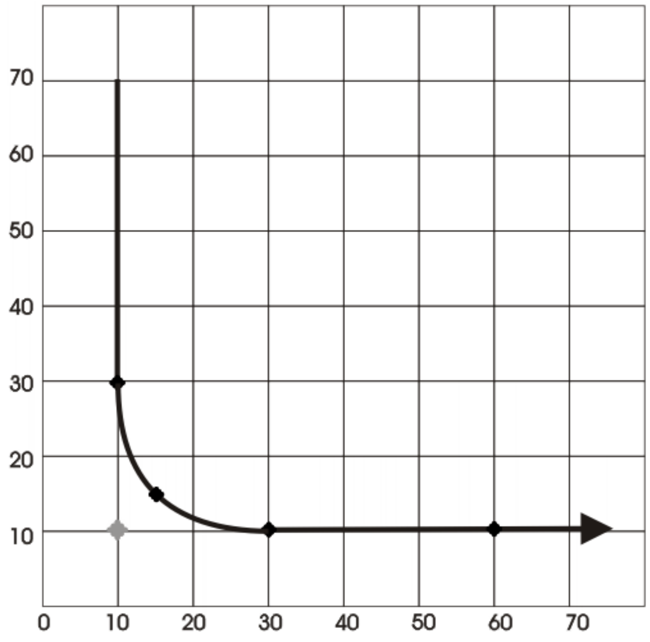


Fig. 9-30: Example: Approximated motion - spline motion 2

With the additional point, the path now corresponds to the approximate positioning arc.

9.8 Orientation control SPLINE

Description

The orientation of the TCP can be different at the start point and end point of a motion. When a CP motion is programmed, it is necessary to select how to deal with the different orientations.

The orientation control for SLIN and SCIRC motions is defined as follows:

- If programming with KRL syntax: by means of the system variable $ORI_TYPE
- If programming with inline forms: in the option window **Motion parameters**

(>>> 10.3.2.2 "Option window “Motion parameter” (SLIN)" Page 282)

(>>> 10.3.3.2 "Option window “Motion parameter” (SCIRC)" Page 284)

(>>> 10.3.4.3 "Option window “Motion parameter” (spline block)" Page 287)

(>>> 10.3.4.8 "Option window “Motion parameter” (spline segment)" Page 290)

Orientation control	Description
■ Option window: **Constant orientation** ■ $ORI_TYPE = #CONSTANT	The orientation of the TCP remains constant during the motion. The orientation of the start point is retained. The programmed orientation of the end point is not taken into consideration.
■ Option window: **Standard** ■ $ORI_TYPE = #VAR	The orientation of the TCP changes continuously during the motion. At the end point, the TCP has the programmed orientation.
■ Option window: **Ignore orientation** ■ $ORI_TYPE = #IGNORE	This option is only available for spline segments (not for the spline block or for individual spline motions). It is used if no specific orientation is required at a point. (>>> "#IGNORE" Page 262)

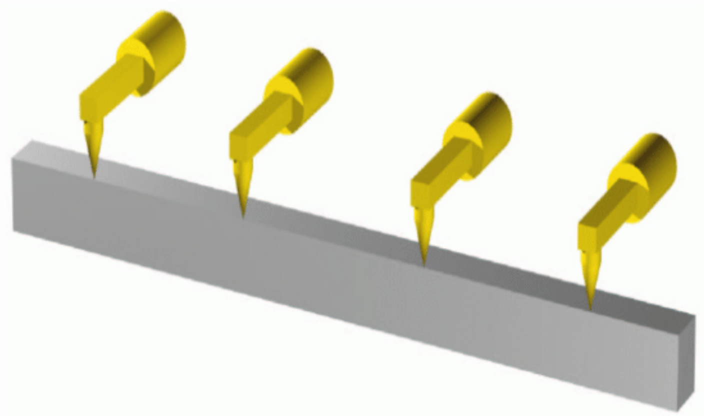


Fig. 9-31: Orientation control - Constant

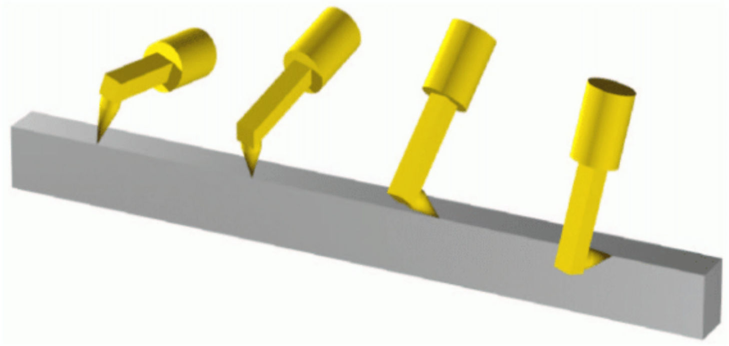


Fig. 9-32: Standard

#IGNORE

$ORI_TYPE = #IGNORE is used if no specific orientation is required at a point. If this option is selected, the taught or programmed orientation of the point is ignored. Instead, the robot controller calculates the optimal orientation for this point on the basis of the orientations of the surrounding points.

Example:

```
SPLINE
  SPL XP1
  SPL XP2
  SPL XP3 WITH $ORI_TYPE=#IGNORE
  SPL XP4 WITH $ORI_TYPE=#IGNORE
  SPL XP5
  SPL XP6
ENDSPLINE
```

The taught or programmed orientation of XP3 and XP4 is ignored.

Characteristics of $ORI_TYPE = #IGNORE:

- In the program run modes MSTEP and ISTEP, the robot stops with the orientations calculated by the robot controller.
- In the case of a block selection to a point with #IGNORE, the robot adopts the orientation calculated by the robot controller.

$ORI_TYPE = #IGNORE is not allowed for the following segments:

- The first segment in a spline block
- The last segment in a spline block
- SCIRC segments with $CIRC_TYPE=#PATH
- Segments followed by a SCIRC segment with $CIRC_TYPE=#PATH
- Segments followed by a segment with $ORI_TYPE=#CONSTANT
- In the case of successive segments with identical Cartesian end points, #IGNORE is not allowed for the first and last segments.

SCIRC

It is also possible to define for SCIRC motions whether the orientation control is to be space-related or path-related.

(>>> 9.8.1 "SCIRC: reference system for the orientation control" Page 263)

It is also possible to define for SCIRC motions whether, and to what extent, the orientation of the auxiliary point is to be taken into consideration. The orientation behavior at the end point can also be defined.

(>>> 9.8.2 "SCIRC: orientation behavior" Page 263)

9.8.1 SCIRC: reference system for the orientation control

It is possible to define for SCIRC motions whether the orientation control is to be space-related or path-related. This can be defined as follows:

- If programming with inline forms: in the option window **Motion parameter**
- If programming with KRL syntax: by means of the system variable $CIRC_TYPE

Orientation control	Description
■ Option window: **Base-related** ■ $CIRC_TYPE = #BASE	Base-related orientation control during the circular motion
■ Option window: **Path-related** ■ $CIRC_TYPE = #PATH	Path-related orientation control during the circular motion

$CIRC_TYPE = #PATH is not allowed for the following motions:

- SCIRC segments for which $ORI_TYPE = #IGNORE
- SCIRC motions preceded by a spline segment for which $ORI_TYPE = #IGNORE

(>>> 9.6.1 "Combinations of $ORI_TYPE and $CIRC_TYPE" Page 249)

9.8.2 SCIRC: orientation behavior

Description

During SCIRC motions, the robot controller can take the programmed orientation of the auxiliary point into consideration. $CIRC_MODE can be used to define whether and to what extent it is taken into consideration.

In the case of SCIRC statements with circular angles, $CIRC_MODE can also be used to define whether the end point is to have the programmed orientation or whether the orientation is to be scaled according to the circular angle.

$CIRC_MODE can only be written to by means of a SCIRC statement. $CIRC_MODE cannot be read.

Syntax

For auxiliary points:

`$CIRC_MODE.AUX_PT.ORI =` *BehaviorAUX*

For end points:

`$CIRC_MODE.TARGET_PT.ORI =` *BehaviorEND*

Explanation of the syntax

Element	Description
BehaviorAUX	Data type: ENUM - **#INTERPOLATE**: The programmed orientation is accepted in the auxiliary point. - **#IGNORE**: The transition from the start orientation to the end orientation is carried out over the shortest possible distance. The programmed orientation of the auxiliary point is disregarded. - **#CONSIDER** (default): The transition from the start orientation to the end orientation passes through the programmed orientation of the auxiliary point, i.e. the orientation of the auxiliary point is accepted at some point during the transition, but not necessarily at the auxiliary point.
BehaviorEND	Data type: ENUM - **#INTERPOLATE**: The programmed orientation of the end point is accepted at the actual end point. (Only possibility for SCIRC without specification of circular angle. If #EXTRAPOLATE is set, #INTERPOLATE is nonetheless executed.) - **#EXTRAPOLATE**: The programmed orientation is accepted at the programmed end point. The orientation at the actual end point is scaled according to the circular angle. (Default for SCIRC with specification of circular angle.)

Restrictions

- If $ORI_TYPE = #IGNORE for a SCIRC segment, $CIRC_MODE is not evaluated.
- If a SCIRC segment is preceded by a SCIRC or SLIN segment with $ORI_TYPE = #IGNORE, #CONSIDER cannot be used in this SCIRC segment.

For SCIRC with circular angle:

- #INTERPOLATE must not be set for the auxiliary point.
- If $ORI_TYPE = #IGNORE, #EXTRAPOLATE must not be set for the end point.
- If it is preceded by a spline segment with $ORI_TYPE = #IGNORE, #EXTRAPOLATE must not be set for the end point.

Example: Auxiliary point

The TCP executes an arc with a Cartesian angle of 192°.

- The orientation at the start point is 0°.
- The orientation at the auxiliary point is 98°.
- The orientation at the end point is 197°.

The re-orientation is thus 197° if the auxiliary point is taken into consideration.

If the orientation at the auxiliary point is ignored, the end orientation can also be achieved by means of a re-orientation of 360° - 197° = 163°.

- **#INTERPOLATE**:

  The programmed orientation of 98° is accepted in the auxiliary point. The re-orientation is thus 197°.
- **#IGNORE**:

  The programmed orientation of the auxiliary point is disregarded. The shorter re-orientation through 163° is used.

- **#CONSIDER**:
  The distance traveled includes the orientation of the auxiliary point, in this case the re-orientation through 197°, i.e. the 98° are accepted at some point during the transition, but not necessarily at the auxiliary point.

Example: End point

The example schematically illustrates the behavior of #INTERPOLATE and #EXTRAPOLATE.

- The pale, dotted arrows show the programmed orientation.
- The dark arrows show the actual orientation where this differs from the programmed orientation.

#INTERPOLATE:

At **TP**, which is situated before **TP_CA**, the programmed orientation has not yet been reached. The programmed orientation is accepted at **TP_CA**.

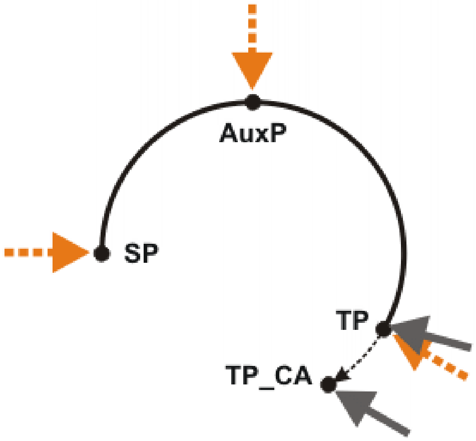


Fig. 9-33: #INTERPOLATE

SP	Start point
AuxP	Auxiliary point
TP	Programmed end point
TP_CA	Actual end point. Determined by the circular angle.

#EXTRAPOLATE:

The programmed orientation is accepted at **TP**. For **TP_CA**, this orientation is scaled in accordance with the circular angle.

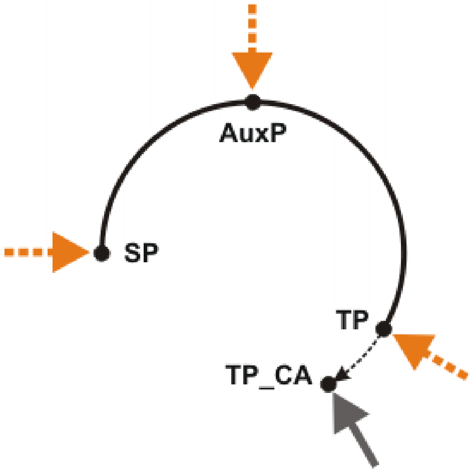


Fig. 9-34: #EXTRAPOLATE

9.9 Status and Turn

Overview

The position (X, Y, Z) and orientation (A, B, C) values of the TCP are not sufficient to define the robot position unambiguously, as different axis positions are possible for the same TCP. Status and Turn serve to define an unambiguous position that can be achieved with different axis positions.

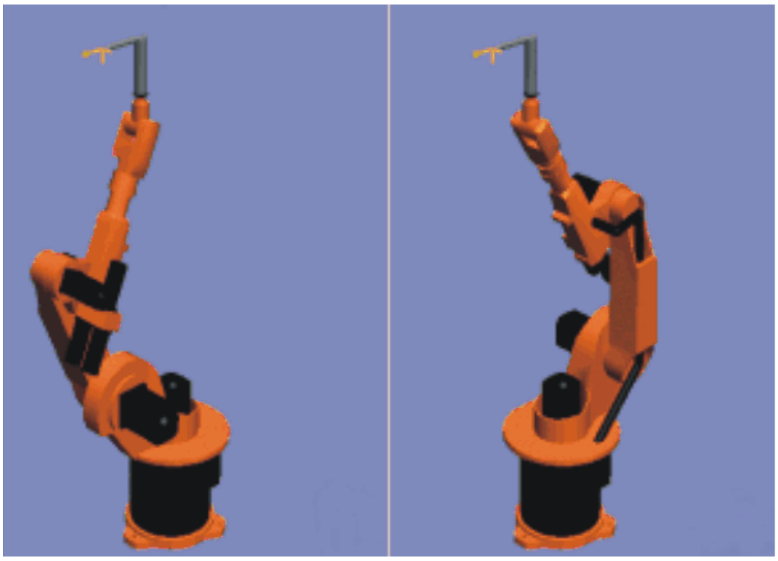


Fig. 9-35: Example: Same TCP position, different axis position

Status (S) and Turn (T) are integral parts of the data types POS and E6POS:

```
STRUC POS REAL X, Y, Z, A, B, C, INT S, T
```

```
STRUC E6POS REAL X, Y, Z, A, B, C, E1, E2, E3, E4, E5, E6, INT S, T
```

KRL program

The robot controller only takes the programmed Status and Turn values into consideration for PTP motions. They are ignored for CP motions.

The first motion instruction in a KRL program must therefore be one of the following instructions so that an unambiguous starting position is defined for the robot:

- A complete PTP instruction of type POS or E6POS
- Or a complete PTP instruction of type AXIS or E6AXIS

“Complete” means that all components of the end point must be specified. The default HOME position is always a complete PTP instruction.

Status and Turn can be omitted in the subsequent instructions:

- The robot controller retains the previous Status value.
- The Turn value is determined by the path in CP motions. In the case of PTP motions, the robot controller selects the Turn value that results in the shortest possible path.

9.9.1 Status

The Status specification prevents ambiguous axis positions.

Bit 0

Bit 0 specifies the position of the intersection of the wrist axes (A4, A5, A6).

Position	Value
Overhead area If the x-value of the intersection of the wrist axes, relative to the A1 coordinate system, is negative, the robot is in the overhead area.	Bit 0 = 1
Basic area If the x-value of the intersection of the wrist axes, relative to the A1 coordinate system, is positive, the robot is in the basic area.	Bit 0 = 0

The A1 coordinate system is identical to the $ROBROOT coordinate system if axis 1 is at 0°. For values not equal to 0°, it moves with axis 1.

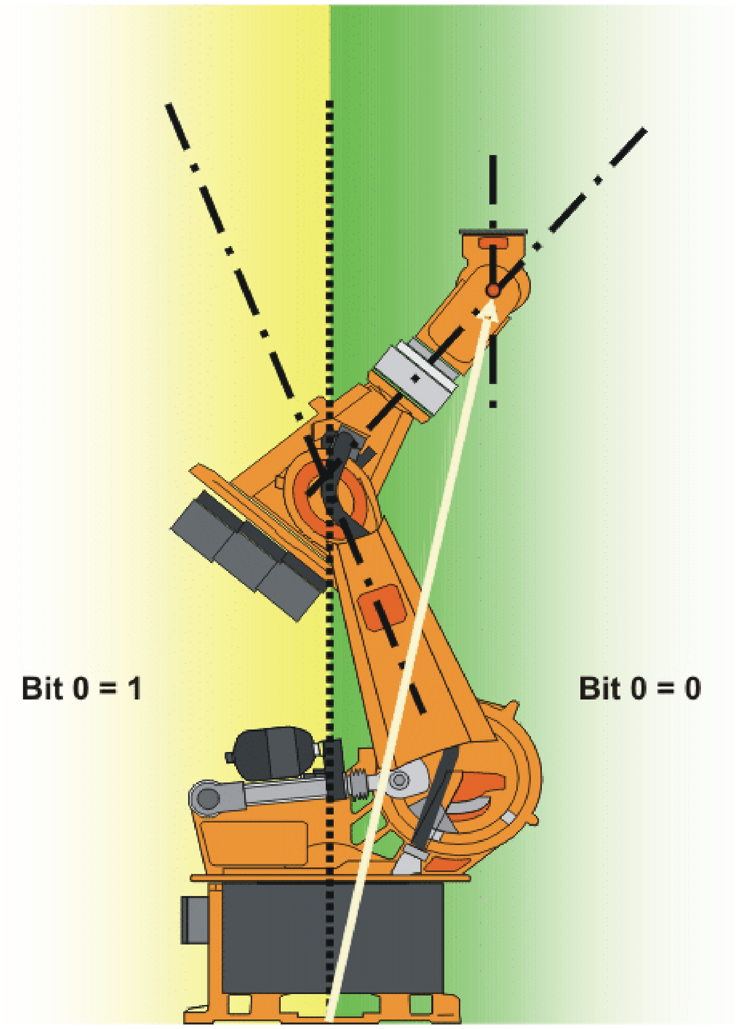


Fig. 9-36: Example: The intersection of the wrist axes (red dot) is in the basic area.

Bit 1

Bit 1 specifies the position of axis 3. The angle at which the value of bit 1 changes depends on the robot type.

For robots whose axes 3 and 4 intersect, the following applies:

Position	Value
A3 ≥ 0°	Bit 1 = 1
A3 < 0°	Bit 1 = 0

For robots with an offset between axis 3 and axis 4, the angle at which the value of bit 1 changes depends on the size of this offset.

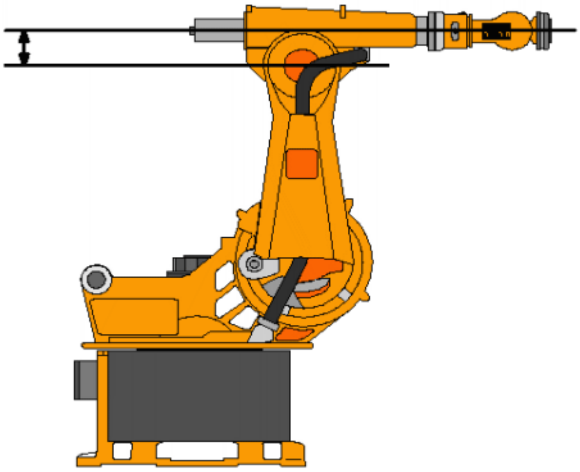


Fig. 9-37: Offset between A3 and A4 – example: KR 30

Bit 2

Bit 2 specifies the position of axis 5.

Position	Value
A5 > 0	Bit 2 = 1
A5 ≤ 0	Bit 2 = 0

Bit 3

Bit 3 is not used and is always 0.

Bit 4

Bit 4 specifies whether or not the point was taught using an absolutely accurate robot.

Depending on the value of the bit, the point can be executed by both absolutely accurate robots and non-absolutely-accurate robots. Bit 4 is for information purposes only and has no influence on how the robot calculates the point. This means, therefore, that when a robot is programmed offline, bit 4 can be ignored.

Description	Value
The point was not taught with an absolutely accurate robot.	Bit 4 = 0
The point was taught with an absolutely accurate robot.	Bit 4 = 1

9.9.2 Turn

Description

The Turn specification makes it possible to move axes through angles greater than +180° or less than -180° without the need for special motion strategies (e.g. auxiliary points). With rotational axes, the individual bits determine the sign before the axis value in the following way:

Bit = 0: angle ≥ 0°

Bit = 1: angle < 0°

Value	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	A6 ≥ 0°	A5 ≥ 0°	A4 ≥ 0°	A3 ≥ 0°	A2 ≥ 0°	A1 ≥ 0°
1	A6 < 0°	A5 < 0°	A4 < 0°	A3 < 0°	A2 < 0°	A1 < 0°

Example

```
DECL POS XP1 = {X 900, Y 0, Z 800, A 0, B 0, C 0, S 6, T 19}
```

T 19 corresponds to T 'B010011'. This means:

Axis	Angle
A1	negative
A2	negative
A3	positive
A4	positive
A5	negative
A6	positive

9.10 Singularities

Overview

The lightweight robot behaves like a KUKA robot with 6 degrees of freedom when, in the case of zero space stiffness, external axis E1 is not moved.

(>>> 9.10.1 "Singularities – 6-axis KUKA robot" Page 270)

External axis E1 is the zero space parameter. If this is selected unfavorably, the A1/A2 singularity can also arise.

(>>> 9.10.2 "A1/A2 singularity" Page 271)

Other singularities can arise if external axis E1 is moved. The lightweight robot with 7 degrees of freedom is in a singularity position when the imaginary line passing through the center of one axis also passes through the center of a second axis.

9.10.1 Singularities – 6-axis KUKA robot

KUKA robots with 6 degrees of freedom have 3 different singularity positions.

- Overhead singularity
- Extended position singularity
- Wrist axis singularity

A singularity position is characterized by the fact that unambiguous reverse transformation (conversion of Cartesian coordinates to axis-specific values) is not possible, even though Status and Turn are specified. In this case, or if very slight Cartesian changes cause very large changes to the axis angles, one speaks of singularity positions.

Overhead

In the overhead singularity, the wrist root point (intersection of axes A4, A5 and A6) is located vertically above axis 1.

The position of axis A1 cannot be determined unambiguously by means of reverse transformation and can thus take any value.

If the end point of a PTP motion is situated in this overhead singularity position, the robot controller may react as follows by means of the system variable $SINGUL_POS[1]:

- **0**: The angle for axis A1 is defined as 0 degrees (default setting).
- **1**: The angle for axis A1 remains the same from the start point to the end point.

Extended position

In the extended position singularity, the wrist root point (intersection of axes A4, A5 and A6) is located in the extension of axes A2 and A3 of the robot.

The robot is at the limit of its work envelope.

Although reverse transformation does provide unambiguous axis angles, low Cartesian velocities result in high axis velocities for axes A2 and A3.

If the end point of a PTP motion is situated in this extended position singularity, the robot controller may react as follows by means of the system variable $SINGUL_POS[2]:

- **0**: The angle for axis A2 is defined as 0 degrees (default setting).
- **1**: The angle for axis A2 remains the same from the start point to the end point.

Wrist axes

In the wrist axis singularity position, the axes A4 and A6 are parallel to one another and axis A5 is within the range ±0.01812°.

The position of the two axes cannot be determined unambiguously by reverse transformation. There is an infinite number of possible axis positions for axes A4 and A6 with identical axis angle sums.

If the end point of a PTP motion is situated in this wrist axis singularity, the robot controller may react as follows by means of the system variable $SINGUL_POS[3]:

- **0**: The angle for axis A4 is defined as 0 degrees (default setting).
- **1**: The angle for axis A4 remains the same from the start point to the end point.

9.10.2 A1/A2 singularity

Description

Axes A1 and A2 form the A1/A2 plane. In the A1/A2 singularity, the zero space parameter is selected such that the TCP has to pass through the A1/A2 plane to reach the programmed end point. In the case of a CP motion, the robot can no longer pass through any part of the A1/A2 plane.

> Exception: If axis E1 is situated in the 0.0 degree position at the start and end point, it is possible to pass through the A1/A2 plane during a CP motion.

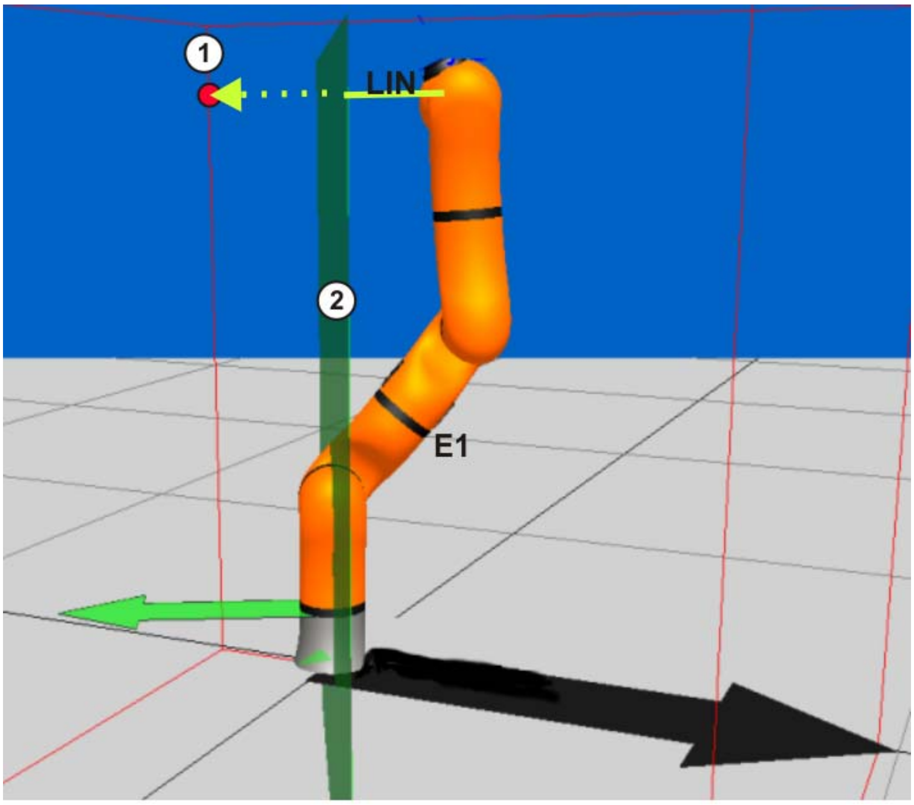


Fig. 9-38: Example of A1/A2 singularity

1	End point of the LIN motion	2	A1/A2 plane

Example

A1/A2 singularity: the TCP cannot pass through the A1/A2 plane with the LIN motion.

```
DEF A1_A2_singular ( )

PTP {A1 0.0, A2 120.0, A3 30.0, A4 0.0, A5 -90.0, A6 0.0, E1 1.0}
$OV_PRO = 10
LIN_REL {X 500}

END
```

10 Programming for user group “User” (inline forms)

Inline forms are available in the KSS for frequently used instructions. They simplify programming.

Instructions can also be programmed without inline forms. Information is contained in the description of the KRL syntax.
(>>> 11 "Programming for user group “Expert” (KRL syntax)" Page 307)

10.1 Names in inline forms

Names for data sets can be entered in inline forms. These include, for example, point names, names for motion data sets, etc.

The following restrictions apply to names:

- Maximum length 23 characters
- No special characters are permissible, with the exception of $.
- The first character must not be a number.

The restrictions do not apply to output names.

Other restrictions may apply in the case of inline forms in technology packages.

10.2 Programming PTP, LIN and CIRC motions

10.2.1 Programming a PTP motion

NOTICE When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

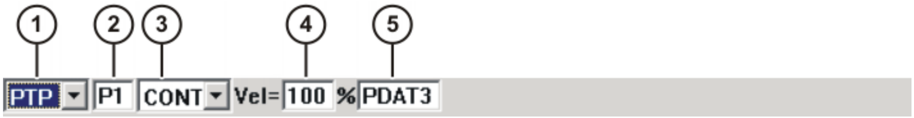
Precondition

- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Move the TCP to the position that is to be taught as the end point.
2. Position the cursor in the line after which the motion instruction is to be inserted.
3. Select the menu sequence **Commands** > **Motion** > **PTP**.
4. Set the parameters in the inline form.
   (>>> 10.2.2 "Inline form for PTP motions" Page 273)
5. Save the instruction by pressing the **Cmd Ok** softkey.

10.2.2 Inline form for PTP motions



PTP P1 CONT Vel=100 % PDAT3

Fig. 10-1: Inline form for PTP motions

Item	Description
1	Motion type - **PTP** - **LIN** - **CIRC**
2	Name of the end point The system automatically generates a name. The name can be overwritten. (>>> 10.1 "Names in inline forms" Page 273) Position the cursor in this box to edit the point data. The corresponding option window is opened. (>>> 10.2.7 "Option window “Frames”" Page 277)
3	- **CONT**: end point is approximated. - **[Empty box]**: the motion stops exactly at the end point.
4	Velocity - **1 … 100%**
5	Name for the motion data set The system automatically generates a name. The name can be overwritten. (>>> 10.1 "Names in inline forms" Page 273) Position the cursor in this box to edit the motion data. The corresponding option window is opened. (>>> 10.2.8 "Option window “Motion parameter” (PTP)" Page 278)

10.2.3 Programming a LIN motion

NOTICE When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

Precondition

- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Move the TCP to the position that is to be taught as the end point.
2. Position the cursor in the line after which the motion instruction is to be inserted.
3. Select the menu sequence **Commands** > **Motion** > **LIN**.
4. Set the parameters in the inline form.
   (>>> 10.2.4 "Inline form for LIN motions" Page 274)
5. Save the instruction by pressing the **Cmd Ok** softkey.

10.2.4 Inline form for LIN motions

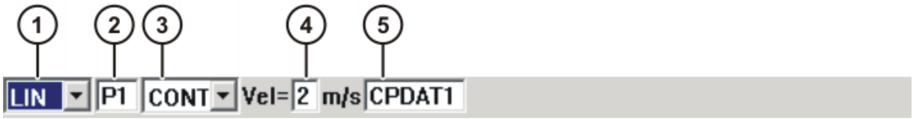


Fig. 10-2: Inline form for LIN motions

Item	Description
1	Motion type - **PTP** - **LIN** - **CIRC**
2	Name of the end point The system automatically generates a name. The name can be overwritten. (>>> 10.1 "Names in inline forms" Page 273) Position the cursor in this box to edit the point data. The corresponding option window is opened. (>>> 10.2.7 "Option window “Frames”" Page 277)
3	- **CONT**: end point is approximated. - **[Empty box]**: the motion stops exactly at the end point.
4	Velocity - **0.001 … 2 m/s**
5	Name for the motion data set The system automatically generates a name. The name can be overwritten. (>>> 10.1 "Names in inline forms" Page 273) Position the cursor in this box to edit the motion data. The corresponding option window is opened. (>>> 10.2.9 "Option window “Motion parameter” (LIN, CIRC)" Page 279)

10.2.5 Programming a CIRC motion

NOTICE When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

Precondition

- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Move the TCP to the position that is to be taught as the auxiliary point.
2. Position the cursor in the line after which the motion instruction is to be inserted.
3. Select the menu sequence **Commands** > **Motion** > **CIRC**.
4. Set the parameters in the inline form.

   (>>> 10.2.6 "Inline form for CIRC motions" Page 276)
5. Press the **Teach Aux** softkey.
6. Move the TCP to the position that is to be taught as the end point.
7. Save the instruction by pressing the **Cmd Ok** softkey.

10.2.6 Inline form for CIRC motions

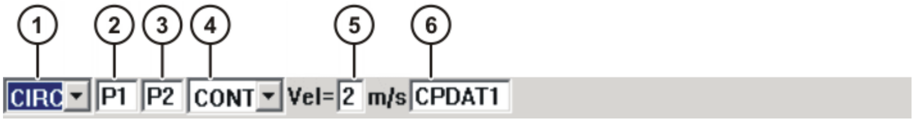


Fig. 10-3: Inline form for CIRC motions

Item	Description
1	Motion type - **PTP** - **LIN** - **CIRC**
2	Name of the auxiliary point The system automatically generates a name. The name can be overwritten. (>>> 10.1 "Names in inline forms" Page 273)
3	Name of the end point The system automatically generates a name. The name can be overwritten. (>>> 10.1 "Names in inline forms" Page 273) Position the cursor in this box to edit the point data. The corresponding option window is opened.
4	- **CONT**: end point is approximated. - **[Empty box]**: the motion stops exactly at the end point.
5	Velocity - **0.001 … 2 m/s**
6	Name for the motion data set The system automatically generates a name. The name can be overwritten. (>>> 10.1 "Names in inline forms" Page 273) Position the cursor in this box to edit the motion data. The corresponding option window is opened.

10.2.7 Option window “Frames”

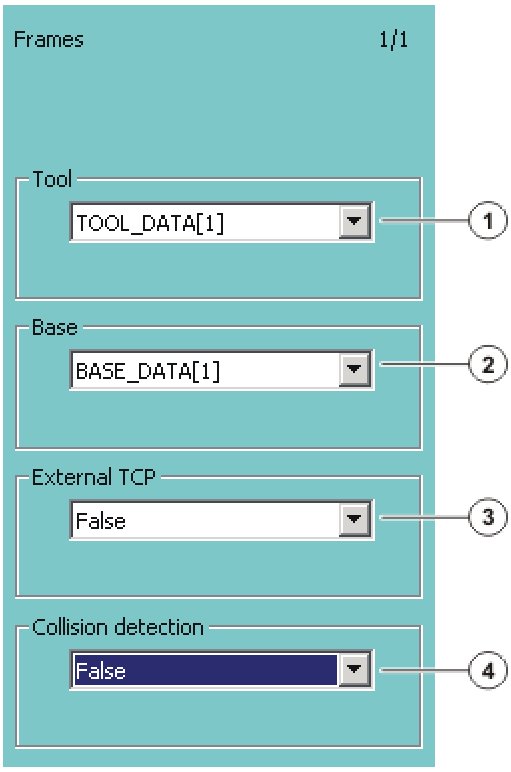
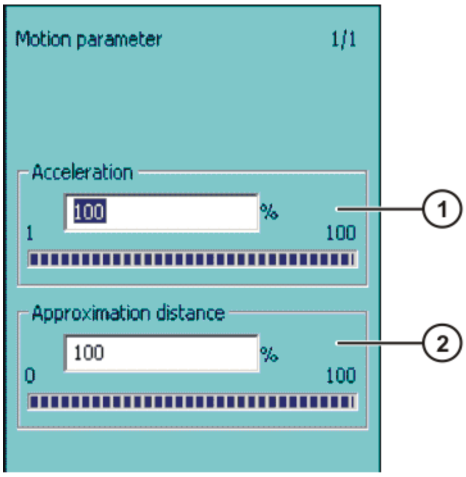


Fig. 10-4: Option window: Frames

Item	Description
1	Tool selection. If **True** in the box **External TCP**: workpiece selection. Range of values: [1] … [16]
2	Base selection. If **True** in the box **External TCP**: fixed tool selection. Range of values: [1] … [32]
3	Interpolation mode ■ **False**: The tool is mounted on the mounting flange. ■ **True**: The tool is a fixed tool.
4	Collision detection is not possible for the lightweight robot (= **False**).

10.2.8 Option window “Motion parameter” (PTP)



Motion parameter 1/1

Acceleration

100 %

1 100

Approximation distance

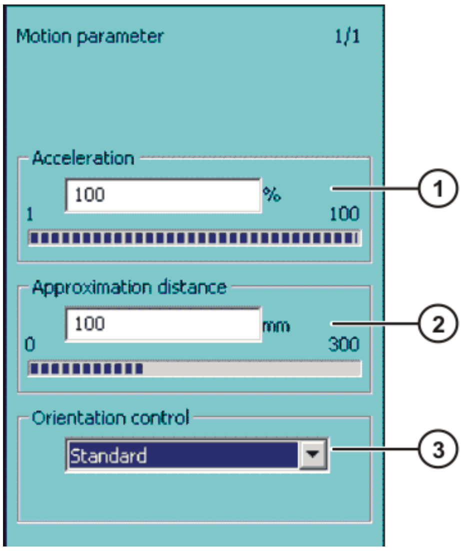
100 %

0 100

Fig. 10-5: Option window “Motion parameter” (PTP)

Item	Description
1	Acceleration Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. - **1 … 100 %**
2	This box is only displayed if **CONT** was selected in the inline form. Furthest distance before the end point at which approximate positioning can begin. Maximum distance 100%: half the distance between the start point and the end point relative to the contour of the PTP motion without approximate positioning The unit for this box can also be mm. This depends on the configuration. (>>> 6.17.2 "Changing the unit of the approximation distance for PTP" Page 159) Distance in mm: The maximum permissible value is half the distance between the start point and the end point. If a higher value is entered, this is ignored and the maximum value is used. - **1 … 100 %** - Or **0 … 300 mm**

10.2.9 Option window “Motion parameter” (LIN, CIRC)



Motion parameter 1/1

Acceleration: 100 % (1 – 100)

Approximation distance: 100 mm (0 – 300)

Orientation control: Standard

Fig. 10-6: Option window “Motion parameter” (LIN, CIRC)

Item	Description
1	Acceleration Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode.
2	Furthest distance before the end point at which approximate positioning can begin The maximum permissible value is half the distance between the start point and the end point. If a higher value is entered, this is ignored and the maximum value is used. This box is only displayed if **CONT** was selected in the inline form.
3	Orientation control selection. ■ **Standard** ■ **Wrist PTP**: This orientation type is not supported. ■ **Constant orientation** (>>> 9.6 "Orientation control LIN, CIRC" Page 246)

10.3 Spline motions

10.3.1 Programming tips for spline motions

- A spline block should cover only 1 process (e.g. 1 adhesive seam). More than one process in a spline block leads to a loss of structural clarity within the program and makes changes more difficult.
- Use SLIN and SCIRC segments in cases where the workpiece necessitates straight lines and arcs. (Exception: use SPL segments for very short straight lines.) Otherwise, use SPL segments, particularly if the points are close together.
- Procedure for defining the path:

  a. First teach or calculate a few characteristic points. Example: points at which the curve changes direction.

b. Test the path. At points where the accuracy is still insufficient, add more SPL points.

- Avoid successive SLIN and/or SCIRC segments, as this often reduces the velocity to 0.

  Program SPL segments between SLIN and SCIRC segments. The length of the SPL segments must be at least > 0.5 mm. Depending on the actual path, significantly larger SPL segments may be required.
- Avoid successive points with identical Cartesian coordinates, as this reduces the velocity to 0.
- The parameters (tool, base, velocity, etc.) assigned to the spline block have the same effect as assignments before the spline block. The assignment to the spline block has the advantage, however, that the correct parameters are read in the case of a block selection.
- Use the option **Ignore Orientation** if no specific orientation is required at a point. The robot controller calculates the optimal orientation for this point on the basis of the orientations of the surrounding points. This way, even large changes in orientation between two points are optimally distributed over the points in between.
- Jerk limitation can be programmed. The jerk is the change in acceleration.

  Procedure:

  a. Use the default values initially.

  b. If vibrations occur at tight corners: reduce values.

  If the velocity drops or the desired velocity cannot be reached: increase values or increase acceleration.
- If the robot executes points on a work surface, a collision with the work surface is possible when the first point is addressed.

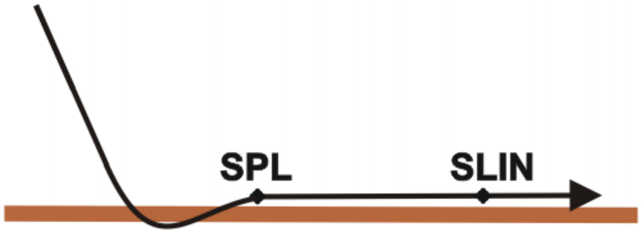


Fig. 10-7: Collision with work surface

In order to avoid a collision, observe the recommendations for the SLIN-SPL-SLIN transition.

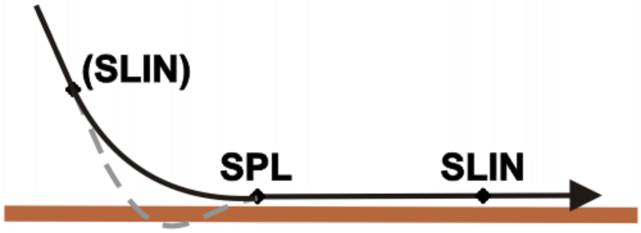


Fig. 10-8: Avoiding a collision with the work surface

10.3.2 Programming a SLIN motion (individual motion)

NOTICE When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

Precondition

- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Move the TCP to the end point.
2. Position the cursor in the line after which the motion is to be inserted.

   (But not within a spline block. This opens another inline form.)
   (>>> 10.3.4.6 "Inline form for spline segment" Page 288)
3. Select **Commands** > **Motion** > **SLIN**.
4. Set the parameters in the inline form.

   (>>> 10.3.2.1 "Inline form “SLIN”" Page 281)
5. Press **Cmd OK**.

10.3.2.1 Inline form “SLIN”

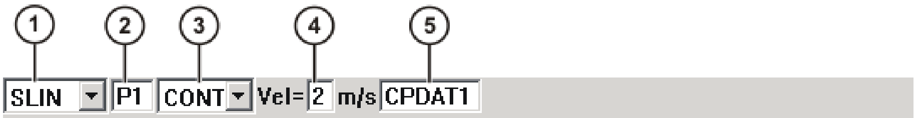
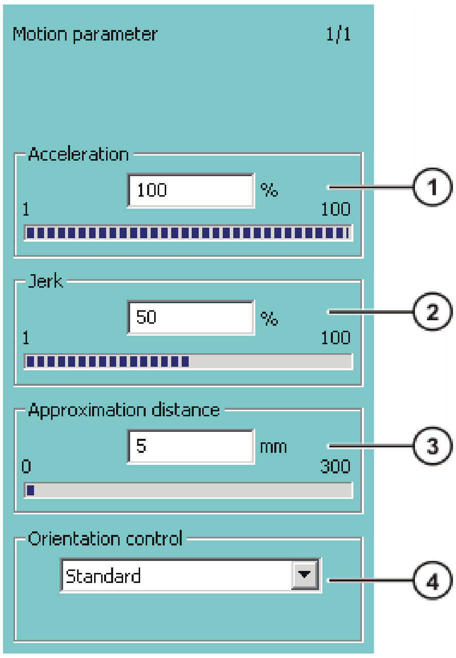


Fig. 10-9: Inline form “SLIN” (individual motion)

Item	Description
1	Motion type **SLIN**
2	Point name for end point. The system automatically generates a name. The name can be overwritten. (>>> 10.1 "Names in inline forms" Page 273) Position the cursor in this box to edit the point data. The corresponding option window is opened. (>>> 10.2.7 "Option window “Frames”" Page 277)
3	■ **CONT**: end point is approximated. ■ **[Empty box]**: the motion stops exactly at the end point.
4	Velocity ■ **0.001 … 2 m/s**
5	Name for the motion data set. The system automatically generates a name. The name can be overwritten. Position the cursor in this box to edit the point data. The corresponding option window is opened. (>>> 10.3.2.2 "Option window “Motion parameter” (SLIN)" Page 282)

10.3.2.2 Option window “Motion parameter” (SLIN)



Motion parameter 1/1

Acceleration: 100 % (1 … 100)

Jerk: 50 % (1 … 100)

Approximation distance: 5 mm (0 … 300)

Orientation control: Standard

Fig. 10-10: Option window “Motion parameter” (SLIN)

Item	Description
1	Acceleration. The value refers to the maximum value specified in the machine data. ■ **1 … 100%**
2	Jerk limitation. The jerk is the change in acceleration. The value refers to the maximum value specified in the machine data. ■ **1 … 100%**
3	This box is only displayed if **CONT** was selected in the inline form. Furthest distance before the end point at which approximate positioning can begin. The maximum permissible value is half the distance between the start point and the end point. If a higher value is entered, this is ignored and the maximum value is used.
4	Orientation control selection.

10.3.3 Programming a SCIRC motion (individual motion)

NOTICE When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

Precondition

- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Move the TCP to the auxiliary point.
2. Position the cursor in the line after which the motion is to be inserted.

   (But not within a spline block. This opens another inline form.)
   (>>> 10.3.4.6 "Inline form for spline segment" Page 288)

3. Select the menu sequence **Commands** > **Motion** > **SCIRC**.
4. Set the parameters in the inline form.

   (>>> 10.3.3.1 "Inline form “SCIRC”" Page 283)
5. Press **Teach Aux**.
6. Move the TCP to the end point.
7. Press **Cmd OK**.

10.3.3.1 Inline form “SCIRC”

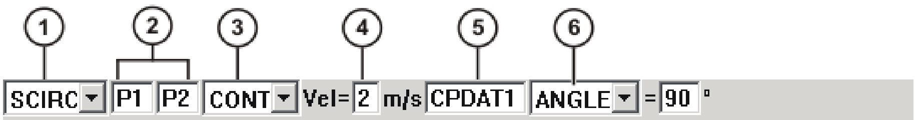
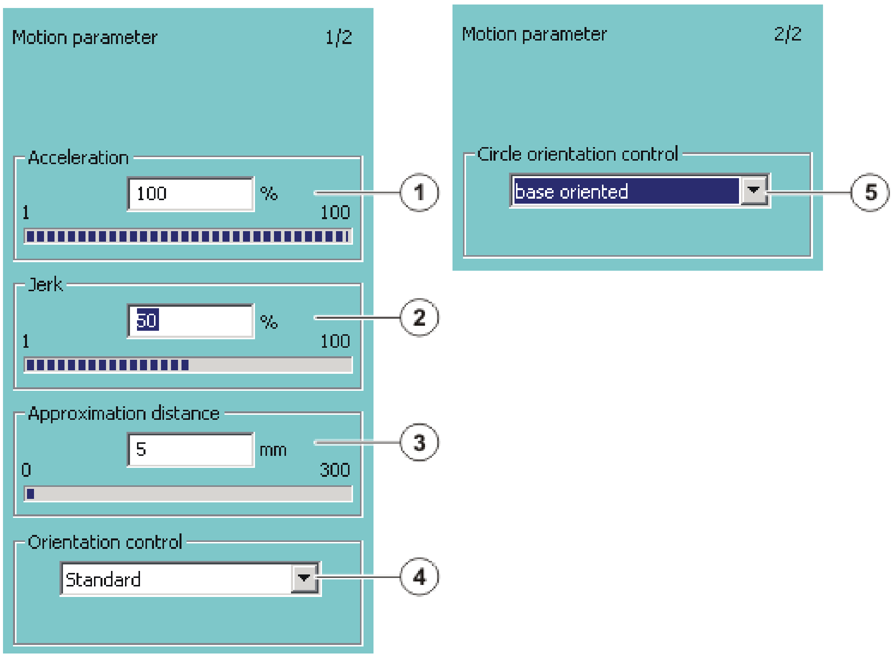


Fig. 10-11: Inline form “SCIRC” (individual motion)

Item	Description
1	Motion type **SCIRC**
2	Point names for auxiliary and end point. The system automatically assigns names. The names can be overwritten. (>>> 10.1 "Names in inline forms" Page 273) Position the cursor in this box to edit the point data. The corresponding option window is opened. (>>> 10.2.7 "Option window “Frames”" Page 277)
3	■ **CONT**: end point is approximated. ■ **[Empty box]**: the motion stops exactly at the end point.
4	Velocity ■ **0.001 … 2 m/s**
5	Name for the motion data set. The system automatically generates a name. The name can be overwritten. Position the cursor in this box to edit the point data. The corresponding option window is opened. (>>> 10.3.3.2 "Option window “Motion parameter” (SCIRC)" Page 284)
6	Specifies the overall angle of the circular motion. This makes it possible to extend the motion beyond the programmed end point or to shorten it. The actual end point thus no longer corresponds to the programmed end point. ■ Positive circular angle: the circular path is executed in the direction Start point › Auxiliary point › End point. ■ Negative circular angle: the circular path is executed in the direction Start point › End point › Auxiliary point. ■ **- 9 999° … + 9 999°** If a circular angle less than -400° or greater than +400° is entered, a request for confirmation is generated when the inline form is saved asking whether entry is to be confirmed or rejected.

10.3.3.2 Option window “Motion parameter” (SCIRC)



Motion parameter 1/2

Acceleration: 100 % (1 … 100)

Jerk: 50 % (1 … 100)

Approximation distance: 5 mm (0 … 300)

Orientation control: Standard

Motion parameter 2/2

Circle orientation control: base oriented

Fig. 10-12: Option window “Motion parameter” (SCIRC)

Item	Description
1	Acceleration. The value refers to the maximum value specified in the machine data. ■ **1 … 100%**
2	Jerk limitation. The jerk is the change in acceleration. The value refers to the maximum value specified in the machine data. ■ **1 … 100%**
3	This box is only displayed if **CONT** was selected in the inline form. Furthest distance before the end point at which approximate positioning can begin. The maximum permissible value is half the distance between the start point and the end point. If a higher value is entered, this is ignored and the maximum value is used.
4	Orientation control selection
5	Orientation control reference system selection.

Circular parameters are displayed on page 3/3 of the option window. The parameters cannot be changed.

10.3.4 Programming a spline block

Description

A spline block can be used to group together several SPL, SLIN and/or SCIRC segments to an overall motion. A spline block that contains no segments is not a motion statement.

A spline block may contain the following:

- Spline segments (only limited by the memory capacity.)
- PATH trigger
- Comments and blank lines

- Inline commands from technology packages that support the spline functionality

A spline block must not include any other instructions, e.g. variable assignments or logic statements. A spline block does not trigger an advance run stop.

Precondition

- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Position the cursor in the line after which the spline block is to be inserted.
2. Select the menu sequence **Commands** > **Motion** > **SPLINE block**.
3. Set the parameters in the inline form.
   (>>> 10.3.4.1 "Inline form for spline block" Page 285)
4. Press **Cmd OK**.
5. Press **Open/close fold**. Spline segments and other lines can now be inserted into the spline block.
   (>>> 10.3.4.4 "Programming an SPL or SLIN segment" Page 287)
   (>>> 10.3.4.5 "Programming an SCIRC segment" Page 288)
   (>>> 10.3.4.9 "Programming PATH triggers in the spline block" Page 291)

10.3.4.1 Inline form for spline block

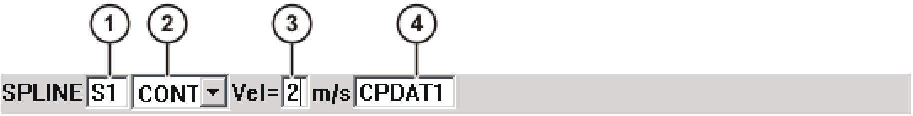
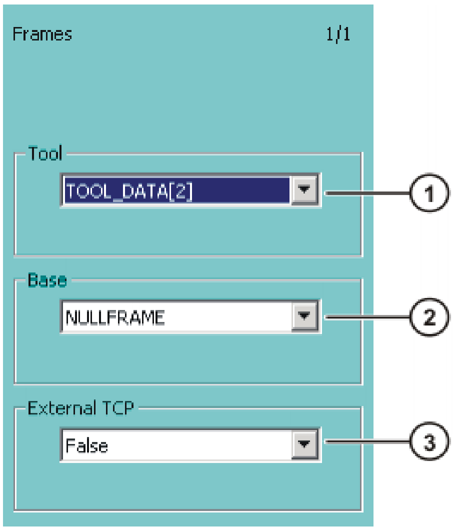

SPLINE S1 CONT Vel= 2 m/s CPDAT1

Fig. 10-13: Inline form for spline block

Item	Description
1	Name of the spline block. The system automatically generates a name. The name can be overwritten. (>>> 10.1 "Names in inline forms" Page 273) Position the cursor in this box to edit the motion data. The corresponding option window is opened. (>>> 10.3.4.2 "Option window “Frames” (spline block)" Page 286)
2	■ **CONT**: end point is approximated. ■ **[Empty box]**: the motion stops exactly at the end point.
3	The velocity is valid by default for the entire spline block. It can also be defined separately for individual segments. ■ **0.001 … 2 m/s**
4	Name for the motion data set. The system automatically generates a name. The name can be overwritten. Position the cursor in this box to edit the motion data. The corresponding option window is opened. (>>> 10.3.4.3 "Option window “Motion parameter” (spline block)" Page 287) The motion data are valid by default for the entire spline block. They can also be defined separately for individual segments.

10.3.4.2 Option window “Frames” (spline block)



Frames 1/1

Tool: TOOL_DATA[2]

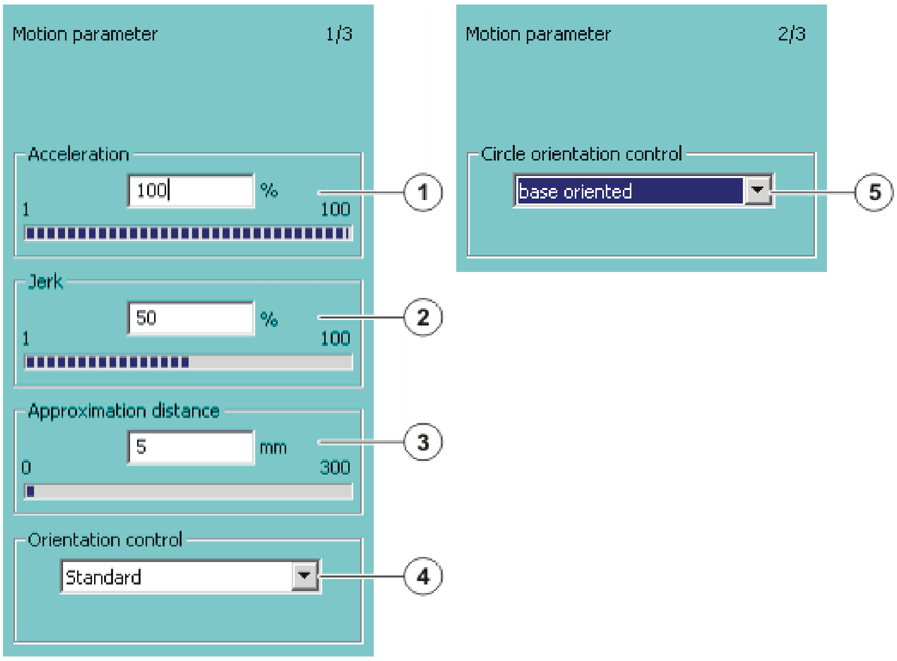
Base: NULLFRAME

External TCP: False

Fig. 10-14: Option window “Frames” (spline block)

Item	Description
1	Tool selection. If **True** in the box **External TCP**: workpiece selection. ■ **[1]** … **[16]**
2	Base selection. If **True** in the box **External TCP**: fixed tool selection. ■ **[1]** … **[32]**
3	Interpolation mode ■ **False**: The tool is mounted on the mounting flange. ■ **True**: The tool is a fixed tool.

10.3.4.3 Option window “Motion parameter” (spline block)



Motion parameter 1/3

Acceleration: 100 % (1 … 100)

Jerk: 50 % (1 … 100)

Approximation distance: 5 mm (0 … 300)

Orientation control: Standard

Motion parameter 2/3

Circle orientation control: base oriented

Fig. 10-15: Option window “Motion parameter” (spline block)

Item	Description
1	Acceleration. The value refers to the maximum value specified in the machine data. ■ **1 … 100%**
2	Jerk limitation. The jerk is the change in acceleration. The value refers to the maximum value specified in the machine data. ■ **1 … 100%**
3	This box is only displayed if **CONT** was selected in the inline form. Furthest distance before the end point at which approximate positioning can begin. The maximum distance is that of the last segment in the spline. If there is only one segment present, the maximum distance is half the segment length. If a higher value is entered, this is ignored and the maximum value is used.
4	Orientation control selection.
5	Orientation control reference system selection. This parameter only affects SCIRC segments (if present) in the spline block.

10.3.4.4 Programming an SPL or SLIN segment

NOTICE When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

Precondition

- Program is selected.
- Operating mode T1 or T2.
- The spline block fold is open.

Procedure

1. Move the TCP to the end point.
2. Position the cursor in the line after which the segment is to be inserted in the spline block.
3. Select the menu sequence **Commands** > **Motion** > **SPL** or **SLIN**.
4. Set the parameters in the inline form.

   (>>> 10.3.4.6 "Inline form for spline segment" Page 288)
5. Press **Cmd OK**.

10.3.4.5 Programming an SCIRC segment

NOTICE When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

Precondition

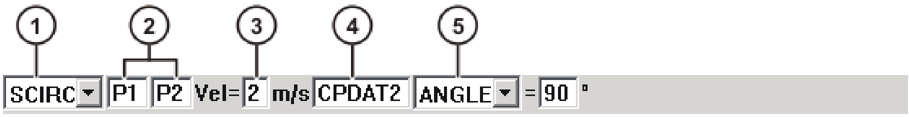
- Program is selected.
- Operating mode T1 or T2.
- The spline block fold is open.

Procedure

1. Move the TCP to the auxiliary point.
2. Position the cursor in the line after which the segment is to be inserted in the spline block.
3. Select the menu sequence **Commands** > **Motion** > **SCIRC**.
4. Set the parameters in the inline form.

   (>>> 10.3.4.6 "Inline form for spline segment" Page 288)
5. Press **Teach Aux**.
6. Move the TCP to the end point.
7. Press **Cmd OK**.

10.3.4.6 Inline form for spline segment



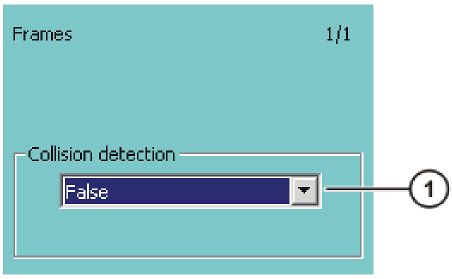
SCIRC P1 P2 Vel=2 m/s CPDAT2 ANGLE = 90 °

Fig. 10-16: Inline form for spline segment

The boxes in the inline form can be displayed or hidden one by one using the **Toggle Param** softkey.

Item	Description
1	Motion type ■ **SPL** ■ **SLIN** ■ **SCIRC**
2	Point name for end point. Only for SCIRC: point names for auxiliary point and end point. The system automatically generates a name. The name can be overwritten. (>>> 10.1 "Names in inline forms" Page 273) Position the cursor in this box to edit the point data. The corresponding option window is opened. (>>> 10.3.4.7 "Option window “Frames” (spline segment)" Page 290)
3	Velocity This only refers to the segment to which it belongs. It has no effect on subsequent segments. ■ **0.001 … 2 m/s**
4	Name for the motion data set. The system automatically generates a name. The name can be overwritten. Position the cursor in this box to edit the point data. The corresponding option window is opened. (>>> 10.3.4.8 "Option window “Motion parameter” (spline segment)" Page 290) The motion data only refer to the segment to which they belong. They have no effect on subsequent segments.
5	Only available if the motion type **SCIRC** has been selected. Specifies the overall angle of the circular motion. This makes it possible to extend the motion beyond the programmed end point or to shorten it. The actual end point thus no longer corresponds to the programmed end point. ■ Positive circular angle: the circular path is executed in the direction Start point › Auxiliary point › End point. ■ Negative circular angle: the circular path is executed in the direction Start point › End point › Auxiliary point. ■ **- 9 999° … + 9 999°** If a circular angle less than -400° or greater than +400° is entered, a request for confirmation is generated when the inline form is saved asking whether entry is to be confirmed or rejected.

10.3.4.7 Option window “Frames” (spline segment)



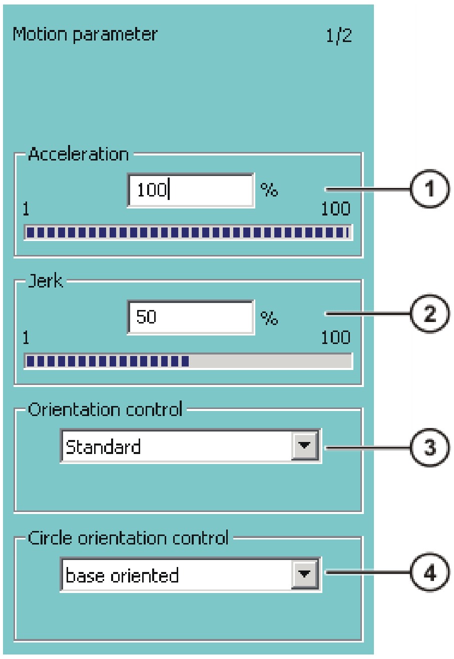
Frames 1/1

Collision detection: False

Fig. 10-17: Option window “Frames” (spline segment)

Item	Description
1	Collision detection is not possible for the lightweight robot (= **False**).

10.3.4.8 Option window “Motion parameter” (spline segment)



Motion parameter 1/2

Acceleration: 100 % (1 … 100)

Jerk: 50 % (1 … 100)

Orientation control: Standard

Circle orientation control: base oriented

Fig. 10-18: Option window “Motion parameter” (spline segment)

Item	Description
1	Acceleration. The value refers to the maximum value specified in the machine data. ■ **1 … 100%**
2	Jerk limitation. The jerk is the change in acceleration. The value refers to the maximum value specified in the machine data. ■ **1 … 100%**
3	Orientation control selection
4	Orientation control reference system selection.

Circular parameters are displayed on page 2/2 of the option window. The parameters cannot be changed.

10.3.4.9 Programming PATH triggers in the spline block

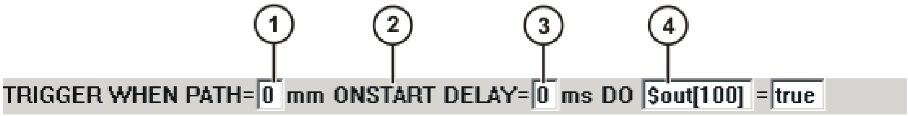
Precondition

- Program is selected.
- Operating mode T1 or T2.
- The spline block fold is open.

Procedure

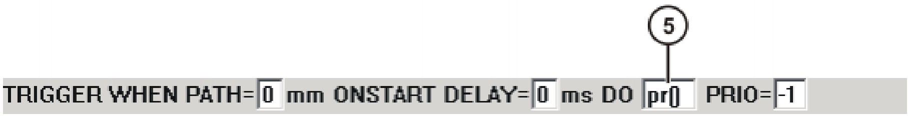
1. Position the cursor in the line after which the trigger is to be inserted in the spline block.
2. Select the menu sequence **Commands** > **Logic** > **Spline trigger**.
3. Set the parameters in the inline form.
   (>>> 10.3.4.10 "Inline form “PATH trigger”" Page 291)
4. Press **Cmd OK**.

10.3.4.10 Inline form “PATH trigger”



TRIGGER WHEN PATH= 0 mm ONSTART DELAY= 0 ms DO $out[100] = true

Fig. 10-19: Inline form “PATH trigger”, with assignment



TRIGGER WHEN PATH= 0 mm ONSTART DELAY= 0 ms DO pr() PRIO= -1

Fig. 10-20: Inline form “PATH trigger”, with subprogram call

Item	Description
1	If the statement is to be shifted in space, the desired distance from the start or end point must be specified here. If no shift in space is desired, enter the value 0. ■ Positive value: shifts the statement towards the end of the motion. ■ Negative value: shifts the statement towards the start of the motion. Only for the user group “Expert”: **Toggle Path** makes it possible to enter a variable, constant or function in this box.
2	**Toggle OnStart** can be used to set or cancel the parameter **ONSTART**. ■ Without **ONSTART**: the PATH value refers to the end point. ■ With **ONSTART**: the PATH value refers to the start point.
3	If the statement is to be shifted in time (relative to the value in item 1), the desired duration must be specified here. If no shift in time is desired, enter the value 0. ■ Positive value: shifts the statement towards the end of the motion. Maximum: 1,000 ms ■ Negative value: shifts the statement towards the start of the motion. Only for the user group “Expert”: the softkey **Toggle Delay** makes it possible to enter a variable, constant or function in this box.
4, 5	Boxes for either a value assignment or a subprogram call are displayed here. **Toggle Type** can be used to switch between the two box types.

Item	Description
4	Value assignment. Possible: - Assignment of a value to a variable - OUT statement
5	Subprogram call A priority must be entered in the **PRIO** box. Priorities 1, 2, 4 to 39 and 81 to 128 are available. Priorities 3 and 40 to 80 are reserved for cases in which the priority is automatically assigned by the system. If the priority is to be assigned automatically by the system, the following is programmed: PRIO = -1. If several triggers call subprograms at the same time, the trigger with the highest priority is processed first, then the triggers of lower priority. 1 = highest priority.

> Further information on triggers, on offsetting the switching point and on the offset limits can be found here:
> (>>> 11.11 "Path-related switching actions (=Trigger)" Page 352)

10.4 Modifying a motion instruction

Modifying an individual motion instruction:

Precondition

- Program is selected or open.
- Operating mode T1 or T2

Procedure

1. Position the cursor in the line containing the instruction that is to be changed.
2. Press **Change**. The inline form for this instruction is opened.
3. Change the parameters.
4. Save changes by pressing **Cmd Ok**.

Modifying several motion instructions at once:

Only consecutive instructions can be changed at once.

Precondition

- Program is selected or open.
- T1 or T2 operating mode
- “Expert” user group

Procedure

1. Position the cursor in the uppernost instruction.
2. Press and hold down SHIFT and press the DOWN ARROW key repeatedly until all the instructions are selected.
3. Press the **Change** softkey. The inline form for the uppermost instruction is opened.
4. Change the parameters.

> The changed parameters are transferred to all the selected instructions. This is independent of whether they originally had the same settings as in the uppermost instruction or not.

5. Save changes by pressing **Cmd Ok**.

10.5 Modifying the coordinates of a taught point

Description The coordinates of a taught point can be modified. This is done by moving to the new position and overwriting the old point with the new position.

Precondition

- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Move the TCP to the desired position.
2. Position the cursor in the line containing the motion instruction that is to be changed.
3. Press the **Change** softkey. The inline form for this instruction is opened.
4. For PTP and LIN motions: Press the **Touch Up** softkey to accept the current position of the TCP as the new end point.

   For CIRC motions:

   - Press the **Teach Aux** softkey to accept the current position of the TCP as the new auxiliary point.
   - Press the **Teach End** softkey to accept the current position of the TCP as the new end point.
5. Confirm the request for confirmation with **Yes**.
6. Save change by pressing the **Cmd Ok** softkey.

10.6 Programming logic instructions

10.6.1 Inputs/outputs

Digital inputs/outputs

The robot controller can manage up to 4096 digital inputs and 4096 digital outputs. The configuration is customer-specific.

Analog inputs/outputs

The robot controller can manage 32 analog inputs and 32 analog outputs. The configuration is customer-specific.

Permissible range of values for inputs/outputs: -1.0 to +1.0. This corresponds to a voltage range from -10 V to +10 V. If the value is exceeded, the input/output takes the maximum value and a message is displayed until the value is back in the permissible range.

The inputs/outputs are managed via the following system variables:

	Inputs	Outputs
Digital	$IN[1] … $IN[4096]	$OUT[1] … $OUT[4096]
Analog	$ANIN[1] … $ANIN[32]	$ANOUT[1] … $ANOUT[32]

10.6.2 Setting a digital output - OUT

Precondition

- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Position the cursor in the line after which the logic instruction is to be inserted.
2. Select the menu sequence **Commands** > **Logic** > **OUT** > **OUT**.
3. Set the parameters in the inline form.

   (>>> 10.6.3 "Inline form “OUT”" Page 294)

4. Save the instruction by pressing the **Cmd Ok** softkey.

10.6.3 Inline form “OUT”

The instruction sets a digital output.

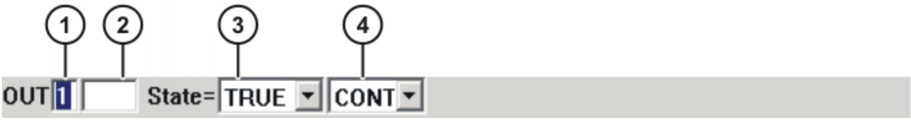


Fig. 10-21: Inline form "OUT"

Item	Description
1	Output number - **1 … 4096**
2	If a name exists for the output, this name is displayed. Only for the user group “Expert”: A name can be entered by pressing the **Longtext** softkey. The name is freely selectable.
3	State to which the output is switched - **TRUE** - **FALSE**
4	- **CONT**: Execution in the advance run - **[Empty box]**: Execution with advance run stop

10.6.4 Setting a pulse output - PULSE

Precondition

- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Position the cursor in the line after which the logic instruction is to be inserted.
2. Select the menu sequence **Commands** > **Logic** > **OUT** > **PULSE**.
3. Set the parameters in the inline form.
   (>>> 10.6.5 "Inline form “PULSE”" Page 294)
4. Save the instruction by pressing the **Cmd Ok** softkey.

10.6.5 Inline form “PULSE”

The instruction sets a pulse of a defined length.

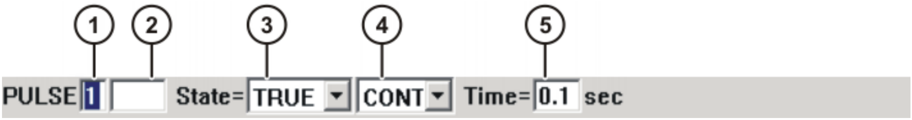


Fig. 10-22: Inline form "PULSE"

Item	Description
1	Output number ■ **1 … 4096**
2	If a name exists for the output, this name is displayed. Only for the user group “Expert”: A name can be entered by pressing the **Longtext** softkey. The name is freely selectable.
3	State to which the output is switched ■ **TRUE**: “High” level ■ **FALSE**: “Low” level
4	■ **CONT**: Execution in the advance run ■ **[Empty box]**: Execution with advance run stop
5	Length of the pulse ■ **0.1 … 3 s**

10.6.6 Setting an analog output - ANOUT

Precondition

- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Position the cursor in the line after which the instruction is to be inserted.
2. Select the menu sequence **Commands** > **Analog output** > **Static** or **Dynamic**.
3. Set the parameters in the inline form.
   (>>> 10.6.7 "Inline form “ANOUT” (static)" Page 295)
   (>>> 10.6.8 "Inline form “ANOUT” (dynamic)" Page 296)
4. Save the instruction by pressing the **Cmd Ok** softkey.

10.6.7 Inline form “ANOUT” (static)

This instruction sets a static analog output.

A maximum of 8 analog outputs (static and dynamic together) can be used at any one time. ANOUT triggers an advance run stop.

The voltage is set to a fixed level by means of a factor. The actual voltage level depends on the analog module used. For example, a 10 V module with a factor of 0.5 provides a voltage of 5 V.

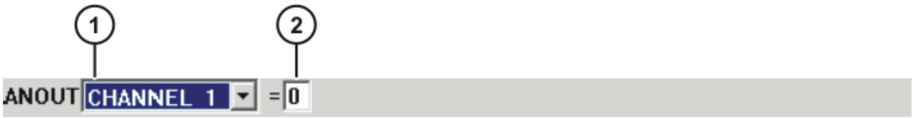


Fig. 10-23: Inline form "ANOUT" (static)

Item	Description
1	Analog output number ■ **CHANNEL_1 … CHANNEL_32**
2	Factor for the voltage ■ **0 … 1** (intervals: 0.01)

10.6.8 Inline form “ANOUT” (dynamic)

This instruction activates or deactivates a dynamic analog output.

A maximum of 4 dynamic analog outputs can be activated at any one time. ANOUT triggers an advance run stop.

The voltage is determined by a factor. The actual voltage level depends on the following values:

- Velocity or function generator

  For example, a velocity of 1 m/s with a factor of 0.5 results in a voltage of 5 V.
- Offset

  For example, an offset of +0.15 for a voltage of 0.5 V results in a voltage of 6.5 V.

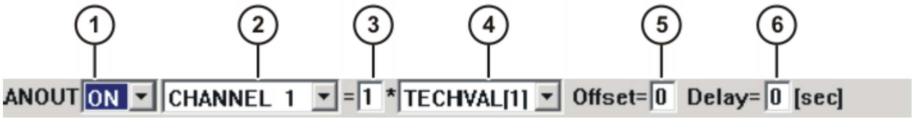


Fig. 10-24: Inline form "ANOUT" (dynamic)

Item	Description
1	Activation or deactivation of the analog output ■ **ON** ■ **OFF**
2	Analog output number ■ **CHANNEL_1 … CHANNEL_32**
3	Factor for the voltage ■ **0 … 10** (intervals: 0.01)
4	■ **VEL_ACT**: The voltage is dependent on the velocity. ■ **TECHVAL[1] … TECHVAL[6]**: The voltage is controlled by a function generator.

Item	Description
5	Value by which the voltage is increased or decreased ■ **-1 … +1** (intervals: 0.01)
6	Time by which the output signal is delayed (+) or brought forward (-) ■ **-0.2 … +0.5 s**

10.6.9 Programming a wait time - WAIT

Precondition

- Program is selected.
- Operating mode T1 or T2.

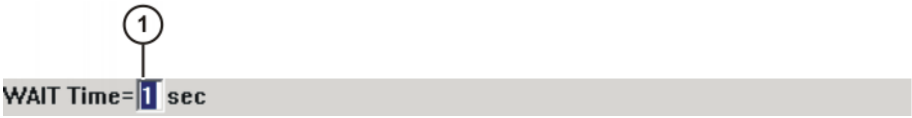
Procedure

1. Position the cursor in the line after which the logic instruction is to be inserted.
2. Select the menu sequence **Commands** > **Logic** > **WAIT**.
3. Set the parameters in the inline form.

   (>>> 10.6.10 "Inline form “WAIT”" Page 297)
4. Save the instruction by pressing the **Cmd Ok** softkey.

10.6.10 Inline form “WAIT”

WAIT can be used to program a wait time. The robot motion is stopped for a programmed time. WAIT always triggers an advance run stop.



WAIT Time=1 sec

Fig. 10-25: Inline form "WAIT"

Item	Description
1	Wait time ■ **≥ 0** s

10.6.11 Programming a signal-dependent wait function - WAITFOR

Precondition

- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Position the cursor in the line after which the logic instruction is to be inserted.
2. Select the menu sequence **Commands** > **Logic** > **WAITFOR**.
3. Set the parameters in the inline form.

   (>>> 10.6.12 "Inline form “WAITFOR”" Page 297)
4. Save the instruction by pressing the **Cmd Ok** softkey.

10.6.12 Inline form “WAITFOR”

The instruction sets a signal-dependent wait function.

If required, several signals (maximum 12) can be linked. If a logic operation is added, boxes are displayed in the inline form for the additional signals and links.

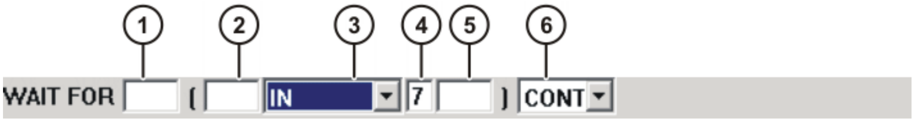


Fig. 10-26: Inline form "WAITFOR"

Item	Description
1	Add external logic operation. The operator is situated between the bracketed expressions. ■ **AND** ■ **OR** ■ **EXOR** Add NOT. ■ **NOT** ■ **[Empty box]** Enter the desired operator by means of the softkey.
2	Add internal logic operation. The operator is situated inside a bracketed expression. ■ **AND** ■ **OR** ■ **EXOR** Add NOT. ■ **NOT** ■ **[Empty box]** Enter the desired operator by means of the softkey.

Item	Description
3	Signal for which the system is waiting - **IN** - **OUT** - **CYCFLAG** - **TIMER** - **FLAG**
4	Signal number - **1 … 4096**
5	If a name exists for the signal, this name is displayed. Only for the user group “Expert”: A name can be entered by pressing the **Longtext** softkey. The name is freely selectable.
6	- **CONT**: Execution in the advance run - **[Empty box]**: Execution with advance run stop

10.6.13 Switching on the path - SYN OUT

Precondition

- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Position the cursor in the line after which the logic instruction is to be inserted.
2. Select the menu sequence **Commands** > **Logic** > **OUT** > **SYN OUT**.
3. Set the parameters in the inline form.

   (>>> 10.6.14 "Inline form “SYN OUT”, option “START/END”" Page 299)

   (>>> 10.6.15 "Inline form “SYN OUT”, option “PATH”" Page 302)
4. Save the instruction by pressing the **Cmd Ok** softkey.

10.6.14 Inline form “SYN OUT”, option “START/END”

A switching action can be triggered relative to the start or end point of a motion block. The switching action can be delayed or brought forward. The motion block can be a LIN, CIRC or PTP motion.

Possible applications include:

- Closing or opening the weld gun during spot welding
- Switching the welding current on/off during arc welding

- Starting or stopping the flow of adhesive in bonding or sealing applications.

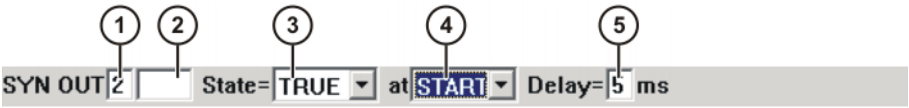


Fig. 10-27: Inline form SYN OUT, option START/END

Item	Description
1	Output number - **1 … 4096**
2	If a name exists for the output, this name is displayed. Only for the user group “Expert”: A name can be entered by pressing the **Longtext** softkey. The name is freely selectable.
3	State to which the output is switched - **TRUE** - **FALSE**
4	Point at which switching is carried out - **START**: Switching is carried out at the start point of the motion block. - **END**: Switching is carried out at the end point of the motion block. - **PATH**: (>>> 10.6.15 "Inline form “SYN OUT”, option “PATH”" Page 302)
5	Switching action delay - **-1000 … +1000 ms** **Note:** The time specification is absolute. The switching point varies according to the velocity of the robot.

Example 1

Start point and end point are exact positioning points.

```
LIN P1 VEL=0.3m/s CPDAT1
LIN P2 VEL=0.3m/s CPDAT2
SYN OUT 1 '' State= TRUE at START Delay=20ms
SYN OUT 2 '' State= TRUE at END Delay=-20ms
LIN P3 VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

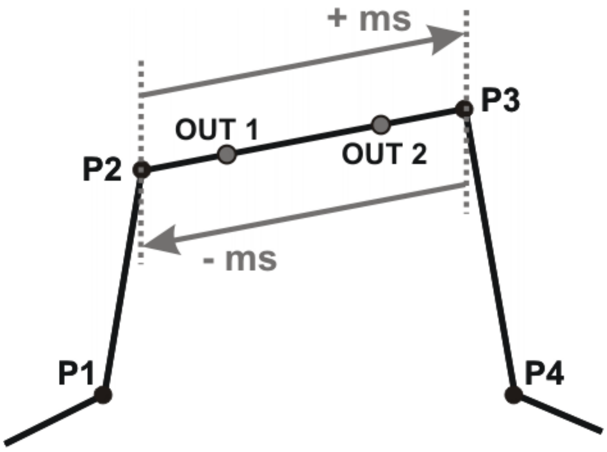


Fig. 10-28

OUT 1 and OUT 2 specify approximate positions at which switching is to occur. The dotted lines indicate the switching limits.

Switching limits:

- START: The switching point can be delayed, at most, as far as exact positioning point P3 (+ ms).
- END: The switching point can be brought forward, at most, as far as exact positioning point P2 (- ms).

If greater values are specified for the delay, the controller automatically switches at the switching limit.

Example 2

Start point is exact positioning point, end point is approximated.

```
LIN P1 VEL=0.3m/s CPDAT1
LIN P2 VEL=0.3m/s CPDAT2
SYN OUT 1 '' State= TRUE at START Delay=20ms
SYN OUT 2 '' State= TRUE at END Delay=-20ms
LIN P3 CONT VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

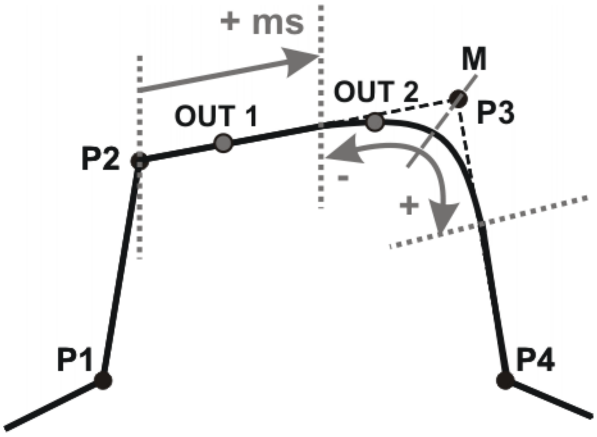


Fig. 10-29

OUT 1 and OUT 2 specify approximate positions at which switching is to occur. The dotted lines indicate the switching limits. M = middle of the approximate positioning range.

Switching limits:

- START: The switching point can be delayed, at most, as far as the start of the approximate positioning range of P3 (+ ms).

- END: The switching point can be brought forward, at most, as far as the start of the approximate positioning range of P3 (-).

  The switching point can be delayed, at most, as far as the end of the approximate positioning range of P3 (+).

If greater values are specified for the delay, the controller automatically switches at the switching limit.

Example 3

Start point and end point are approximated

```
LIN P1 VEL=0.3m/s CPDAT1
LIN P2 CONT VEL=0.3m/s CPDAT2
SYN OUT 1 '' State= TRUE at START Delay=20ms
SYN OUT 2 '' State= TRUE at END Delay=-20ms
LIN P3 CONT VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

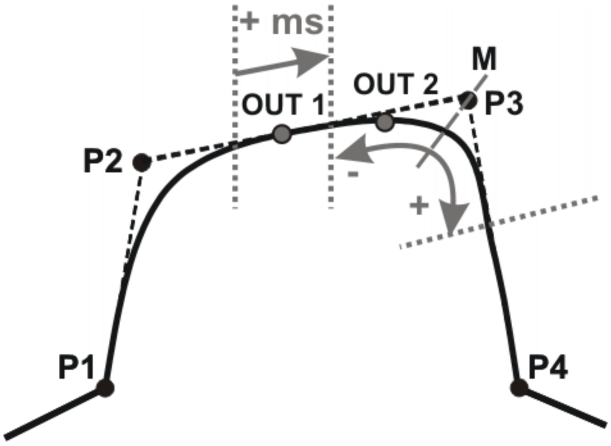


Fig. 10-30

OUT 1 and OUT 2 specify approximate positions at which switching is to occur. The dotted lines indicate the switching limits. M = middle of the approximate positioning range.

Switching limits:

- START: The switching point can be situated, at the earliest, at the end of the approximate positioning range of P2.

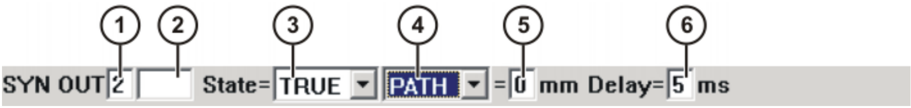
  The switching point can be delayed, at most, as far as the start of the approximate positioning range of P3 (+ ms).
- END: The switching point can be brought forward, at most, as far as the start of the approximate positioning range of P3 (-).

  The switching point can be delayed, at most, as far as the end of the approximate positioning range of P3 (+).

If greater values are specified for the delay, the controller automatically switches at the switching limit.

10.6.15 Inline form “SYN OUT”, option “PATH”

A switching action can be triggered relative to the end point of a motion block. The switching action can be shifted in space and delayed or brought forward. The motion block can be a LIN or CIRC motion. It must not be a PTP motion.



SYN OUT 2 State= TRUE PATH = 0 mm Delay= 5 ms

Fig. 10-31: Inline form SYN OUT, option PATH

Item	Description
1	Output number ■ **1 … 4096**
2	If a name exists for the output, this name is displayed. Only for the user group “Expert”: A name can be entered by pressing the **Longtext** softkey. The name is freely selectable.
3	State to which the output is switched ■ **TRUE** ■ **FALSE**
4	Point at which switching is carried out ■ **PATH**: Switching is carried out at the end point of the motion block. ■ **START**: (>>> 10.6.14 "Inline form “SYN OUT”, option “START/END”" Page 299) ■ **END**: (>>> 10.6.14 "Inline form “SYN OUT”, option “START/END”" Page 299)
5	Distance from the switching point to the end point ■ **-2000 … +2000 mm** This box is only displayed if **PATH** has been selected.
6	Switching action delay ■ **-1000 … +1000 ms** **Note:** The time specification is absolute. The switching point varies according to the velocity of the robot.

Example 1

Start point is exact positioning point, end point is approximated.

```
LIN P1 VEL=0.3m/s CPDAT1
SYN OUT 1 '' State= TRUE at START PATH=20mm Delay=-5ms
LIN P2 CONT VEL=0.3m/s CPDAT2
LIN P3 CONT VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

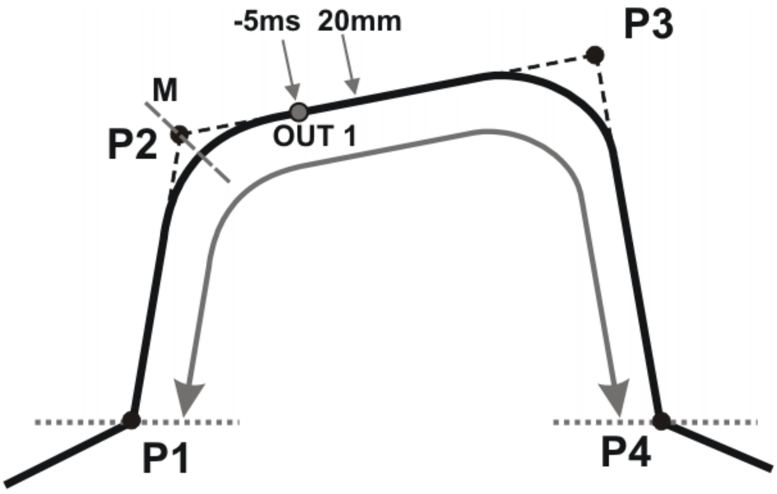


Fig. 10-32

OUT 1 specifies the approximate position at which switching is to occur. The dotted lines indicate the switching limits. M = middle of the approximate positioning range.

Switching limits:

- The switching point can be brought forward, at most, as far as exact positioning point P1.
- The switching point can be delayed, at most, as far as the next exact positioning point P4. If P3 was an exact positioning point, the switching point could be delayed, at most, as far as P3.

If greater values are specified for the shift in space or time, the controller automatically switches at the switching limit.

Example 2

Start point and end point are approximated

```
LIN P1 CONT VEL=0.3m/s CPDAT1
SYN OUT 1 '' State= TRUE at START PATH=20mm Delay=-5ms
LIN P2 CONT VEL=0.3m/s CPDAT2
LIN P3 CONT VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

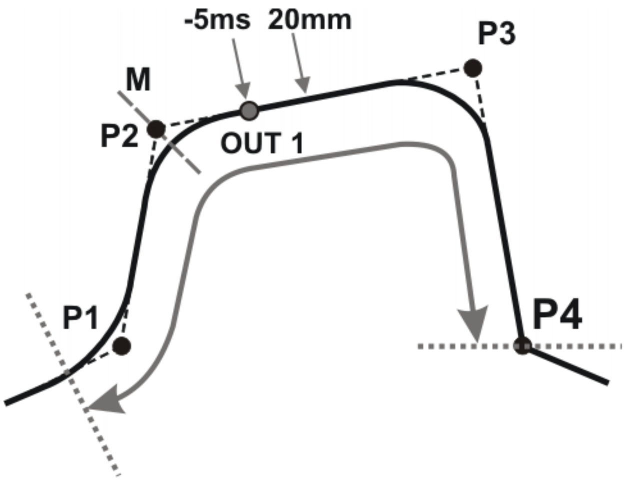


Fig. 10-33

OUT 1 specifies the approximate position at which switching is to occur. The dotted lines indicate the switching limits. M = middle of the approximate positioning range.

Switching limits:

- The switching point can be brought forward, at most, as far as the start of the approximate positioning range of P1.
- The switching point can be delayed, at most, as far as the next exact positioning point P4. If P3 was an exact positioning point, the switching point could be delayed, at most, as far as P3.

If greater values are specified for the shift in space or time, the controller automatically switches at the switching limit.

10.6.16 Setting a pulse on the path - SYN PULSE

Precondition

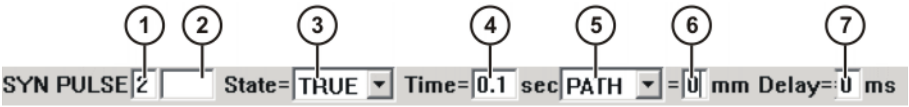
- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Position the cursor in the line after which the logic instruction is to be inserted.
2. Select the menu sequence **Commands** > **Logic** > **OUT** > **SYN PULSE**.
3. Set the parameters in the inline form.
   (>>> 10.6.17 "Inline form “SYN PULSE”" Page 305)
4. Save the instruction by pressing the **Cmd Ok** softkey.

10.6.17 Inline form “SYN PULSE”

A pulse can be triggered relative to the start or end point of a motion block. The pulse can be delayed or brought forward and shifted in space.



SYN PULSE 2 State=TRUE Time=0.1 sec PATH =0 mm Delay=0 ms

Fig. 10-34: Inline form "SYN PULSE"

Item	Description
1	Output number - **1 … 4096**
2	If a name exists for the output, this name is displayed. Only for the user group “Expert”: A name can be entered by pressing the **Longtext** softkey. The name is freely selectable.
3	State to which the output is switched - **TRUE** - **FALSE**
4	Duration of the pulse - **0.1 … 3 s**

Item	Description
5	- **START**: The pulse is triggered at the start point of the motion block. - **END**: The pulse is triggered at the end point of the motion block. See SYN OUT for examples and switching limits. (>>> 10.6.14 "Inline form “SYN OUT”, option “START/END”" Page 299) - **PATH**: The pulse is triggered at the end point of the motion block. See SYN OUT for examples and switching limits. (>>> 10.6.15 "Inline form “SYN OUT”, option “PATH”" Page 302)
6	Distance from the switching point to the end point - **-2000 … +2000 mm** This box is only displayed if **PATH** has been selected.
7	Switching action delay - **-1000 … +1000 ms** **Note:** The time specification is absolute. The switching point varies according to the velocity of the robot.

10.6.18 Modifying a logic instruction

Precondition

- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Position the cursor in the line containing the instruction that is to be changed.
2. Press **Change**. The inline form for this instruction is opened.
3. Change the parameters.
4. Save changes by pressing **Cmd Ok**.

11 Programming for user group “Expert” (KRL syntax)

If a selected program is edited in the user group “Expert”, the cursor must then be removed from the edited line and positioned in any other line!
Only in this way is it certain that the editing will be applied when the program is deselected again.

11.1 Overview of KRL syntax

Variables and declarations	
DECL	(>>> 11.4.1 "DECL" Page 313)
ENUM	(>>> 11.4.2 "ENUM" Page 314)
IMPORT … IS	(>>> 11.4.3 "IMPORT ... IS" Page 315)
STRUC	(>>> 11.4.4 "STRUC" Page 316)

Motion programming	
CIRC	(>>> 11.5.5 "CIRC" Page 322)
CIRC_REL	(>>> 11.5.6 "CIRC_REL" Page 323)
LIN	(>>> 11.5.3 "LIN" Page 319)
LIN_REL	(>>> 11.5.4 "LIN_REL" Page 320)
PTP	(>>> 11.5.1 "PTP" Page 317)
PTP_REL	(>>> 11.5.2 "PTP_REL" Page 318)
SPLINE … ENDSPLINE	(>>> 11.6.1 "SPLINE ... ENDSPLINE" Page 324)
SLIN	(>>> 11.6.2 "SLIN" Page 325)
SCIRC	(>>> 11.6.3 "SCIRC" Page 326)
SPL	(>>> 11.6.4 "SPL" Page 328)
TIME_BLOCK	(>>> 11.6.5 "TIME_BLOCK" Page 328)

Program execution control	
CONTINUE	(>>> 11.7.1 "CONTINUE" Page 332)
EXIT	(>>> 11.7.2 "EXIT" Page 333)
FOR … TO … ENDFOR	(>>> 11.7.3 "FOR ... TO ... ENDFOR" Page 333)
GOTO	(>>> 11.7.4 "GOTO" Page 334)
HALT	(>>> 11.7.5 "HALT" Page 335)
IF … THEN … ENDIF	(>>> 11.7.6 "IF ... THEN ... ENDIF" Page 335)
LOOP … ENDLOOP	(>>> 11.7.7 "LOOP ... ENDLOOP" Page 336)
REPEAT … UNTIL	(>>> 11.7.8 "REPEAT ... UNTIL" Page 336)
SWITCH … CASE … ENDSWITCH	(>>> 11.7.9 "SWITCH ... CASE ... ENDSWITCH" Page 337)
WAIT … FOR	(>>> 11.7.10 "WAIT FOR" Page 338)
WAIT … SEC	(>>> 11.7.11 "WAIT SEC" Page 338)
WHILE … ENDWHILE	(>>> 11.7.12 "WHILE ... ENDWHILE" Page 339)

Inputs/outputs	
ANIN	(>>> 11.8.1 "ANIN" Page 339)
ANOUT	(>>> 11.8.2 "ANOUT" Page 340)
PULSE	(>>> 11.8.3 "PULSE" Page 342)
SIGNAL	(>>> 11.8.4 "SIGNAL" Page 345)

Subprograms and functions	
RETURN	(>>> 11.9.1 "RETURN" Page 346)

Interrupt programming	
BRAKE	(>>> 11.10.1 "BRAKE" Page 347)
INTERRUPT	(>>> 11.10.3 "INTERRUPT" Page 349)
INTERRUPT … DECL … WHEN … DO	(>>> 11.10.2 "INTERRUPT ... DECL ... WHEN ... DO" Page 347)
RESUME	(>>> 11.10.4 "RESUME" Page 350)

Path-related switching actions (=Trigger)	
TRIGGER WHEN DISTANCE	(>>> 11.11.1 "TRIGGER WHEN DISTANCE" Page 352)
TRIGGER WHEN PATH	(>>> 11.11.2 "TRIGGER WHEN PATH" Page 355)
TRIGGER WHEN PATH (for spline)	(>>> 11.11.3 "TRIGGER WHEN PATH (for SPLINE)" Page 358)

Communication
(>>> 11.12 "Communication" Page 362)

System functions	
VARSTATE()	(>>> 11.13.1 "VARSTATE()" Page 362)

Manipulating string variables
(>>> 11.14 "Editing string variables" Page 364)

11.2 Symbols and fonts

The following symbols and fonts are used in the syntax descriptions:

Syntax element	Representation
KRL code	■ Courier font ■ Upper-case letters Examples: GLOBAL; ANIN ON; OFFSET
Elements that must be replaced by program-specific entries	■ Italics ■ Upper/lower-case letters Examples: *Distance*; *Time*; *Format*
Optional elements	■ In angle brackets Example: <STEP *Increment*>
Elements that are mutually exclusive	■ Separated by the "|" symbol Example: IN |OUT

11.3 Important KRL terms

11.3.1 SRC files and DAT files

A KRL program generally consists of an **SRC file** and a **DAT file** of the same name.

- SRC file: contains the program code.
- DAT file: contains permanent data and point coordinates. The DAT file is also called a **data list**.

The SRC file and associated DAT file together are called a **module**.

Depending on the user group, programs in the Navigator are displayed as modules or individual files:

- User group "User"

  A program is displayed as a module. The SRC file and the DAT file exist in the background. They are not visible for the user and cannot be edited individually.
- User group "Expert"

  By default, the SRC file and the DAT file are displayed individually. They can be edited individually.

11.3.2 Subprograms and functions

Subprograms

Subprograms are programs which are accessed by means of branches from the main program. Once the subprogram has been executed, the main program is resumed from the line directly after the subprogram call.

- **Local subprograms** are contained in the same SRC file as the main program. They can be made to be recognized globally using the keyword GLOBAL (>>> 11.3.5 "Areas of validity" Page 311).
- **Global subprograms** are programs with a separate SRC file of their own, which is accessed from another program by means of a branch.

Functions

Functions, like subprograms, are programs which are accessed by means of branches from the main program. In addition, however, they also have a data type and always return a value to the main program.

11.3.3 Naming conventions and keywords

Names

Examples of names in KRL: variable names, program names, point names

- Names in KRL can have a maximum length of 24 characters.

  In some cases, less than 24 characters are allowed, e.g. a maximum of 23 characters in inline forms.
- Names in KRL can consist of letters (A-Z), numbers (0-9) and the signs “_” and “$”.
- Names in KRL must not begin with a number.
- Names in KRL must not be keywords.

Other restrictions may apply in the case of inline forms in technology packages.

> The names of all system variables begin with the “$” sign. To avoid confusion, do not begin the names of user-defined variables with this sign.

Keywords

Keywords are sequences of letters having a fixed meaning. They must not be used in programs in any way other than with this meaning. No distinction is made between uppercase and lowercase letters. A keyword remains valid irrespective of the way in which it is written.

Example: The sequence of letters CASE is an integral part of the KRL syntax SWITCH … CASE … ENDSWITCH. For this reason, CASE must not be used in any other way, e.g. as a variable name.

The system distinguishes between reserved and non-reserved keywords:

- Reserved keywords

  These may only be used with their defined meaning.
- Non-reserved keywords

  With non-reserved keywords, the meaning is restricted to a particular context. Outside of this context, a non-reserved keyword is interpreted by the compiler as a name.

> In practice, it is not helpful to distinguish between reserved and non-reserved keywords. To avoid error messages or compiler problems, keywords are thus never used other than with their defined meaning.

Overview of important keywords:

All elements of the KRL syntax described in this documentation that are not program-specific are keywords.

The following important keywords are worth a particular mention:

AXIS
BOOL
CHAR
CAST_FROM
CAST_TO
CCLOSE
CHANNEL
CIOCTL
CONFIRM
CONST
COPEN
CREAD
CWRITE
DEF
DEFAULT
DEFDAT
DEFFCT
E6AXIS
E6POS
END
ENDDAT

ENDFCT
ENDFOR
ENDIF
ENDLOOP
ENDSWITCH
ENDWHILE
EXT
EXTFCT
FALSE
FRAME
GLOBAL
INT
MAXIMUM
MINIMUM
POS
PRIO
PUBLIC
SREAD
SWRITE
REAL
TRUE

11.3.4 Data types

Overview

There are 2 kinds of data types:

- User-defined data types

  User-defined data types are always derived from the data types ENUM or STRUC.
- Predefined data types, e.g.:
  - Simple data types
  - Data types for motion programming

The following simple data types are predefined:

Data type	Keyword	Description
Integer	`INT`	Integer ■ **$-2^{31}-1$ … $2^{31}-1$** Examples: 1; 32; 345
Real	`REAL`	Floating-point number ■ **+1.1E-38 … +3.4E+38** Examples: 1.43; 38.50; 300.25
Boolean	`BOOL`	Logic state ■ **TRUE** ■ **FALSE**
Character	`CHAR`	1 character ■ **ASCII character** Examples: "A"; "1"; "q"

The following data types for motion programming are predefined:

Structure type AXIS

A1 to A6 are angle values (rotational axes) or translation values (translational axes) for the axis-specific movement of robot axes 1 to 6.

```
STRUC AXIS REAL A1, A2, A3, A4, A5, A6
```

Structure type E6AXIS

E1 to E6 are angle values or translation values of the external axes 7 to 12.

```
STRUC E6AXIS REAL A1, A2, A3, A4, A5, A6, E1, E2, E3, E4, E5, E6
```

Structure type FRAME

X, Y and Z are space coordinates, while A, B and C are the orientation of the coordinate system.

```
STRUC FRAME REAL X, Y, Z, A, B, C
```

Structure types POS and E6POS

S (Status) and T (Turn) define axis positions unambiguously.

```
STRUC POS REAL X, Y, Z, A, B, C, INT S, T
```

```
STRUC E6POS REAL X, Y, Z, A, B, C, E1, E2, E3, E4, E5, E6, INT S, T
```

11.3.5 Areas of validity

Local

Data object	Area of validity
Variable	Valid in the program code between DEF and END containing the declaration of the variables.
Constant	Valid in the module to which the data list in which the constant was declared belongs.
User-defined data type	If the data type has been defined in an SRC file: valid at, or below, the program level in which it was declared. If the data type has been defined in a DAT file: valid in the SRC file that belongs to the DAT file.

Data object	Area of validity
Subprogram	Valid in the main program of the shared SRC file.
Function	Valid in the main program of the shared SRC file.
Interrupt	Valid at, or below, the programming level in which it was declared.

Global

The data objects referred to under “Local” are globally valid if they are declared using the keyword GLOBAL.

> GLOBAL can only be used for variables and user-defined data types if they have been declared in a data list.
> To use GLOBAL, the entry GLOBAL_KEY in the file PROGRESS.INI, in the INIT directory, must be set to TRUE: GLOBAL_KEY=TRUE

Variables and user-defined data types are also globally valid if they were declared in the section USER GLOBALS in $CONFIG.DAT.

If there are local and global variables with the same name, the compiler uses the local variable within its area of validity.

Always globally valid:

- The first program in an SRC file. By default, it bears the name of the SRC file.
- Predefined data types
- KRL system variables
- Variables declared in $CONFIG.DAT

Examples

The examples show where the keyword GLOBAL must be positioned.

Declaration of a global variable:

`<DECL> GLOBAL` *Data type Variable name*

Declaration of a global subprogram:

Main program
`GLOBAL DEF` *Subprogram name ()*

11.3.6 Constants

The value of a constant can no longer be modified during program execution after initialization. Constants can be used to prevent a value from being changed accidentally during program execution.

Constants must be declared and, at the same time, initialized in a data list. The data type must be preceded by the keyword CONST.

`DECL <GLOBAL> CONST` *Data_Type Variable_Name = Value*

> The keyword CONST must only be used in data lists.
> **Precondition:** To use CONST, the entry CONST_KEY in the file PROGRESS.INI, in the INIT directory, must be set to TRUE: CONST_KEY=TRUE

11.4 Variables and declarations

11.4.1 DECL

Description

Declaration of variables, arrays and constants

Syntax

Declaration of variables

Declaration of variables in programs:

`<DECL>` *Data_Type Name1 <, ..., NameN>*

Declaration of variables in data lists:

`<DECL> <GLOBAL>` *Data_Type Name1 <, ..., NameN>*

Declaration of variables in data lists with simultaneous initialization:

`<DECL> <GLOBAL>` *Data_Type Name = Value*

In the case of declaration with simultaneous initialization, a separate DECL declaration is required for each variable. It is not possible to declare and initialize several variables with a single DECL declaration.

> If a non-declared variable is used in the program, this variable is automatically assigned the default data type POS.

Declaration of arrays

Declaration of arrays in programs:

`<DECL>` *Data_Type Name1 [Dimension1 <, ..., Dimension3>] <, ..., NameN [DimensionN1 <,..., DimensionN3>] >*

Declaration of arrays in data lists:

`<DECL> <GLOBAL>` *Data_Type Name1 [Dimension1 <, ..., Dimension3>] <, ..., NameN [DimensionN1 <,..., DimensionN3>] >*

For the declaration of arrays or constant arrays in data lists with simultaneous initialization:

- It is not permissible to declare and initialize in a single line. The initialization must, however, follow directly after the line containing the declaration. There must be no lines, including blank lines, in between.
- If several elements of an array are initialized, the elements must be specified in ascending sequence of the array index (starting from the right-hand array index).
- If the same character string is to be assigned to all of the elements of an array of type CHAR as a default setting, it is not necessary to initialize each array element individually. The right-hand array index is omitted. (No index is written for a one-dimensional array index.)

Declaration of arrays in data lists with simultaneous initialization:

```
<DECL> <GLOBAL> Data_Type Name [Dimension1 <,..., Dimension3> ]
Name [1 <, 1, 1> ] = Value1
<Name [1 <, 1, 2> ] = Value2>
...
Name [Dimension1 <, Dimension2, Dimension3> ] = ValueN
```

Declaration of constant arrays in data lists with simultaneous initialization:

```
DECL <GLOBAL> CONST Data_Type Name [Dimension1 <,..., Dimension3> ]
Name [1 <, 1, 1> ] = Value1
<Name [1 <, 1, 2> ] = Value2>
...
Name [Dimension1 <, Dimension2, Dimension3> ] = ValueN
```

Explanation of the syntax

Element	Description
`DECL`	DECL can be omitted if *Data_Type* is a predefined data type. If *Data_Type* is a user-defined data type, then DECL is obligatory.
`GLOBAL`	(>>> 11.3.5 "Areas of validity" Page 311)
`CONST`	The keyword CONST must only be used in data lists. **Precondition:** To use CONST, the entry CONST_KEY in the file PROGRESS.INI, in the INIT directory, must be set to TRUE: `CONST_KEY=TRUE`
Data type	Specification of the desired data type
Name	Name of the object (variable, array or constant) that is being declared.
Dimension	Type: INT *Dimension* defines the number of array elements for the dimension in question. Arrays have a minimum of 1 and a maximum of 3 dimensions.
Value	The data type of *Value* must be compatible with *Data_Type*, but not necessarily identical. If the data types are compatible, the system automatically matches them.

Example 1

Declarations with predefined data types. The keyword DECL can also be omitted.

```
DECL INT X
DECL INT X1, X2
DECL REAL ARRAY_A[7], ARRAY_B[5], A
```

Example 2

Declarations of arrays with simultaneous initialization (only possible in data lists).

```
INT A[7]
A[1]=27
A[2]=313
A[6]=11
CHAR TEXT1[80]
TEXT1[]="message"
CHAR TEXT2[2,80]
TEXT2[1,]="first message"
TEXT2[2,]="second message"
```

11.4.2 ENUM

Description

Definition of an enumeration type (= ENUM data type)

Syntax

`<GLOBAL> ENUM` *NameEnumType Constant1*`<, ..., `*ConstantN*`>`

Explanation of the syntax

Element	Description
`GLOBAL`	(>>> 11.3.5 "Areas of validity" Page 311)
NameEnumType	Name of the new enumeration type. Recommendation: For user-defined data types, assign names ending in _TYPE, to distinguish them from variable names.
Constant	The constants are the values that a variable of the enumeration type can take. Each constant may only occur once in the definition of the enumeration type.

Example 1

Definition of an enumeration type with the name COUNTRY_TYPE.

```
ENUM COUNTRY_TYP SWITZERLAND, AUSTRIA, ITALY, FRANCE
```

Declaration of a variable of type COUNTRY_TYPE:

```
DECL COUNTRY_TYP MYCOUNTRY
```

Initialization of the variable of type COUNTRY_TYPE:

```
MYCOUNTRY = #AUSTRIA
```

Example 2

An enumeration type with the name SWITCH_TYPE and the constants ON and OFF is defined.

```
DEF PROG()
ENUM SWITCH_TYP ON, OFF
DECL SWITCH_TYP GLUE
   IF A>10 THEN
      GLUE=#ON
   ELSE
      GLUE=#OFF
   ENDIF
END
```

11.4.3 IMPORT ... IS

Description

Imports a variable or a complete array from an external data list. The data object can be imported into an SRC or DAT file.

Each data object that is to be imported requires its own IMPORT declaration.

If an imported data object is accessed in a program, this triggers an advance run stop.

Precondition for importing: The name in the DEF line in the external data list must be followed by the keyword PUBLIC.

`DEFDAT` *Data list name* `PUBLIC`

Syntax

`IMPORT` *Data type name* `IS /R1/`*Data source*`..`*Old name*

The IMPORT line is a declaration and must be situated in the declaration section.

Explanation of the syntax

Element	Description
Data type	Data type of the data object as declared in the external data list. It is not possible to assign a different data type to the data object during importing.
Name	A different name can be assigned to the data object during importing. The original name is retained in the data source. If an array is imported, *Name* must be followed by square brackets. The brackets must be empty, with the exception of commas in the case of multi-dimensional arrays: ■ One-dimensional array: *Name[]* ■ Two-dimensional array: *Name[,]* ■ Three-dimensional array: *Name[,,]*
Data source	Name of the external data list without the dot and file extension. The data objects must be imported from the data lists in which they were originally created.
Old name	Name in the external data list of the data object to be imported. Unlike with *Name*, no brackets are specified for arrays.

Data source and *Old name* are connected to one another by two periods. No blanks may appear between the two periods.

Example 1

Import of the value of the integer variable VAR from the data list DATA1. The name VAR is retained.

```
IMPORT INT VAR IS /R1/DATA1..VAR
```

Example 2

Import of the two-dimensional POS array POS_EX from the data list POSITION. The name of the array must be POS1 in the new program.

```
IMPORT POS POS1[,] IS /R1/POSITION..POS_EX
```

11.4.4 STRUC

Description

Definition of a structure type (= STRUC data type). Several data types are combined to form a new data type.

Syntax

`<GLOBAL> STRUC` *Name structure type Data type1 Component1A*<, *Component1B*, ...> < , *Data type2 Component2A*<, *Component2B*, ...>>

Explanation of the syntax

Element	Description
`GLOBAL`	(>>> 11.3.5 "Areas of validity" Page 311)
Name structure type	Name of the new structure type. The names of user-defined data types should end in _TYPE, to distinguish them from variable names.
Data type	TYPE: Any data type Structure types are also permissible as data types.
Component	Name of the component. It may only be used once in the structure type. Arrays can only be used as components of a structure type if they have the type CHAR and are one-dimensional. In this case, the array limit follows the name of the array in square brackets in the definition of the structure type.

Value assignment

There are 2 ways of assigning values to variables based on a STRUC data type:

- Assignment of values to several components of a variable: with an **aggregate**
- Assignment of a value to a single component of a variable: with the **point separator**

Information regarding the aggregate:

- The values of an aggregate can be simple constants or themselves aggregates; they may not, however, be variables (see also Example 3).
- Not all components of the structure have to be specified in an aggregate.
- The components do not need to be specified in the order in which they have been defined.
- Each component may only be contained once in an aggregate.
- The name of the structure type can be specified at the beginning of an aggregate, separated by a colon.

Example 1

Definition of a structure type CAR_TYPE with the components AIR_COND, YEAR and PRICE.

```
STRUC CAR_TYP BOOL AIR_COND, INT YEAR, REAL PRICE
```

Declaration of a variable of type CAR_TYPE:

```
DECL CAR_TYP MYCAR
```

Initialization of the variable MYCAR of type CAR_TYPE with an **aggregate**:

A variable based on a structure type does not have to be initialized with an aggregate. It is also possible to initialize the components individually with the point separator.

```
MYCAR = {CAR_TYP: PRICE 15000, AIR_COND TRUE, YEAR 2003}
```

Modification of an individual component using the **point separator**:

```
MYCAR.AIR_COND = FALSE
```

Example 2

Definition of a structure type S_TYPE with the component NUMBER of data type REAL and of the array component TEXT[80] of data type CHAR.

```
STRUC S_TYP REAL NUMBER, CHAR TEXT[80]
```

Example 3

Example of aggregates as values of an aggregate:

```
STRUC INNER_TYP INT A, B, C
STRUC OUTER_TYP INNER_TYP Q, R
DECL OUTER_TYP MYVAR
...
MYVAR = {Q {A 1, B 4}, R {A 3, C 2}}
```

11.5 Motion programming: PTP, LIN, CIRC

11.5.1 PTP

Description

Executes a point-to-point motion to the end point. The coordinates of the end point are absolute.

Syntax

`PTP` *End point* `<C_PTP` <*CP approximation*>>

Explanation of the syntax

Element	Description
End point	Type: POS, E6POS, AXIS, E6AXIS, FRAME The end point can be specified in Cartesian or axis-specific coordinates. Cartesian coordinates refer to the BASE coordinate system. If not all components of the end point are specified, the controller takes the values of the previous position for the missing components.

Element	Description
`C_PTP`	Causes the end point to be approximated. The specification C_PTP is sufficient for PTP-PTP approximate positioning. In the case of PTP-CP approximation, i.e. if the approximated PTP block is followed by a LIN or CIRC block, *CP approximation* must also be specified.
CP approximation	Only for PTP-CP approximate positioning. This parameter defines the earliest point at which the approximate positioning can begin. The possible specifications are: ■ `C_DIS` Distance parameter (default): Approximation starts, at the earliest, when the distance to the end point falls below the value of $APO.CDIS. ■ `C_ORI` Orientation parameter: Approximation starts, at the earliest, when the dominant orientation angle falls below the value of $APO.CORI. ■ `C_VEL` Velocity parameter: Approximation starts, at the earliest, when the velocity in the deceleration phase to the end point falls below the value of $APO.CVEL.

Example 1

End point specified in Cartesian coordinates.

```
PTP {X 12.3,Y 100.0,Z 50,A 9.2,B 50,C 0,S ’B010’,T ’B1010’}
```

Example 2

End point specified in axis-specific coordinates. The end point is approximated.

```
PTP {A1 10,A2 -80.6,A3 -50,A4 0,A5 14.2, A6 0} C_PTP
```

Example 3

End point specified with only 2 components. For the rest of the components, the controller takes the values of the previous position.

```
PTP {Z 500,X 123.6}
```

11.5.2 PTP_REL

Description

Executes a point-to-point motion to the end point. The coordinates of the end point are relative to the current position.

> A REL statement always refers to the current position of the robot. For this reason, if a REL motion is interrupted, the robot executes the entire REL motion again, starting from the position at which it was interrupted.

Syntax

`PTP_REL` *End point* `<C_PTP` *<CP approximation*>>

Explanation of the syntax

Element	Description
End point	Type: POS, E6POS, AXIS, E6AXIS The end point can be specified in Cartesian or axis-specific coordinates. The controller interprets the coordinates as relative to the current position. Cartesian coordinates refer to the BASE coordinate system. If not all components of the end point are specified, the controller sets the value of the missing components to 0. In other words, the absolute values of these components remain unchanged.
`C_PTP`	Causes the end point to be approximated. The specification C_PTP is sufficient for PTP-PTP approximate positioning. In the case of PTP-CP approximation, i.e. if the approximated PTP block is followed by a LIN or CIRC block, *CP approximation* must also be specified.
CP approximation	Only for PTP-CP approximate positioning. This parameter defines the earliest point at which the approximate positioning can begin. The possible specifications are: ■ `C_DIS` Distance parameter (default): Approximation starts, at the earliest, when the distance to the end point falls below the value of $APO.CDIS. ■ `C_ORI` Orientation parameter: Approximation starts, at the earliest, when the dominant orientation angle falls below the value of $APO.CORI. ■ `C_VEL` Velocity parameter: Approximation starts, at the earliest, when the velocity in the deceleration phase to the end point falls below the value of $APO.CVEL.

Example 1

Axis 2 is moved 30 degrees in a negative direction. None of the other axes moves.

```
PTP_REL {A2 -30}
```

Example 2

The robot moves 100 mm in the X direction and 200 mm in the negative Z direction from the current position. Y, A, B, C and S remain constant. T is calculated in relation to the shortest path.

```
PTP_REL {X 100,Z -200}
```

11.5.3 LIN

Description

Executes a linear motion to the end point. The coordinates of the end point are absolute.

Syntax

`LIN` *End point* <*CP approximation*>

Explanation of the syntax

Element	Description
End point	Type: POS, E6POS, FRAME If not all components of the end point are specified, the controller takes the values of the previous position for the missing components. The Status and Turn specifications for an end point of type POS or E6POS are ignored in the case of LIN (and CIRC) motions. The coordinates refer to the BASE coordinate system.
CP approximation	This parameter causes the end point to be approximated. It also defines the earliest point at which the approximate positioning can begin. The possible specifications are: ■ `C_DIS` Distance parameter: Approximation starts, at the earliest, when the distance to the end point falls below the value of $APO.CDIS. ■ `C_ORI` Orientation parameter: Approximation starts, at the earliest, when the dominant orientation angle falls below the value of $APO.CORI. ■ `C_VEL` Velocity parameter: Approximation starts, at the earliest, when the velocity in the deceleration phase to the end point falls below the value of $APO.CVEL.

Example

End point with two components. For the rest of the components, the controller takes the values of the previous position.

```
LIN {Z 500,X 123.6}
```

11.5.4 LIN_REL

Description

Executes a linear motion to the end point. The coordinates of the end point are relative to the current position.

> A REL statement always refers to the current position of the robot. For this reason, if a REL motion is interrupted, the robot executes the entire REL motion again, starting from the position at which it was interrupted.

Syntax

`LIN_REL` *End point <CP approximation>* `<#BASE|#TOOL>`

Explanation of the syntax

Element	Description
End point	Type: POS, E6POS, FRAME The end point must be specified in Cartesian coordinates. The controller interprets the coordinates as relative to the current position. The coordinates can refer to the BASE or TOOL coordinate system. If not all components of the end point are specified, the controller automatically sets the value of the missing components to 0. In other words, the absolute values of these components remain unchanged. The Status and Turn specifications for an end point of type POS or E6POS are disregarded in the case of LIN motions.
CP approximation	This parameter causes the end point to be approximated. It also defines the earliest point at which the approximate positioning can begin. The possible specifications are: ■ `C_DIS` Distance parameter: Approximation starts, at the earliest, when the distance to the end point falls below the value of $APO.CDIS. ■ `C_ORI` Orientation parameter: Approximation starts, at the earliest, when the dominant orientation angle falls below the value of $APO.CORI. ■ `C_VEL` Velocity parameter: Approximation starts, at the earliest, when the velocity in the deceleration phase to the end point falls below the value of $APO.CVEL.
`#BASE`, `#TOOL`	■ #BASE Default setting. The coordinates of the end point refer to the BASE coordinate system. ■ #TOOL The coordinates of the end point refer to the TOOL coordinate system. The specification of #BASE or #TOOL refers only to the corresponding LIN_REL statement. It has no effect on subsequent statements.

Example 1

The TCP moves 100 mm in the X direction and 200 mm in the negative Z direction from the current position in the BASE coordinate system. Y, A, B, C and S remain constant. T is determined by the motion.

```
LIN_REL {X 100,Z -200}
```

Example 2

The TCP moves 100 mm from the current position in the negative X direction in the TOOL coordinate system. Y, Z, A, B, C and S remain constant. T is determined by the motion.

This example is suitable for moving the tool backwards against the tool direction. The precondition is that the tool direction has been calibrated along the X axis.

```
LIN_REL {X -100} #TOOL
```

11.5.5 CIRC

Description

Executes a circular motion. An auxiliary point and an end point must be specified in order for the controller to be able to calculate the circular motion.

The coordinates of the auxiliary point and end point are absolute.

Syntax

`CIRC` *Auxiliary point, End point*<`,` `CA` *Circular angle*> <*CP approximation*>

Explanation of the syntax

Element	Description
Auxiliary point	Type: POS, E6POS, FRAME If not all components of the auxiliary point are specified, the controller takes the values of the previous position for the missing components. The orientation angles and the Status and Turn specifications for an auxiliary point are always disregarded. The auxiliary point cannot be approximated. The motion always stops exactly at this point. The coordinates refer to the BASE coordinate system.
End point	Type: POS, E6POS, FRAME If not all components of the end point are specified, the controller takes the values of the previous position for the missing components. The Status and Turn specifications for an end point of type POS or E6POS are ignored in the case of CIRC (and LIN) motions. The coordinates refer to the BASE coordinate system.
Circular angle	Specifies the overall angle of the circular motion. This makes it possible to extend the motion beyond the programmed end point or to shorten it. The actual end point thus no longer corresponds to the programmed end point. Unit: degrees. There is no limit; in particular, a circular angle greater than 360° can be programmed. ■ Positive circular angle: the circular path is executed in the direction Start point › Auxiliary point › End point. ■ Negative circular angle: the circular path is executed in the direction Start point › End point › Auxiliary point.
CP approximation	This parameter causes the end point to be approximated. It also defines the earliest point at which the approximate positioning can begin. The possible specifications are: ■ `C_DIS` Distance parameter: Approximation starts, at the earliest, when the distance to the end point falls below the value of $APO.CDIS. ■ `C_ORI` Orientation parameter: Approximation starts, at the earliest, when the dominant orientation angle falls below the value of $APO.CORI. ■ `C_VEL` Velocity parameter: Approximation starts, at the earliest, when the velocity in the deceleration phase to the end point falls below the value of $APO.CVEL.

Example

The end point of the circular motion is defined by a circular angle of 260°. The end point is approximated.

```
CIRC {X 5,Y 0, Z 9.2},{X 12.3,Y 0,Z -5.3,A 9.2,B -5,C 20}, CA 260
C_ORI
```

11.5.6 CIRC_REL

Description

Executes a circular motion. An auxiliary point and an end point must be specified in order for the controller to be able to calculate the circular motion.

The coordinates of the auxiliary point and end point are relative to the current position.

> A REL statement always refers to the current position of the robot. For this reason, if a REL motion is interrupted, the robot executes the entire REL motion again, starting from the position at which it was interrupted.

Syntax

`CIRC_REL` *Auxiliary point, End point*<`, CA` *Circular angle*> <*CP approximation*>

Explanation of the syntax

Element	Description
Auxiliary point	Type: POS, E6POS, FRAME The auxiliary point must be specified in Cartesian coordinates. The controller interprets the coordinates as relative to the current position. The coordinates refer to the BASE coordinate system. If $ORI_TYPE, Status and/or Turn are specified, these specifications are ignored. If not all components of the auxiliary point are specified, the controller sets the value of the missing components to 0. In other words, the absolute values of these components remain unchanged. The orientation angles and the Status and Turn specifications for an auxiliary point are disregarded. The auxiliary point cannot be approximated. The motion always stops exactly at this point.
End point	Type: POS, E6POS, FRAME The end point must be specified in Cartesian coordinates. The controller interprets the coordinates as relative to the current position. The coordinates refer to the BASE coordinate system. If not all components of the end point are specified, the controller sets the value of the missing components to 0. In other words, the absolute values of these components remain unchanged. The Status and Turn specifications for an end point of type POS or E6POS are disregarded.

Element	Description
Circular angle	Specifies the overall angle of the circular motion. This makes it possible to extend the motion beyond the programmed end point or to shorten it. The actual end point thus no longer corresponds to the programmed end point. Unit: degrees. There is no limit; in particular, a circular angle > 360° can be programmed. ■ Positive circular angle: the circular path is executed in the direction Start point › Auxiliary point › End point. ■ Negative circular angle: the circular path is executed in the direction Start point › End point › Auxiliary point.
CP approximation	This parameter causes the end point to be approximated. It also defines the earliest point at which the approximate positioning can begin. The possible specifications are: ■ `C_DIS` Distance parameter: Approximation starts, at the earliest, when the distance to the end point falls below the value of $APO.CDIS. ■ `C_ORI` Orientation parameter: Approximation starts, at the earliest, when the dominant orientation angle falls below the value of $APO.CORI. ■ `C_VEL` Velocity parameter: Approximation starts, at the earliest, when the velocity in the deceleration phase to the end point falls below the value of $APO.CVEL.

Example

The end point of the circular motion is defined by a circular angle of 500°. The end point is approximated.

```
CIRC_REL {X 100,Y 3.2,Z -20},{Y 50},CA 500 C_VEL
```

11.6 Motion programming: spline

11.6.1 SPLINE ... ENDSPLINE

Description

A spline block may contain the following:

- Spline segments (only limited by the memory capacity.)
- PATH trigger
- 1 time block

  (>>> 11.6.5 "TIME_BLOCK" Page 328)
- Comments
- Blank lines

A spline block must not include any other instructions, e.g. variable assignments or logic statements. There must be no trigger between the last segment and ENDSPLINE.

A spline block does not trigger an advance run stop.

The following components must be specified for the first point in a spline block:

- X, Y, Z
- A, B, C
- E1 to E6 (if present)

This also applies if the first point of the auxiliary point is a SCIRC segment.

Syntax

SPLINE <WITH *SysVarSpline* = *Value1* <,*Value2* , …, *ValueN*> >

Spline segment1

<*Trigger*>

…

<*Spline segmentN-1*

<*Trigger*>>

<*Spline segmentN*>

ENDSPLINE <C_DIS>

Explanation of the syntax

Element	Description
SysVar-Spline	The following system variables may be used: $BASE, $TOOL, $IPO_MODE, $LOAD, $VEL, $VEL_AXIS, $VEL_EXTAX, $ACC, $ACC_AXIS, $ACC_EXTAX, $APO, $JERK, $ORI_TYPE, $CIRC_TYPE
Value	Value assignment to the system variable. The value is valid for every segment in the spline block except segments to which a value is assigned separately. A value can also be assigned to the system variables $VEL, $VEL_EXTAX, $ACC_EXTAX and $JERK via a function. The same restrictions apply to these functions as to functions in the spline trigger. (>>> 11.11.3 "TRIGGER WHEN PATH (for SPLINE)" Page 358)
C_DIS	Approximate the end point of the spline block (= last point in the block). $APO is used to define the earliest point at which the approximate positioning can begin.

Example

```
SPLINE
  SPL P1
  TRIGGER WHEN PATH=GET_PATH() ONSTART DELAY=0 DO <subprog> PRIO=-1
  SPL P2
  SLIN P3
  SPL P4
  SCIRC P5, P6 WITH $VEL.CP=0.2
  SPL P7 WITH $ACC={CP 2.0, ORI1 200, ORI2 200}
  SCIRC P8, P9
  SPL P10
ENDSPLINE
```

11.6.2 SLIN

Description

SLIN can be programmed as an individual block or as a segment in a spline block. It is possible to copy an individual SLIN motion into a spline block, but only if it does not contain an assignment to $BASE, $TOOL, $IPO_MODE or $LOAD.

Syntax

SLIN *End point* <WITH *SysVarSlin* = *Value1* <,*Value2* , …, *ValueN*>> <C_DIS>

Explanation of the syntax

Element	Description
End point	Type: POS, E6POS, FRAME The coordinates refer to the BASE coordinate system. ■ The following applies for spline segments: If not all components of the end point are specified, the controller takes the values of the previous segment for the missing components. The first segment in the spline block must be specified completely. ■ The following applies for individual motions: If not all components of the end point are specified, the controller takes the values of the previous position for the missing components. If the previous position is the end point of a circle with a circular angle, the values of the programmed end point are applied, not the values resulting from the circular angle. If no previous position is known (in the case of a block selection), the missing components are taken from the current robot position.
SysVarSlin	The following system variables may be used: $ACC, $ACC_AXIS, $ACC_EXTAX, $JERK, $ORI_TYPE, $VEL, $VEL_AXIS, $VEL_EXTAX ■ The following may also be used for SLIN in the spline block: $EX_AX_IGNORE ■ The following may also be used for individual SLIN motions: $APO, $BASE, $IPO_MODE, $LOAD, $TOOL
Value	Value assignment to the system variable. In the case of SLIN segments: The assignment applies only for this segment. It has no effect on subsequent segments. A value can also be assigned to the system variables $VEL, $VEL_EXTAX, $ACC_EXTAX and $JERK via a function. The same restrictions apply to these functions as to functions in the spline trigger. (>>> 11.11.3 "TRIGGER WHEN PATH (for SPLINE)" Page 358)
C_DIS	Only possible for individual SLIN motions: Approximate the end point. $APO is used to define the earliest point at which the approximate positioning can begin.

11.6.3 SCIRC

Description

SCIRC can be programmed as an individual block or as a segment in a spline block. It is possible to copy an individual SCIRC motion into a spline block, but only if it does not contain an assignment to $BASE, $TOOL, $IPO_MODE or $LOAD.

Syntax

`SCIRC` *Auxiliary point*, *End point* <`,  CA` *Auxiliary point*> <`WITH` *SysVarScirc* = *Value1* <,*Value2* , …, *ValueN*>> <`C_DIS`>

Explanation of the syntax

Element	Description
Auxiliary point End point	Type: POS, E6POS, FRAME The coordinates refer to the BASE coordinate system. ■ The following applies for spline segments: If not all components of the point are specified, the controller takes the values of the previous segment for the missing components. The first segment in the spline block must be specified completely. ■ The following applies for individual motions: If not all components of the point are specified, the controller takes the values of the previous position for the missing components. If the previous position is the end point of a circle with a circular angle, the values of the programmed end point are applied, not the values resulting from the circular angle. If no previous position is known (in the case of a block selection), the missing components are taken from the current robot position.
Circular angle	Specifies the overall angle of the circular motion. This makes it possible to extend the motion beyond the programmed end point or to shorten it. The actual end point thus no longer corresponds to the programmed end point. Unit: degrees. There is no limit; in particular, a circular angle greater than 360° can be programmed. ■ Positive circular angle: the circular path is executed in the direction Start point › Auxiliary point › End point. ■ Negative circular angle: the circular path is executed in the direction Start point › End point › Auxiliary point.
SysVarScirc	The following system variables may be used: $ACC, $ACC_AXIS, $ACC_EXTAX, $CIRC_MODE, $CIRC_TYPE, $JERK, $ORI_TYPE, $VEL, $VEL_AXIS, $VEL_EXTAX ■ The following may also be used for SCIRC in the spline block: $EX_AX_IGNORE ■ The following may also be used for individual SCIRC motions: $APO, $BASE, $IPO_MODE, $LOAD, $TOOL
Value	Value assignment to the system variable. In the case of SCIRC segments: The assignment applies only for this segment. It has no effect on subsequent segments. A value can also be assigned to the system variables $ACC_EXTAX, $JERK, $VEL and $VEL_EXTAX via a function. The same restrictions apply to these functions as to functions in the spline trigger. (>>> 11.11.3 "TRIGGER WHEN PATH (for SPLINE)" Page 358)
C_DIS	Only possible for individual SCIRC motions: Approximate the end point. $APO is used to define the earliest point at which the approximate positioning can begin.

Example

```
SCIRC P2, P3 WITH $CIRC_TYPE=#PATH
```

11.6.4 SPL

Description

SPL can only be programmed as a segment in a spline block. It is not possible to program an individual SPL motion.

Syntax

`SPL` *End point* <`WITH` *SysVarSpl* = *Value1* <,*Value2* , …, *ValueN*>>

Explanation of the syntax

Element	Description
End point	Type: POS, E6POS, FRAME The coordinates refer to the BASE coordinate system. If not all components of the end point are specified, the controller takes the values of the previous segment for the missing components. The first segment in the spline block must be specified completely.
SysVarSpl	The following system variables may be used: $ACC, $ACC_AXIS, $ACC_EXTAX, $CIRC_TYPE, $EX_AX_IGNORE, $JERK, $ORI_TYPE, $VEL, $VEL_AXIS, $VEL_EXTAX
Value	Value assignment to the system variable. The assignment applies only for this segment. It has no effect on subsequent segments. A value can also be assigned to the system variables $ACC_EXTAX, $JERK, $VEL and $VEL_EXTAX via a function. The same restrictions apply to these functions as to functions in the spline trigger. (>>> 11.11.3 "TRIGGER WHEN PATH (for SPLINE)" Page 358)

Example

```
SPL P4 WITH $ACC={CP 2.0, ORI1 200, ORI2 200}
```

11.6.5 TIME_BLOCK

Description

TIME_BLOCK can be used to execute a spline block or part of one in a defined time. It is also possible to allocate time components to areas of TIME_BLOCK.

Points can be modified in, added to or removed from the spline block without changing the time specifications. This enables the user to correct the Cartesian path and retain the existing timing.

A spline block may include 1 time block, i.e. 1 statement of the type TIME_BLOCK START … TIME_BLOCK END. This, in turn, may contain any number of TIME_BLOCK PART statements. The time block can only be used in spline blocks.

Syntax

`SPLINE`

<*Spline segment…*>

…

TIME_BLOCK START

<*Spline segment…*

...

<**TIME_BLOCK PART = Component[1]**>>

…

Spline segmentN

...

<TIME_BLOCK PART = Component[N]>

TIME_BLOCK END = Overall time

<*Spline segmentN+1*>

…

`ENDSPLINE`

Explanation of the syntax

It is not essential for there to be spline segments before TIME_BLOCK START and after TIME_BLOCK END. It is nonetheless advisable to program as follows:

- There is at least 1 spline segment between SPLINE and TIME_BLOCK START.
- There is at least 1 spline segment between TIME_BLOCK END and ENDSPLINE.

Advantages:

- The programmed overall time is maintained exactly even in the case of approximate positioning.
- In the case of a block selection, the original velocity profile is maintained if there is at least 1 spline segment between SPLINE and TIME_BLOCK START.
- Segments before TIME_BLOCK START make it possible to accelerate to the desired velocity.

Element	Description
Component[x]	Type: INT or REAL; constant, variable or function Desired time component [x] for the following distance: ■ From the point before TIME_BLOCK PART = Component[x-1] to the point before TIME_BLOCK PART = Component[x] ■ If TIME_BLOCK PART = Component[x-1] does not exist: From the point before TIME_BLOCK START to the point before TIME_BLOCK PART = Component[x] The time components are maintained as accurately as possible by the robot controller. Generally, however, they are not maintained exactly. The user can assign the components in such a way that they add up to 100. The components can then be considered as percentages. The components do not have to add up to 100, however, and can have any sum! The robot controller always equates *Overall time* to the sum of the components. This allows the components to be used very flexibly and also changed. If time components are assigned, there must always be a TIME_BLOCK PART before TIME_BLOCK END. There must be no segments in between.
Overall time	Type: INT or REAL; constant, variable or function; unit: s Time in which the following distance is traveled: ■ From the point before TIME_BLOCK START to the point before TIME_BLOCK END The value must be greater than 0. The overall time is maintained exactly. If this time cannot possibly be maintained, e.g. because too short a time has been programmed, the robot executes the motion in the fastest possible time. In T1 and T2, a message is also displayed.

If the value for *Component[x]* or *Overall time* is assigned via a function, the same restrictions apply as for the functions in the spline trigger.

(>>> 11.11.3 "TRIGGER WHEN PATH (for SPLINE)" Page 358)

Example

```
SPLINE
  SLIN P1
  SPL P2
  TIME_BLOCK START
    SLIN P3
  TIME_BLOCK PART = 12.7
    SPL P4
    SPL P5
    SPL P6
  TIME_BLOCK PART = 56.4
    SCIRC P7, P8
    SPL P9
  TIME_BLOCK PART = 27.8
  TIME_BLOCK END = 3.9
  SLIN P10
ENDSPLINE
```

Points P2 to P9 are executed exactly in the programmed time of 3.9 s.

The robot controller equates the overall time of 3.9 s to the sum of the components, i.e. 96.9. This gives the value of the components in seconds.

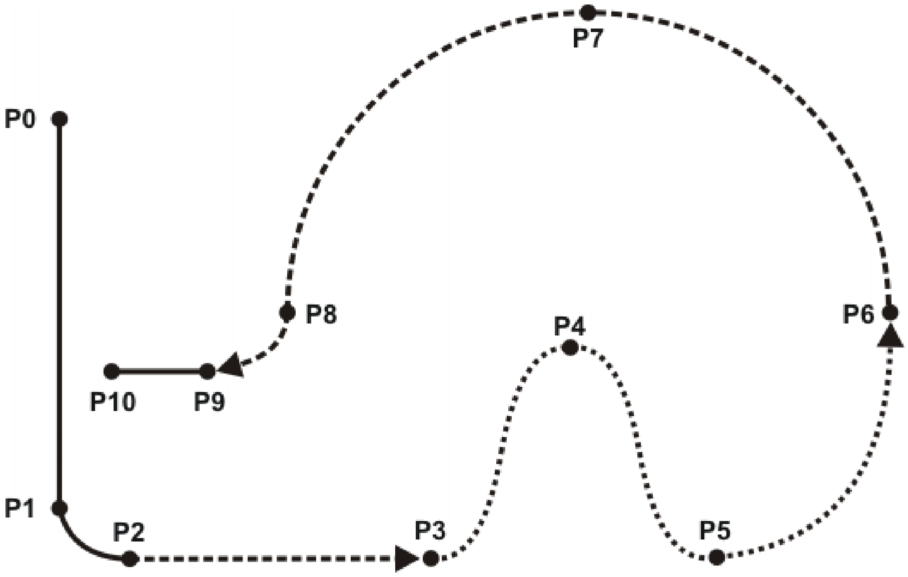


Fig. 11-1: Example TIME_BLOCK

Block selection

In the case of a block selection, the original velocity profile is maintained if there is at least 1 spline segment between SPLINE and TIME_BLOCK START.

(A block selection to TIME_BLOCK is the equivalent of a block selection to the subsequent motion block.)

Example: Block selection with segment before TIME_BLOCK START

```
LIN P1
SPLINE
  SPL P2
  TIME_BLOCK START
    SLIN P3
  TIME_BLOCK PART = 0.5
    SPL P4
    SPL P5
    SPL P6
  TIME_BLOCK PART = 2.2
  TIME_BLOCK END = 2.7
  ...
```

- The defined overall time applies from the point before TIME_BLOCK START, i.e. P2.
- A block selection is now made to P5.
- For continued motion from P5, the robot controller plans the velocity for the entire spline block.

  The first point in the spline block is P2. The velocity is thus planned from the same point as if the spline were to be executed during normal program execution (= from P1).

Example: Block selection without segment before TIME_BLOCK START

```
LIN P1
SPLINE
  TIME_BLOCK START
    SLIN P2
  TIME_BLOCK PART = 0.5
    SPL P3
    SPL P4
    SPL P5
  TIME_BLOCK PART = 2.2
  TIME_BLOCK END = 2.7
  ...
```

- The defined overall time applies from the point before TIME_BLOCK START, i.e. P1.
- A block selection is now made to P5.
- For continued motion from P5, the robot controller plans the velocity for the entire spline block.

  The first point in the spline block is P2. P1 is situated before the spline block and can thus no longer be taken into consideration by the robot controller.

  It is thus not possible to plan the velocity for the spline block as exactly as if it were to be executed during normal program execution (= from P1).

System variable

The data of the time-based spline can be read via the system variable $PATHTIME. $PATHTIME is filled with the data as soon as the robot controller has completed the planning of the spline block. The data are retained until the next spline block has been planned.

$PATHTIME is a structure and consists of the following components:

Component	Description
REAL $PATHTIME.TOTAL	Overall time actually required (s)
REAL $PATHTIME.SCHEDULED	Overall time planned (s)
REAL $PATHTIME.PROGRAMMED	Overall time programmed (s)
INT $PATHTIME.N_SECTIONS	Number N of time components
REAL $PATHTIME.MAX_DEV	Maximum deviation of all time components between the programmed time and the planned time (%)
INT $PATHTIME.MAX_DEV_SECTION	Number of the time component with the greatest deviation between the programmed time and the planned time

11.7 Program execution control

11.7.1 CONTINUE

Description

Prevents an advance run stop that would otherwise occur in the following program line.

> CONTINUE always applies to the following program line, even if this is a blank line!

Syntax

`CONTINUE`

Example

Preventing both advance run stops:

```
CONTINUE
$OUT[1]=TRUE
CONTINUE
$OUT[2]=FALSE
```

In this case, the outputs are set in the advance run. The exact point at which they are set cannot be foreseen.

11.7.2 EXIT

Description

Exit from a loop. The program is then continued after the loop. EXIT may be used in any loop.

Syntax

`EXIT`

Example

The loop is exited when `$IN[1]` is set to TRUE. The program is then continued after ENDLOOP.

```
DEF EXIT_PROG()
PTP HOME
LOOP
   PTP POS_1
   PTP POS_2
   IF $IN[1] == TRUE THEN
      EXIT
   ENDIF
   CIRC HELP_1, POS_3
   PTP POS_4
ENDLOOP
PTP HOME
END
```

11.7.3 FOR ... TO ... ENDFOR

Description

A statement block is repeated until a counter exceeds or falls below a defined value.

After the last execution of the statement block, the program is resumed with the first statement after ENDFOR. The loop execution can be exited prematurely with EXIT.

Loops can be nested. In the case of nested loops, the outer loop is executed completely first. The inner loop is then executed completely.

Syntax

FOR *Counter* = *Start* TO *End* <STEP *Increment*>

<*Statements*>

ENDFOR

Explanation of the syntax

Element	Description
Counter	Type: INT Variable that counts the number of times the loop has been executed. The preset value is *Start*. The variable must first be declared. The value of *Counter* can be used in statements inside and outside of the loop. Once the loop has been exited, *Counter* retains its most recent value.
Start; End	Type: INT *Counter* must be preset to the value *Start*. Each time the loop is executed, the value of *Counter* is automatically increased by the increment. If the value exceeds or falls below the *End* value, the loop is terminated.
Increment	Type: INT Value by which *Counter* is changed every time the loop is executed The value may be negative. Default value: 1. ■ Positive value: the loop is ended if *Counter* is greater than *End*. ■ Negative value: the loop is ended if *Counter* is less than *End*. The value may not be either zero or a variable.

Example

The variable B is incremented by 1 after each of 5 times the loop is executed.

```
INT A
...
FOR A=1 TO 10 STEP 2
   B=B+1
ENDFOR
```

11.7.4 GOTO

Description

Unconditional jump to a specified position in the program. Program execution is resumed at this position.

The destination must be in the same subprogram or function as the GOTO statement.

The following jumps are not possible:

- Into an IF statement from outside.
- Into a loop from outside.
- From one CASE statement to another CASE statement.

> GOTO statements lead to a loss of structural clarity within a program. It is better to work with IF, SWITCH or a loop instead.

Syntax

GOTO *Label*

...

*Label***:**

Explanation of the syntax

Element	Description
Label	Position to which a jump is made. At the destination position, *Label* must be followed by a colon.

Example 1

Unconditional jump to the program position `GLUESTOP`.

```
GOTO GLUESTOP
...
GLUESTOP:
```

Example 2

Unconditional jump from an IF statement to the program position `END`.

```
IF X>100 THEN
   GOTO ENDE
ELSE
   X=X+1
ENDIF
A=A*X
...
ENDE:
END
```

11.7.5 HALT

Description

Stops the program. The last motion instruction to be executed will, however, be completed.

Execution of the program can only be resumed using the Start key. The next instruction after HALT is then executed.

In an interrupt program, program execution is only stopped after the advance run has been completely executed.

Syntax

`HALT`

11.7.6 IF ... THEN ... ENDIF

Description

Conditional branch. Depending on a condition, either the first statement block (THEN block) or the second statement block (ELSE block) is executed. The program is then continued after ENDIF.

The ELSE block may be omitted. If the condition is not satisfied, the program is then continued at the position immediately after ENDIF.

There is no limit on the number of statements contained in the statement blocks. Several IF statements can be nested in each other.

Syntax

`IF` *Condition* `THEN`

Statements

`<ELSE`

Statements`>`

`ENDIF`

Explanation of the syntax

Element	Description
Condition	Type: BOOL Possible: ■ Variable of type BOOL ■ Function of type BOOL ■ Logic operation, e.g. a comparison, with a result of type BOOL

Example 1

IF statement without ELSE

```
IF A==17 THEN
   B=1
ENDIF
```

Example 2

IF statement with ELSE

```
IF $IN[1]==TRUE THEN
   $OUT[17]=TRUE
ELSE
   $OUT[17]=FALSE
ENDIF
```

11.7.7 LOOP ... ENDLOOP

Description

Loop that endlessly repeats a statement block. The loop execution can be exited with EXIT.

Loops can be nested. In the case of nested loops, the outer loop is executed completely first. The inner loop is then executed completely.

Syntax

`LOOP`

Statements

`ENDLOOP`

Example

The loop is executed until input $IN[30] is set to true.

```
LOOP
   LIN P_1
   LIN P_2
   IF $IN[30]==TRUE THEN
      EXIT
   ENDIF
ENDLOOP
```

11.7.8 REPEAT ... UNTIL

Description

Non-rejecting loop. Loop that is repeated until a certain condition is fulfilled.

The statement block is executed at least once. The condition is checked after each loop execution. If the condition is met, program execution is resumed at the first statement after the UNTIL line.

Loops can be nested. In the case of nested loops, the outer loop is executed completely first. The inner loop is then executed completely.

Syntax

`REPEAT`

Statements

`UNTIL` *Termination condition*

Explanation of the syntax

Element	Description
Termination condition	Type: BOOL Possible: ■ Variable of type BOOL ■ Function of type BOOL ■ Logic operation, e.g. a comparison, with a result of type BOOL

Example 1

The loop is to be executed until $IN[1] is true.

```
R=1
REPEAT
   R=R+1
UNTIL $IN[1]==TRUE
```

Example 2

The loop is executed once, even though the termination condition is already fulfilled before the loop execution, because the termination condition is not checked until the end of the loop. After execution of the loop, `R` has the value 102.

```
R=101
REPEAT
   R=R+1
UNTIL R>100
```

11.7.9 SWITCH ... CASE ... ENDSWITCH

Description

Selects one of several possible statement blocks, according to a selection criterion. Every statement block has at least one identifier. The block whose identifier matches the selection criterion is selected.

Once the block has been executed, the program is resumed after ENDSWITCH.

If no identifier agrees with the selection criterion, the DEFAULT block is executed. If there is no DEFAULT block, no block is executed and the program is resumed after ENDSWITCH.

> ℹ The SWITCH statement cannot be prematurely exited using EXIT.

Syntax

`SWITCH` *Selection criterion*

`CASE` *Identifier1* `<,`*Identifier2*`,...>`

Statement block

`<CASE` *IdentifierM* `<,`*IdentifierN*`,...>`

Statement block `>`

`<DEFAULT`

Default statement block`>`

`ENDSWITCH`

There must be no blank line or comment between the SWITCH line and the first CASE line. DEFAULT may only occur once in a SWITCH statement.

Explanation of the syntax

Element	Description
Selection criterion	Type: INT, CHAR, ENUM This can be a variable, a function call or an expression of the specified data type.
Identifier	Type: INT, CHAR, ENUM The data type of the identifier must match the data type of the selection criterion. A statement block can have any number of identifiers. Multiple block identifiers must be separated from each other by a comma.

Example 1

Selection criterion and identifier are of type INT.

```
INT VERSION
...
SWITCH VERSION
   CASE 1
      UP_1()
   CASE 2,3
      UP_2()
      UP_3()
      UP_3A()
   DEFAULT
      ERROR_UP()
ENDSWITCH
```

Example 2

Selection criterion and identifier are of type CHAR. The statement `SP_5()` is never executed here because the identifier `C` has already been used.

```
SWITCH NAME
   CASE "A"
      UP_1()
   CASE "B","C"
      UP_2()
      UP_3()
   CASE "C"
      UP_5()
ENDSWITCH
```

11.7.10 WAIT FOR

Description

Stops the program until a specified condition is fulfilled. Program execution is then resumed.

> If, due to incorrect formulation, the expression can never take the value TRUE, the compiler does not recognize this. In this case, execution of the program will be permanently halted because the program is waiting for a condition that cannot be fulfilled.

Syntax

`WAIT FOR` *Condition*

Explanation of the syntax

Element	Description
Condition	Type: BOOL Condition, the fulfillment of which allows program execution to be resumed. ■ If the condition is already TRUE when WAIT is called, program execution is not halted. ■ If the condition is FALSE, program execution is stopped until the condition is TRUE.

Example

Interruption of program execution until `$IN[17]` is TRUE:

```
WAIT FOR $IN[17]
```

Interruption of program execution until `BIT1` is FALSE:

```
WAIT FOR BIT1==FALSE
```

11.7.11 WAIT SEC

Description

Halts execution of the program and continues it after a wait time. The wait time is specified in seconds.

Syntax

`WAIT SEC` *Wait time*

Explanation of the syntax

Element	Description
Wait time	Type: INT, REAL Number of seconds for which program execution is to be interrupted. If the value is negative, the program does not wait. With small wait times, the accuracy is determined by a multiple of 12 ms.

Example

Interruption of program execution for 17.156 seconds:

```
WAIT SEC 17.156
```

Interruption of program execution in accordance with the variable value of V_WAIT in seconds:

```
WAIT SEC V_ZEIT
```

11.7.12 WHILE ... ENDWHILE

Description

Rejecting loop. Loop that is repeated as long as a certain condition is fulfilled.

If the condition is not met, program execution is resumed at the first statement after the ENDWHILE line. The condition is checked before each loop execution. If the condition is not already fulfilled beforehand, the statement block is not executed.

Loops can be nested. In the case of nested loops, the outer loop is executed completely first. The inner loop is then executed completely.

Syntax

`WHILE` *Repetition condition*

Statement block

`ENDWHILE`

Explanation of the syntax

Element	Description
Repetition condition	Type: BOOL Possible: ■ Variable of type BOOL ■ Function of type BOOL ■ Logic operation, e.g. a comparison, with a result of type BOOL

Example 1

The loop is executed 99 times. After execution of the loop, `W` has the value 100.

```
W=1
WHILE W<100
   W=W+1
ENDWHILE
```

Example 2

The loop is executed as long as `$IN[1]` is true.

```
WHILE $IN[1]==TRUE
   W=W+1
ENDWHILE
```

11.8 Inputs/outputs

11.8.1 ANIN

Description

Cyclical reading (every 12 ms) of an analog input.

ANIN triggers an advance run stop.

Syntax

Starting cyclical reading:

`ANIN ON` *Value* `=` *Factor* `*` *Signal name* `*` <*±Offset*>

Ending cyclical reading:

`ANIN OFF` *Signal name*

> The robot controller has 32 analog inputs ($ANIN[1] … $ANIN[32]).
>
> - A maximum of three ANIN ON statements can be used at the same time.
> - A maximum of two ANIN ON statements can use the same variable *Value* or access the same analog input.
> - All of the variables used in an ANIN statement must be declared in data lists (locally or in $CONFIG.DAT).

Explanation of the syntax

Element	Description
Value	Type: REAL The result of the cyclical reading is stored in *Value*. *Value* can be a variable or a signal name for an output.
Factor	Type: REAL Any factor. It can be a constant, variable or signal name.
Signal name	Type: REAL Specifies the analog input. *Signal name* must first have been declared with `SIGNAL` . It is not possible to specify the analog input $ANIN[x] directly instead of the signal name. The values of an analog input $ANIN[x] range between +1.0 and -1.0 and represent a voltage of +10 V to -10 V.
Offset	Type: REAL It can be a constant, variable or signal name.

Example

In this example, the program override (= system variable $OV_PRO) is defined by means of the analog input $ANIN[1].

$ANIN[1] must first be linked to a freely selected signal name, in this case SIGNAL_1, in the declaration section.

```
SIGNAL SIGNAL_1 $ANIN[1]
...
ANIN ON $OV_PRO = 1.0 * SIGNAL_1
```

The cyclical scanning of `SIGNAL_1` is ended using the `ANIN OFF` statement.

```
ANIN OFF SIGNAL_1
```

11.8.2 ANOUT

Description

Cyclical writing (every 12 ms) to an analog output.

ANOUT triggers an advance run stop.

Syntax

Starting cyclical writing:

`ANOUT ON` *Signal name* `=` *Factor* `*` *Control element* <*±Offset*> `<DELAY =` *±Time*`>` `<MINIMUM =` *Minimum value*`>` `<MAXIMUM =` *Maximum value*`>`

Ending cyclical writing:

`ANOUT OFF` *Signal name*

The robot controller has 32 analog outputs ($ANOUT[1] … $ANOUT[32]).

- A maximum of four ANOUT ON statements can be used at the same time.
- All of the variables used in an ANOUT statement must be declared in data lists (locally or in $CONFIG.DAT).

Explanation of the syntax

Element	Description
Signal name	Type: REAL Specifies the analog output. *Signal name* must first have been declared with SIGNAL . It is not possible to specify the analog output $ANOUT[x] directly instead of the signal name. The values of an analog output $ANOUT[x] range between +1.0 and -1.0 and represent a voltage of +10 V to -10 V.
Factor	Type: REAL Any factor. It can be a constant, variable or signal name.
Control element	Type: REAL It can be a constant, variable or signal name.
Offset	Type: REAL It can be a constant, variable or signal name.
Time	Type: REAL Unit: seconds. By using the keyword DELAY and entering a positive or negative amount of time, the output signal can be delayed (+) or set early (-).
Minimum value, Maximum value	Type: REAL Minimum and/or maximum voltage to be present at the output. The actual value does not fall below/exceed these values, even if the calculated values fall outside this range. Permissible values: -1.0 to +1.0 (corresponds to -10 V to +10 V). It can be a constant, variable, structure component or array element. The minimum value must always be less than the maximum value. The sequence of the keywords MINIMUM and MAXIMUM must be observed.

Example

In this example, the output $ANOUT[5] controls the adhesive output.

A freely selected name, in this case GLUE, is assigned to the analog output in the declaration section. The amount of adhesive is to be dependent on the current path velocity (= system variable $VEL_ACT). Furthermore, the output signal is to be generated 0.5 seconds early. The minimum voltage is to be 3 V.

```
SIGNAL GLUE $ANOUT[5]
...
ANOUT ON GLUE = 0.5 * $VEL_ACT DELAY=-0.5 MINIMUM=0.30
```

The cyclical analog output is ended by using ANOUT OFF:

```
ANOUT OFF GLUE
```

11.8.3 PULSE

Description

Sets a pulse. The output is set to a defined level for a specified duration. The output is then reset automatically by the system. The output is set and reset irrespective of the previous level of the output.

At any one time, pulses may be set at a maximum of 16 outputs.

If PULSE is programmed before the first motion block, the pulse duration also elapses if the Start key is released again and the robot has not yet reached the BCO position.

The PULSE statement triggers an advance run stop. It is only executed concurrently with robot motion if it is used in a TRIGGER statement.

> The pulse is not terminated in the event of an EMERGENCY STOP, an operator stop or an error stop!

Syntax

`PULSE (`*Signal, Level, Pulse duration*`)`

Explanation of the syntax

Element	Description
Signal	Type: BOOL Output to which the pulse is to be fed. The following are permitted: ■ OUT[No] ■ Signal variable
Level	Type: BOOL Logic expression: ■ TRUE represents a positive pulse (high). ■ FALSE represents a negative pulse (low).
Pulse duration	Type: REAL Range of values: 0.1 to 3.0 seconds. Pulse durations outside this range trigger a program stop. Pulse interval: 0.1 seconds, i.e. the pulse duration is rounded up or down. The PULSE statement is executed in the controller at the low-priority clock rate. This results in a tolerance in the order of the pulse interval (0.1 seconds). The time deviation is about 1% - 2% on average. The deviation is about 13% for very short pulses.

$OUT+PULSE

If an output is already set before the pulse, it will be reset by the falling edge of the pulse.

```
$OUT[50] = TRUE
PULSE($OUT[50],TRUE,0.5)
```

Actual pulse characteristic at output 50:

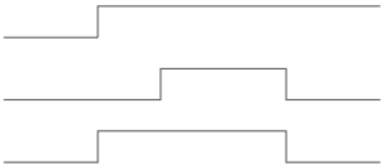


Fig. 11-2: $OUT+PULSE, example 1

If a negative pulse is applied to an output that is set to Low, the output remains Low until the end of the pulse and is then set to High:

```
$OUT[50] = FALSE
PULSE($OUT[50],FALSE,0.5)
```

Actual pulse characteristic at output 50:

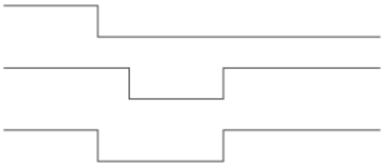


Fig. 11-3: $OUT+PULSE, example 2

PULSE+$OUT

If the same output is set during the pulse duration, it will be reset by the falling edge of the pulse.

```
PULSE($OUT[50],TRUE,0.5)
$OUT[50] = TRUE
```

Actual pulse characteristic at output 50:

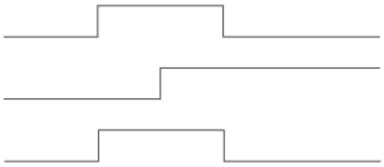


Fig. 11-4: PULSE+$OUT, example 1

If the output is reset during the pulse duration, the pulse duration is reduced accordingly:

```
PULSE($OUT[50],TRUE,0.5)
$OUT[50] = FALSE
```

Actual pulse characteristic at output 50:

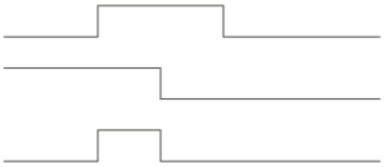


Fig. 11-5: PULSE+$OUT, example 2

If an output is set to FALSE during a pulse and then back to TRUE, the pulse is interrupted and then resumed when the output is set to TRUE. The overall duration from the first rising edge to the last falling edge (i.e. including the duration of the interruption) corresponds to the duration specified in the PULSE statement.

```
PULSE($OUT[50],TRUE,0.8)
$OUT[50]=FALSE
$OUT[50]=TRUE
```

Actual pulse characteristic at output 50:

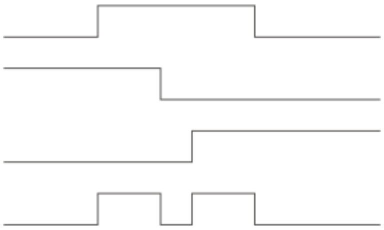


Fig. 11-6: PULSE+$OUT, example 3

The actual pulse characteristic is only specified as above if $OUT[x]=TRUE is set during the pulse. If $OUT[x]=TRUE is not set until after the pulse (see line 3), then the actual pulse characteristic is as follows (line 4):

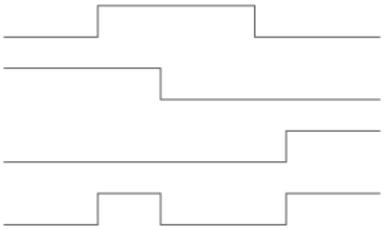


Fig. 11-7: PULSE+$OUT, example 4

PULSE+PULSE

If several PULSE statements overlap, it is always the last PULSE statement that determines the end of the overall pulse duration.

If a pulse is activated again before the falling edge, the duration of the second pulse starts at this moment. The overall pulse duration is thus shorter than the sum of the values of the first and second pulses:

```
PULSE($OUT[50],TRUE,0.5)
PULSE($OUT[50],TRUE,0.5)
```

Actual pulse characteristic at output 50:

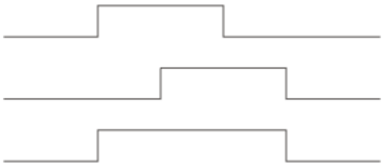


Fig. 11-8: PULSE+PULSE, example 1

If, during the pulse duration of a positive pulse, a negative pulse is sent to the same output, only the second pulse is taken into consideration from this moment onwards:

```
PULSE($OUT[50],TRUE,0.5)
PULSE($OUT[50],FALSE,0.5)
```

Actual pulse characteristic at output 50:

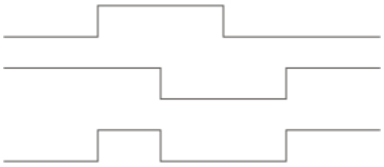


Fig. 11-9: PULSE+PULSE, example 2

```
PULSE($OUT[50],TRUE,3.0)
PULSE($OUT[50],FALSE,1.0)
```

Actual pulse characteristic at output 50:

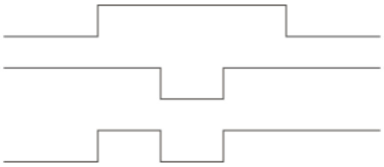


Fig. 11-10: PULSE+PULSE, example 3

PULSE+END

If a pulse is programmed before the END statement, the duration of program execution is increased accordingly.

```
PULSE($OUT[50],TRUE,0.8)
END
```

Program active

Actual pulse characteristic at output 50:

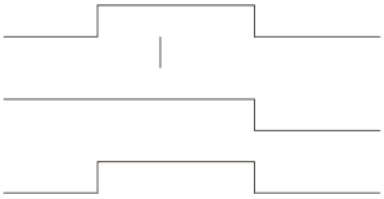


Fig. 11-11: PULSE+END, example

PULSE+RESET/ CANCEL

If program execution is reset (RESET) or aborted (CANCEL) while a pulse is active, the pulse is immediately reset:

```
PULSE($OUT[50],TRUE,0.8)
RESET or CANCEL
```

Actual pulse characteristic at output 50:

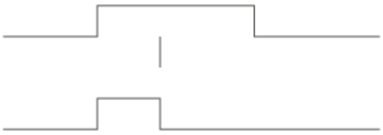


Fig. 11-12: PULSE+RESET, example

11.8.4 SIGNAL

Description

SIGNAL declarations must appear in the declaration section.

- SIGNAL links predefined signal variables for inputs or outputs with a name.

  A SIGNAL declaration is required in order to be able to address an analog input or output. An input or output may appear in several SIGNAL declarations.
- SIGNAL declarations that are predefined in the system can be deactivated by means of SIGNAL in conjunction with the keyword FALSE.

  Can only be used in KRC:\STEU:\MADA:\$machine.dat.

Syntax

Declaration of signal names for inputs and outputs:

`SIGNAL` *Signal name Signal variable* `<TO` *Signal variable*>

Deactivation of a SIGNAL declaration predefined in the system:

`SIGNAL` *System signal name* `FALSE`

Explanation of the syntax

Element	Description
Signal name	Any name
Signal variable	Predefined signal variable. The following types are available: ■ $IN[x] ■ $OUT[x] ■ $ANIN[x] ■ $ANOUT[x]

Element	Description
TO	Groups together several consecutive binary inputs or outputs (max. 32) to form a digital input or output. The combined signals can be addressed with a decimal name, a hexadecimal name (prefix H) or with a bit pattern name (prefix B). They can also be processed with Boolean operators.
System signal name	Signal name predefined in the system, e.g. $T1.
FALSE	Deactivates a SIGNAL declaration predefined in the system. The inputs or outputs to which the SIGNAL declaration refers are thus available again for other purposes. FALSE is not a Boolean value here, but a keyword. The option TRUE is not available. If the SIGNAL declaration that has been deactivated by means of FALSE is to be reactivated, the program line containing the entry FALSE must be deleted.

Example 1

The output $OUT[7] is assigned the name START_PROCESS. The output $OUT[7] is set.

```
SIGNAL START_PROCESS $OUT[7]
START_PROCESS = TRUE
```

Example 2

The outputs $OUT[1] to $OUT[8] are combined to form one digital output under the name OUTWORT. The outputs $OUT[3], $OUT[4], $OUT[5] and $OUT[7] are set.

```
SIGNAL OUTWORT $OUT[1] TO $OUT[8]
OUTWORT = 'B01011100'
```

11.9 Subprograms and functions

11.9.1 RETURN

Description

Jump from a subprogram or function back to the program from which the subprogram or function was called.

Subprograms

RETURN can be used to return to the main program if a certain condition is met in the subprogram. No values from the subprogram can be transferred to the main program.

Functions

Functions must be ended by a RETURN statement containing the value that has been determined. The determined value is hereby transferred to the program from which the function was called.

Syntax

For subprograms:

`RETURN`

For functions:

`RETURN` *Function value*

Explanation of the syntax

Element	Description
Function value	Type: The data type of *Function value* must match the data type of the function. *Function value* is the value determined by the function. The value can be specified as a constant, a variable or an expression.

Example 1

Return from a subprogram to the program from which it was called, dependent on a condition.

```
DEF PROG_2()
   ...
   IF $IN[5]==TRUE THEN
   RETURN
   ...
END
```

Example 2

Return from a function to the program from which it was called. The value X is transferred.

```
DEFFCT INT CALCULATE(X:IN)
   INT X
   X=X*X
   RETURN X
ENDFCT
```

11.10 Interrupt programming

11.10.1 BRAKE

Description

BRAKE stops the robot.

> BRAKE may only be used in an interrupt program.

The interrupt program is not continued until the robot has come to a stop. The robot motion is resumed as soon as the interrupt program has been completed.

Syntax

`BRAKE <F>`

Explanation of the syntax

Element	Description
F	F triggers a STOP 1. In the case of a BRAKE statement without F, the robot brakes with a STOP 2.

Example

(>>> 11.10.3 "INTERRUPT" Page 349)

11.10.2 INTERRUPT ... DECL ... WHEN ... DO

Description

In the case of a defined event, e.g. an input, the controller interrupts the current program and executes a defined subprogram. The event and the subprogram are defined by INTERRUPT ... DECL ... WHEN ... DO.

Once the subprogram has been executed, the interrupted program is resumed at the point at which it was interrupted. Exception: RESUME.

A subprogram called by an interrupt is called an interrupt program.

A maximum of 32 interrupts may be declared simultaneously. An interrupt declaration may be overwritten by another at any time.

> The interrupt declaration is a statement. It must be situated in the statements section of the program and not in the declaration section!

> When first declared, an interrupt is deactivated. The interrupt must be activated before the system can react to the defined event!
> (>>> 11.10.3 "INTERRUPT" Page 349)

Syntax

```
<GLOBAL> INTERRUPT DECL Prio WHEN Event DO Subprogram
```

Explanation of the syntax

Element	Description
`GLOBAL`	An interrupt is only recognized at, or below, the level in which it is declared. In other words, an interrupt declared in a subprogram is not recognized in the main program (and cannot be activated there). If an interrupt is also to be recognized at higher levels, the declaration must be preceded by the keyword GLOBAL.
Prio	Type: INT If several interrupts occur at the same time, the interrupt with the highest priority is processed first, then those of lower priority. 1 = highest priority. Priorities 1, 2, 4 to 39 and 81 to 128 are available. **Note:** Priorities 3 and 40 to 80 are reserved for use by the system. They must **not** be used by the user because this would cause system-internal interrupts to be overwritten and result in errors.
Event	Type: BOOL Event that is to trigger the interrupt. Structure components are impermissible. The following are permitted: ■ a Boolean constant ■ a Boolean variable ■ a signal name ■ a comparison ■ a simple logic operation: NOT, OR, AND or EXOR
Subprogram	The name of the interrupt program to be executed. Runtime variables may not be transferred to the interrupt program as parameters, with the exception of variables declared in a data list.

> GLOBAL can only be used for variables and user-defined data types if they have been declared in a data list.
> To use GLOBAL, the entry GLOBAL_KEY in the file PROGRESS.INI, in the INIT directory, must be set to TRUE: GLOBAL_KEY=TRUE

Example 1

Declaration of an interrupt with priority 23 that calls the subprogram SP1 if $IN[12] is true. The parameters 20 and VALUE are transferred to the subprogram.

```
INTERRUPT DECL 23 WHEN $IN[12]==TRUE DO UP1(20,WERT)
```

Example 2

Two objects, the positions of which are detected by two sensors connected to inputs 6 and 7, are located on a programmed path. The robot is to be moved subsequently to these two positions.

For this purpose, the two detected positions are saved as points P_1 and P_2. These points are then addressed in the second section of the main program.

If the robot controller detects an event defined by means of INTERRUPT ... DECL ... WHEN ... DO, it always saves the current robot position in the system variables $AXIS_INT (axis-specific) and $POS_INT (Cartesian).

Main program:

```
DEF PROG()
...
INTERRUPT DECL 10 WHEN $IN[6]==TRUE DO UP1()
INTERRUPT DECL 20 WHEN $IN[7]==TRUE DO UP2()
...
INTERRUPT ON
LIN START
LIN END
INTERRUPT OFF
LIN P_1
LIN P_2
...
END
```

Local interrupt program 1:

```
DEF UP1()
P_1=$POS_INT
END
```

Local interrupt program 2:

```
DEF UP2()
P_2=$POS_INT
END
```

11.10.3 INTERRUPT

Description

Executes one of the following actions:

- Activates an interrupt.
- Deactivates an interrupt.
- Disables an interrupt.
- Enables an interrupt.

The interrupt must previously have been declared. (>>> 11.10.2 "INTERRUPT ... DECL ... WHEN ... DO" Page 347)

Syntax

`INTERRUPT` *Action* <*Number*>

Explanation of the syntax

Element	Description
Action	■ `ON`: Activates an interrupt. ■ `OFF`: Deactivates an interrupt. ■ `DISABLE`: Disables an activated interrupt. ■ `ENABLE`: Enables a disabled interrupt.
Number	Type: INT Number (= priority) of the interrupt to which the *Action* is to refer. *Number* can be omitted. In this case, `ON` or `OFF` refers to all declared interrupts, while `DISABLE` or `ENABLE` refers to all active interrupts.

Up to 16 interrupts may be active at any one time. In this regard, particular attention must be paid to the following:

- If, in the case of `INTERRUPT ON`, the *Number* is omitted, all declared interrupts are activated. The maximum permissible total of 16 may not be exceeded, however.
- If a trigger calls a subprogram, it counts as an active interrupt until the subprogram has been executed.

If, in the interrupt declaration, a Boolean variable, e.g. an input, has been defined as the *Event*:

- In this case, the interrupt is triggered by a change of state, e.g. in the case of $IN[x]==TRUE by the change from FALSE to TRUE. The state must therefore not already be present at INTERRUPT ON, as the interrupt is not then triggered!
- Furthermore, the following must also be considered in this case: the change of state must not occur until at least one interpolation cycle after INTERRUPT ON.

  (This can be achieved by programming a WAIT SEC 0.012 after INTERRUPT ON. If no advance run stop is desired, a CONTINUE command can also be programmed before the WAIT SEC.)

  The reason for this is that INTERRUPT ON requires one interpolation cycle (= 12 ms) before the interrupt is actually activated. If the state changes before this, the interrupt cannot detect the change.

Example 1

The interrupt with priority 2 is activated. (The interrupt must already be declared.)

```
INTERRUPT ON 2
```

Example 2

A non-path-maintaining EMERGENCY STOP is executed via the hardware during application of adhesive. The application of adhesive is stopped by the program and the adhesive gun is repositioned onto the path after enabling (by input 10).

```
DEF PROG()
...
INTERRUPT DECL 1 WHEN $STOPMESS DO STOP_PROG()
LIN P_1
INTERRUPT ON
LIN P_2
INTERRUPT OFF
...
END
```

```
DEF STOP_PROG()
BRAKE F
GLUE=FALSE
WAIT FOR $IN[10]
LIN $POS_RET
GLUE=TRUE
END
```

11.10.4 RESUME

Description

RESUME may only occur in interrupt programs. (Not in interrupt programs, however, that are called by an interrupt that is declared as GLOBAL.) RESUME cancels all running interrupt programs and subprograms up to the level at which the current interrupt was declared.

When the RESUME statement is activated, the advance run pointer must not be at the level where the interrupt was declared, but at least one level lower.

Changing the variable $BASE in the interrupt program only has an effect there. The computer advance run, i.e. the variable $ADVANCE, must not be modified in the interrupt program.

The behavior of the robot controller after RESUME depends on the following motion instruction:

- PTP instruction: is executed as a PTP motion.
- LIN instruction: is executed as a LIN motion.
- CIRC instruction: is always executed as a LIN motion!

  Following a RESUME statement, the robot is not situated at the original start point of the CIRC motion. The motion will thus differ from how it was originally planned; this can potentially be very dangerous, particularly in the case of CIRC motions.

⚠ WARNING If the first motion instruction after RESUME is a CIRC motion, this is always executed as LIN! This must be taken into consideration when programming RESUME statements. The robot must be able to reach the end point of the CIRC motion safely, by means of a LIN motion, from any position in which it could find itself when the RESUME statement is executed.
Failure to observe this may result in death to persons, physical injuries or damage to property.

Syntax

```
RESUME
```

Example

The robot is to search for a part on a path. The part is detected by means of a sensor at input 15. Once the part has been found, the robot is not to continue to the end point of the path, but to return to the interrupt position and pick up the part. The main program is then to be resumed.

Motions that are to be canceled by means of BRAKE and RESUME must be programmed in a subprogram. (Here `SEARCH()`.)

Main program:

```
DEF PROG()
INI
...
INTERRUPT DECL 21 WHEN $IN[15] DO FOUND()
PTP HOME
...
SEARCH()
$ADVANCE=3
...
END
```

Subprogram with search path:

When the RESUME statement is activated, the advance run pointer must not be at the level where the current interrupt was declared. To prevent this, the advance run is set to 0 here in the subprogram.

```
DEF SEARCH()
INTERRUPT ON 21
LIN START_SEARCH
LIN END_SEARCH
$ADVANCE=0
...
END
```

Interrupt program:

LIN $POS_INT is the return motion to the position at which the interrupt was triggered. After LIN $POS_INT (in the example: …), the robot grips the part. RESUME causes the main program to be resumed after the part has been gripped. Without the RESUME statement, the subprogram SEARCH would be resumed after END.

```
DEF FOUND()
INTERRUPT OFF
BRAKE
LIN $POS_INT
...
RESUME
END
```

11.11 Path-related switching actions (=Trigger)

11.11.1 TRIGGER WHEN DISTANCE

Description

The Trigger triggers a defined statement. The statement refers to the start point or end point of the motion block in which the Trigger is situated in the program. The statement is executed parallel to the robot motion.

The statement can be shifted in time. It is then not triggered exactly at the start or end point, but brought forward or delayed.

Syntax

`TRIGGER WHEN DISTANCE=`*Position* `DELAY=`*Time* `DO` *Statement* `<PRIO=`*Priority*`>`

Explanation of the syntax

Element	Description
Position	Type: INT; variable or constant Defines the point at which the statement is triggered. Possible values: 0 or 1. ■ **0:** The statement is triggered at the start point of the motion block. If the start point is approximated, the statement is triggered at the end of the approximate positioning arc. ■ **1:** The statement is triggered at the end point. If the end point is approximated, the statement is triggered in the middle of the approximate positioning arc.
Time	Type: REAL; variable or constant; unit: ms Shifts the statement in time. Obligatory specification: if no time shift is desired, set *Time* = 0. The statement cannot be shifted freely in time. The shifts that are available depend on the value selected for *Position*: ■ **Position = 0 (start point)** In this case, the statement can only be triggered with a delay, i.e. it is only possible to select a positive value for *Time*. The statement can be delayed, **at most, as far as the end point**. If the end point is approximated, the statement can be delayed, at most, as far as the start of the approximate positioning arc. ■ **Position = 1 (end point)** In this case, a distinction must be made as to whether the end point is an exact positioning point or an approximate positioning point. ■ **Exact positioning point:** In this case, the statement can only be triggered earlier, i.e. it is only possible to select a negative value for *Time*. The statement can be brought forward, **at most, as far as the start point**. If the start point is approximated, the statement can be brought forward, at most, as far as the end of the approximate positioning arc. ■ **Approximate positioning point:** In this case, the statement can be triggered earlier or with a delay, i.e. it is possible to select a negative or positive value for *Time*. The statement can be shifted, **at most, as far as the start or end of the approximate positioning arc** of the end point.

Element	Description
Statement	Possible: ■ Assignment of a value to a variable ■ OUT statement ■ PULSE statement ■ Subprogram call. In this case, *Priority* must be specified.
Priority	Type: INT; variable or constant Priority of the trigger. Only relevant if *Statement* calls a subprogram, and then obligatory. Priorities 1, 2, 4 to 39 and 81 to 128 are available. Priorities 3 and 40 to 80 are reserved for cases in which the priority is automatically assigned by the system. If the priority is to be assigned automatically by the system, the following is programmed: `PRIO = -1`.. If several triggers call subprograms at the same time, the trigger with the highest priority is processed first, then the triggers of lower priority. 1 = highest priority.

> If a trigger calls a subprogram, it counts as an active interrupt until the subprogram has been executed. Up to 16 interrupts may be active at any one time.

System variables

Useful system variables for working with triggers:

System variable	Data type	Description
$DIST_NEXT	REAL	Distance to next point. Value is negative or 0. This system variable can be used for teaching triggers.
$DISTANCE	REAL	Overall length of current CP motion

Example 1

130 milliseconds after P_2, $OUT[8] is set to TRUE.

```
LIN P_2
TRIGGER WHEN DISTANCE=0 DELAY=130 DO $OUT[8]=TRUE
LIN P_3
```

Example 2

In the middle of the approximate positioning arc of P_5, the subprogram MY_SUBPROG with priority 5 is called.

```
PTP P_4
TRIGGER WHEN DISTANCE=1 DELAY=0 DO MY_SUBPROG() PRIO=5
PTP P_5 C_DIS
```

Example 3

Explanation of the diagram:

In the diagram, the approximate positions in which the Triggers would be triggered are indicated by arrows.

In addition to these points, the start, middle and end of each approximate positioning arc are indicated.

```
 1  DEF PROG()
 2  ...
 3  PTP P_0
 4  TRIGGER WHEN DISTANCE=0 DELAY=40 DO A=12
 5  ...
 6  TRIGGER WHEN DISTANCE=1 DELAY=-20 DO UP1() PRIO=10
 7  ...
 8  LIN P_1
 9  TRIGGER WHEN DISTANCE=0 DELAY=10 DO UP2(A) PRIO=5
10  ...
11  TRIGGER WHEN DISTANCE=1 DELAY=15 DO B=1
12  ...
13  LIN P_2 C_DIS
14  TRIGGER WHEN DISTANCE=0 DELAY=10 DO UP2(B) PRIO=12
15  ...
16  TRIGGER WHEN DISTANCE=1 DELAY=0 DO UP(A,B,C) PRIO=6
17  ...
18  LIN P_3 C_DIS
19  TRIGGER WHEN DISTANCE=0 DELAY=50 DO UP2(A) PRIO=4
20  ...
21  TRIGGER WHEN DISTANCE=1 DELAY=-80 DO A=0
22  ...
23  LIN P_4
24  ...
25  END
```

Line	Description
4	Switching range: 0 - 1
6	Switching range: 0 - 1
9	Switching range: 1 - 2*start
11	Switching range: 2*start - 2*end
14	Switching range: 2*end - 3*start
16	Switching range: 3*start - 3*end
19	Switching range: 3*end - 4
21	Switching range: 3*end - 4

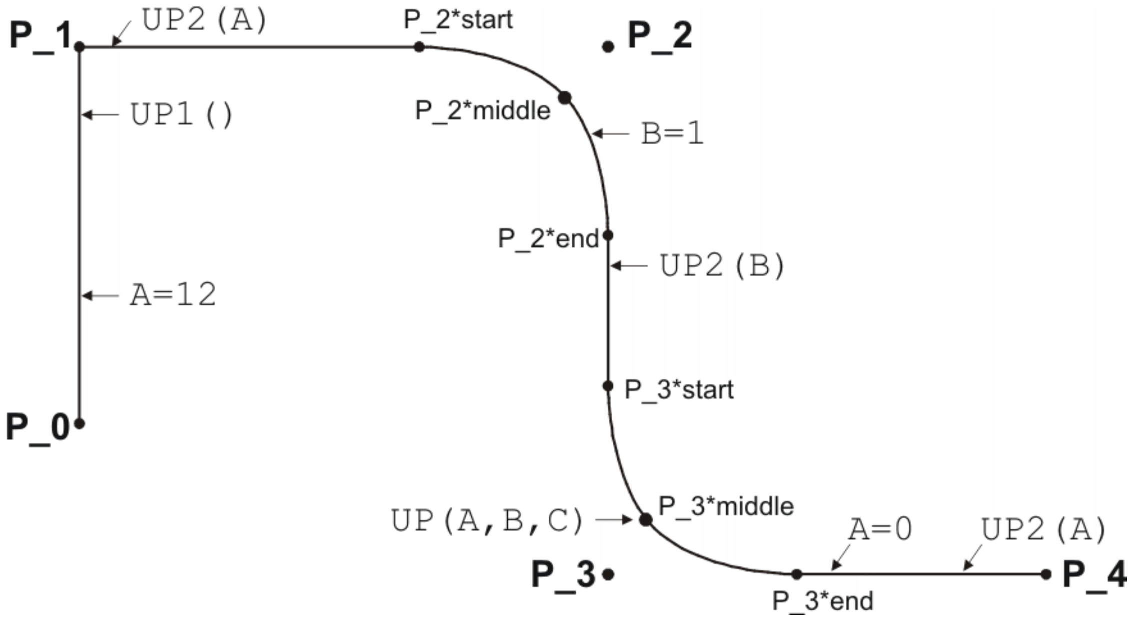


Fig. 11-13: Example of TRIGGER WHEN DISTANCE

11.11.2 TRIGGER WHEN PATH

Description

The Trigger triggers a defined statement. The statement refers to the end point of the motion block in which the Trigger is situated in the program. It is possible to shift the statement in time and/or space so that it is not triggered exactly at the end point, but before or after it.

The statement is executed parallel to the robot motion.

> ℹ The end point must be LIN or CIRC. It must not be PTP.

Syntax

`TRIGGER WHEN PATH =`*Distance* `DELAY=`*Time* `DO` *Statement* `<PRIO=`*Priority*`>`

Explanation of the syntax

Element	Description
Distance	Type: REAL; variable or constant; unit: mm Obligatory specification. If no shift in space is desired, set *Distance* = 0. If the statement is to be shifted in space, the desired distance from the end point must be specified here. If this end point is approximated, *Distance* is the distance to the position on the approximate positioning arc closest to the end point. ■ Positive value: shifts the statement towards the end of the motion. ■ Negative value: shifts the statement towards the start of the motion. The statement cannot be shifted freely. Maximum possible shift: see below, section “Switching range”.
Time	Type: REAL; variable or constant; unit: ms Obligatory specification. If no shift in time is desired, set *Time* = 0. If the statement is to be shifted in time (relative to PATH), the desired duration must be specified here. ■ Positive value: shifts the statement towards the end of the motion. ■ Negative value: shifts the statement towards the start of the motion. The statement cannot be shifted freely. Maximum possible shift: see below, section “Switching range”.

Element	Description
Statement	Possible: ■ Assignment of a value to a variable ■ OUT statement ■ PULSE statement ■ Subprogram call. In this case, *Priority* must be specified.
Priority	Type: INT; variable or constant Priority of the trigger. Only relevant if *Statement* calls a subprogram, and then obligatory. Priorities 1, 2, 4 to 39 and 81 to 128 are available. Priorities 3 and 40 to 80 are reserved for cases in which the priority is automatically assigned by the system. If the priority is to be assigned automatically by the system, the following is programmed: `PRIO = -1`. If several triggers call subprograms at the same time, the trigger with the highest priority is processed first, then the triggers of lower priority. 1 = highest priority.

> If a trigger calls a subprogram, it counts as an active interrupt until the subprogram has been executed. Up to 16 interrupts may be active at any one time.

System variables

Useful system variables for working with triggers:

System variable	Data type	Description
$DIST_NEXT	REAL	Distance to next point. Value is negative or 0. This system variable can be used for teaching triggers.
$DISTANCE	REAL	Overall length of current CP motion

Switching range

- Shift towards the end of the motion:

  A statement can be shifted, **at most, as far as the next exact positioning point** after TRIGGER WHEN PATH (skipping all approximate positioning points).

  In other words, if the end point is an exact positioning point, the statement cannot be shifted beyond the end point.

- Shift towards the start of the motion:

  A statement can be shifted, **at most, as far as the start point of the motion block** (i.e. as far as the last point before TRIGGER WHEN PATH).

  If the start point is an approximated LIN or CIRC point, the statement can be brought forward, at most, as far as the start of its approximate positioning arc.

  If the start point is an approximated PTP point, the statement can be brought forward, at most, as far as the end of its approximate positioning arc.

Example

```
LIN P_2 C_DIS
TRIGGER WHEN PATH = -20.0 DELAY= -10 DO $OUT[2]=TRUE
LIN P_3 C_DIS
LIN P_4 C_DIS
LIN P_5
```

In the diagram, the approximate position in which the $OUT[2]=TRUE statement would be triggered is indicated by an arrow.

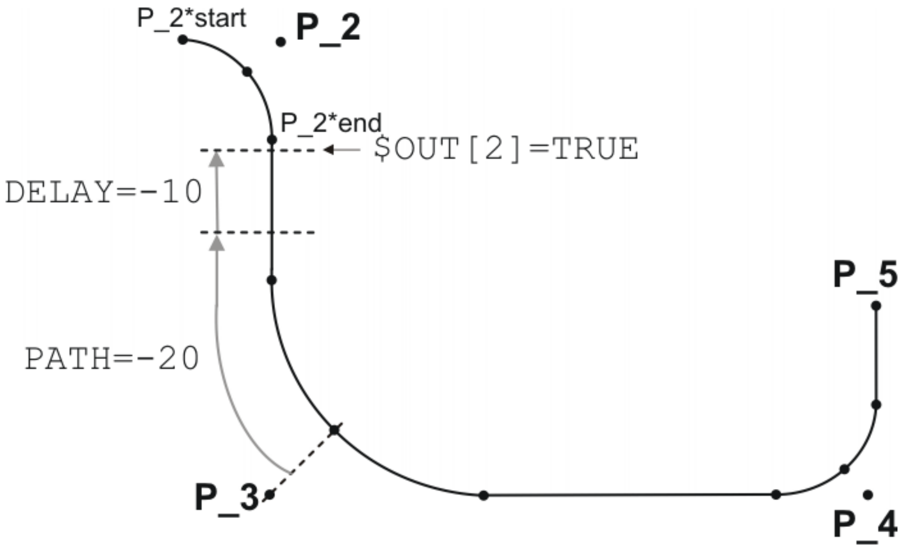


Fig. 11-14: Example of TRIGGER WHEN PATH

Switching range: P_2*start to P_5.

If P_2 were not approximated, the switching range would be P_2 to P_5.

The switching range goes to P_5 because P_5 is the next exact positioning point after the TRIGGER statement. If P_3 were not approximated, the switching range would be P_2 to P_3, as P_3 is the next exact positioning point in the program after the Trigger statement.

11.11.3 TRIGGER WHEN PATH (for SPLINE)

Description

This trigger can only be used in spline blocks.

The Trigger triggers a defined statement. The statement refers to the start point or end point of the motion block in which the Trigger is situated in the program. The statement is executed parallel to the robot motion.

The statement can be shifted in time and/or space. It is then not triggered exactly at the start or end point, but beforehand or afterwards.

- In the negative direction, the statement can be shifted, at most, as far as the first point before the spline block.
- In the positive direction, the statement can be shifted as far as the last point of the spline block if this is an exact positioning point. If this point is approximated, the statement can be shifted, at most, as far as the next exact positioning point.

Syntax

`TRIGGER WHEN PATH =` *Distance* `<ONSTART> DELAY =` *Time* `DO` *Statement* `<PRIO =` *Priority*`>`

Functions

`PATH` and `DELAY` can call functions. The following restrictions apply here:

- The KRL program containing the function must have the attribute **Hidden**. (>>> 8.1.2 "Displaying or modifying file properties" Page 220)
- The function must be globally valid.
- The functions may only contain the following statements or elements:
  - Value assignments
  - IF statements
  - Comments
  - Blank lines
  - RETURN

- Read system variable
- Call predefined KRL function

Explanation of the syntax

Element	Description
Distance	Type: REAL; variable, constant or function; unit: mm Obligatory specification. If no shift in space is desired, set *Distance* = 0. If the statement is to be shifted in space, the desired distance from the start or end point must be specified here. ■ Positive value: shifts the statement towards the end of the motion. ■ Negative value: shifts the statement towards the start of the motion.
`ONSTART`	the PATH value refers to the start point. Without ONSTART: the PATH value refers to the end point. (>>> 11.11.3.1 "Spline: trigger point in the case of approximate positioning" Page 360)
Time	Type: REAL; variable, constant or function; unit: ms Obligatory specification. If no shift in time is desired, set *Time* = 0. If the statement is to be shifted in time (relative to PATH), the desired duration must be specified here. ■ Positive value: shifts the statement towards the end of the motion. Maximum: 1,000 ms ■ Negative value: shifts the statement towards the start of the motion.
Statement	Possible: ■ Assignment of a value to a variable ■ OUT statement ■ PULSE statement ■ subprogram call. In this case, *Priority* must be specified.
Priority	Type: INT; variable or constant Priority of the trigger. Only relevant if *Statement* calls a subprogram, and then obligatory. Priorities 1, 2, 4 to 39 and 81 to 128 are available. Priorities 3 and 40 to 80 are reserved for cases in which the priority is automatically assigned by the system. If the priority is to be assigned automatically by the system, the following is programmed: `PRIO = -1`. If several triggers call subprograms at the same time, the trigger with the highest priority is processed first, then the triggers of lower priority. 1 = highest priority.

If a trigger calls a subprogram, it counts as an active interrupt until the subprogram has been executed. Up to 16 interrupts may be active at any one time.

System variables Useful system variables for working with triggers:

System variable	Data type	Description
$DIST_NEXT	REAL	Distance to next point. Value is negative or 0. This system variable can be used for teaching triggers.
$DISTANCE	REAL	Overall length of current CP motion

Example

```
1  PTP P0
2  SPLINE
3   SPL P1
4   SPL P2
5   SPL P3
6   SPL P4
7   TRIGGER WHEN PATH=0 ONSTART DELAY=10 DO $OUT[5]=TRUE
8   SCIRC P5, P6
9   SPL P7
10  TRIGGER WHEN PATH=-20.0 DELAY=0 DO SUBPR_2() PRIO=-1
11  SLIN P8
12 ENDSPLINE
```

The Trigger in line 10 would have the same result if it was positioned directly before the spline block (i.e. between line 1 and line 2). In both cases, it refers to the last point of the spline motion: P8.

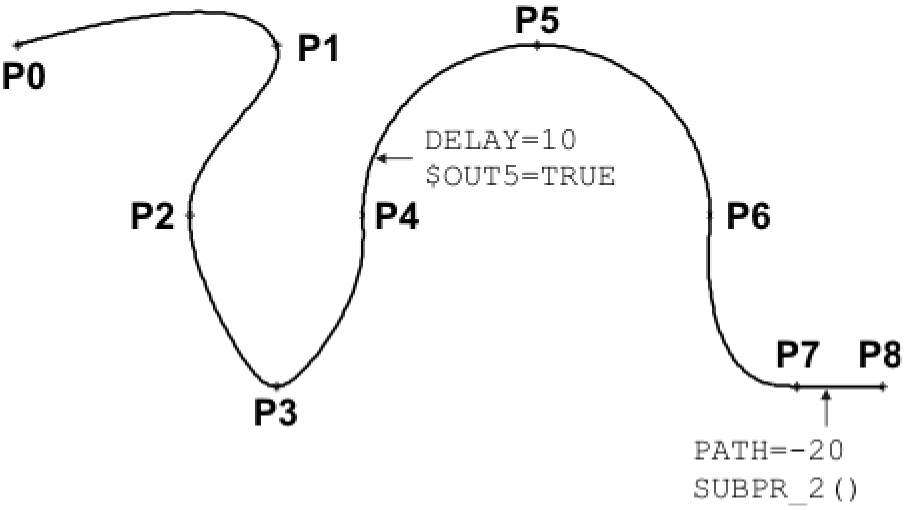


Fig. 11-15: Example of TRIGGER WHEN PATH (for spline)

11.11.3.1 Spline: trigger point in the case of approximate positioning

The point at which the trigger is triggered is the end point of the motion block in which the trigger is situated. If ONSTART is used, it is the start point.

If this end or start point is approximated, the trigger point is transferred onto the approximate positioning arc by a distance corresponding to the approximation distance.

- The trigger point is not generally in the middle of the approximate positioning arc.
- Triggers that are triggered at the same point in the case of exact positioning are triggered at different points in the case of approximate positioning.

  The trigger that refers to the end point is triggered later than the trigger that refers to the start point

Example

Trigger 1 and trigger 2 would be triggered at the same point if P3 was not approximated.

- Trigger 1 refers to the end point of the block in which it is situated, i.e. P3.
- Trigger 2 (= with ONSTART) refers to the start point of the block in which it is situated, i.e. also P3.

Trigger 1:

```
SPLINE
  ...
  SLIN P2
  TRIGGER WHEN PATH=0 DELAY=0 DO ...
  SLIN P3
ENDSPLINE C_DIS
SPLINE
  SLIN4
  ...
ENDSPLINE
```

Trigger 2:

```
SPLINE
  ...
  SLIN P2
  SLIN P3
ENDSPLINE C_DIS
SPLINE
  TRIGGER WHEN PATH=0 ONSTART DELAY=0 DO ...
  SLIN4
  ...
ENDSPLINE
```

P3 is approximated, however. It is transferred onto the approximate positioning arc by a distance corresponding to the approximation distance (= P3').

Trigger 1:

Trigger 1 is situated in the block P2 --> P3 and refers to the end point. The robot controller calculates how far the distance would be from the start of the approximate positioning arc to the end point with exact positioning. This distance is then applied to the approximate positioning arc.

The distance **$P_{StartApprox}$ --> P3'** is the same as **$P_{StartApprox}$ --> P3**.

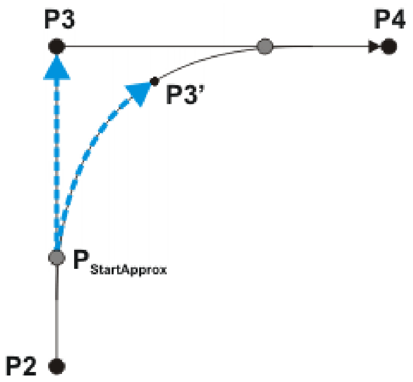


Fig. 11-16: Trigger 1: Trigger point in the case of approximate positioning

P3	Trigger point in the case of exact positioning
P3'	Trigger point in the case of approximate positioning
$P_{StartApprox}$	Start of the approximate positioning arc
$P_{EndApprox}$	End of the approximate positioning arc

Trigger 2:

Trigger 2 is situated in the block P3 --> P4 and refers to the start point. The robot controller calculates how far the distance would be from the end of the approximate positioning arc back to the start point with exact positioning. This distance is then applied to the approximate positioning arc.

The distance $\mathbf{P_{EndApprox}}$ --> **P3'** is the same as $\mathbf{P_{EndApprox}}$ --> **P3**.

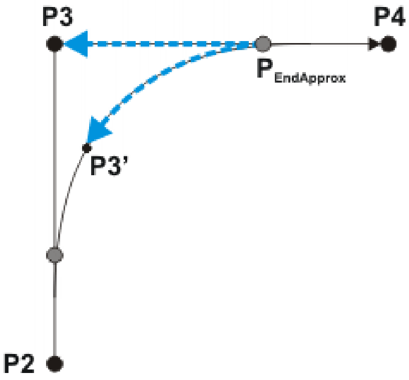


Fig. 11-17: Trigger 2: Trigger point in the case of approximate positioning

11.12 Communication

Information about the following statements is contained in the Expert documentation CREAD/CWRITE.

- CAST_FROM
- CAST_TO
- CCLOSE
- CHANNEL
- CIOCTL
- COPEN
- CREAD
- CWRITE
- SREAD
- SWRITE

11.13 System functions

11.13.1 VARSTATE()

Description

VARSTATE() can be used to monitor the state of a variable.

VARSTATE() is a function with a return value of type VAR_STATE. VAR_STATE is an enumeration type that is defined as follows in the system:

```
ENUM VAR_STATE DECLARED, INITIALIZED, UNKNOWN
```

VARSTATE is defined as follows in the system:

```
VAR_STATE VARSTATE(CHAR VAR_STR[80]:IN)
```

Example 1

```
DEF PROG1()
INT MYVAR
...
IF VARSTATE("MYVAR")==#UNKNOWN THEN
   $OUT[11]=TRUE
ENDIF
...
IF VARSTATE("MYVAR")==#DECLARED THEN
   $OUT[12]=TRUE
ENDIF
...
IF VARSTATE("ANYVAR")==#UNKNOWN THEN
   $OUT[13]=TRUE
ENDIF
...
MYVAR=9
...
IF VARSTATE("MYVAR")==#DECLARED THEN
   $OUT[14]=TRUE
ENDIF
...
IF VARSTATE("MYVAR")==#INITIALIZED THEN
   $OUT[15]=TRUE
ENDIF
...
END
```

Explanation of the state monitoring:

- The first IF condition is false, as MYVAR has already been declared. Output 11 is not set.
- The second IF condition is true, as MYVAR has been declared. Output 12 is set.
- The third IF condition is true, on the condition that there is also no variable with the name ANYVAR in $CONFIG.DAT. Output 13 is set.
- The fourth IF condition is false, as MYVAR has not only been declared, but has also already been initialized here. Output 14 is not set.
- The fifth IF condition is true, as MYVAR has been initialized. Output 15 is set.

Example 2

```
DEF PROG2()
INT MYVAR
INT YOURVAR
DECL VAR_STATE STATUS
...
STATUS=VARSTATE("MYVAR")
UP()
...
STATUS=VARSTATE("YOURVAR")
UP()
...
END
```

```
DEF UP()
...
IF VARSTATE("STATUS")==#DECLARED THEN
   $OUT[100]=TRUE
ENDIF
...
END
```

Explanation of the state monitoring:

In this example, the state is monitored indirectly, i.e. via an additional variable. The additional variable must be of type VAR_STATE. The keyword DECL must not be omitted in the declaration. The name of the additional variable may be freely selected. In this example it is STATUS.

11.14 Editing string variables

Various functions are available for editing string variables. The functions can be used in SRC files, in SUB files and in the variable correction function.

11.14.1 String variable length in the declaration

Description

The function `StrDeclLen()` determines the length of a string variable according to its declaration in the declaration section of a program.

Syntax

Length = `StrDeclLen(`*StrVar[]*`)`

Explanation of the syntax

Element	Description
Length	Type: INT Variable for the return value. Return value: Length of the string variable as declared in the declaration section
StrVar[]	Type: CHAR array String variable whose length is to be determined Since the string variable StrVar[ ] is an array of type CHAR, individual characters and constants are not permissible for length determination.

Example

```
1  CHAR ProName[24]
2  INT StrLength
   …
3  StrLength = StrDeclLen(ProName)
4  StrLength = StrDeclLen($Trace.Name[ ])
```

Line	Description
3	StrLength = 24
4	StrLength = 7

11.14.2 String variable length after initialization

Description

The function `StrLen()` determines the length of the character string of a string variable as defined in the initialization section of the program.

Syntax

Length = `StrLen(`*StrVar*`)`

Explanation of the syntax

Element	Description
Length	Type: INT Variable for the return value. Return value: Number of characters currently assigned to the string variable
StrVar	Type: CHAR Character string or variable whose length is to be determined

Example

```
1  CHAR PartA[50]
2  INT AB
   …
3  PartA[] = "This is an example"
4  AB = StrLen(PartA[])
```

Line	Description
4	AB = 18

11.14.3 Deleting the contents of a string variable

Description

The function `StrClear()` deletes the contents of a string variable.

Syntax

Result = `StrClear(`*StrVar[]*`)`

Explanation of the syntax

Element	Description
Result	Type: BOOL Variable for the return value. Return value: ■ The contents of the string variable have been deleted: TRUE ■ The contents of the string variable have not been deleted: FALSE
StrVar[]	Type: CHAR array Variable whose character string is to be deleted

Example

```
IF (NOT StrClear($Loop_Msg[])) THEN
HALT
ENDIF
```

The function can be used within IF branches without the return value being explicitly assigned to a variable. This applies to all functions for editing string variables.

11.14.4 Extending a string variable

Description

The function `StrAdd()` can be used to expand a string variable with the contents of another string variable.

Syntax

Sum = `StrAdd(`*StrDest[]*`,` *StrToAdd[]*`)`

Explanation of the syntax

Element	Description
Sum	Type: INT Variable for the return value. Return value: Sum of StrDest[ ] and StrToAdd[ ] If the sum is longer than the previously defined length of *StrDest[ ]*, the return value is 0. This is also the case if the sum is greater than 470 characters.
StrDest[]	Type: CHAR array The string variable to be extended Since the string variable StrDest[ ] is an array of type CHAR, individual characters and constants are not permissible.
StrToAdd[]	Type: CHAR array The character string by which the variable is to be extended

Example

```
DECL CHAR A[50], B[50]
INT AB, AC
…
A[] = "This is an "
B[] = "example"
AB = StrAdd(A[],B[])
```

Line	Description
5	A[ ] = “This is an example” AB = 18

11.14.5 Searching a string variable

Description

The function `StrFind()` can be used to search a string variable for a character string.

Syntax

Result = `StrFind(`*StartAt, StrVar[], StrFind[], CaseSens*`)`

Explanation of the syntax

Element	Description
Result	Type: INT Variable for the return value. Return value: Position of the first character found. If no character is found, the return value is 0.
StartAt	Type: INT The search is started from this position.
StrVar[]	Type: CHAR array The string variable to be searched
StrFind[]	Type: CHAR array The character string that is being looked for.
CaseSens	■ #CASE_SENS: Upper and lower case are taken into consideration. ■ #NOT_CASE_SENS: Upper and lower case are ignored.

Example

```
1  DECL CHAR A[5]
2  INT B
3  A[]="ABCDE"
4  B = StrFind(1, A[], "AC", #CASE_SENS)
5  B = StrFind(1, A[], "a", #NOT_CASE_SENS)
6  B = StrFind(1, A[], "BC", #Case_Sens)
7  B = StrFind(1, A[], "bc", #NOT_CASE_SENS)
```

Line	Description
4	B = 0
5	B = 1
6	B = 2
7	B = 2

11.14.6 Comparing the contents of string variables

Description

The function `StrComp()` can be used to compare two string variables.

Syntax

Comp = `StrComp(`*StrComp1[], StrComp2[], CaseSens*`)`

Explanation of the syntax

Element	Description
Comp	Type: BOOL Variable for the return value. Return value: ■ The character strings match: TRUE ■ The character strings do not match: FALSE
StrComp1[]	Type: CHAR array String variable that is compared with StrComp2[].
StrComp2[]	Type: CHAR array String variable that is compared with StrComp1[].
CaseSens	■ #CASE_SENS: Upper and lower case are taken into consideration. ■ #NOT_CASE_SENS: Upper and lower case are ignored.

Example

```
1  DECL CHAR A[5]
2  BOOL B
3  A[]="ABCDE"
4  B = StrComp(A[], "ABCDE", #CASE_SENS)
5  B = StrComp(A[], "abcde", #NOT_CASE_SENS)
6  B = StrComp(A[], "abcd", #NOT_CASE_SENS)
7  B = StrComp(A[], "acbde", #NOT_CASE_SENS)
```

Line	Description
4	B = TRUE
5	B = TRUE
6	B = FALSE
7	B = FALSE

11.14.7 Copying a string variable

Description

The function StrCopy() can be used to copy the contents of a string variable to another string variable.

Syntax

Copy = StrCopy(*StrDest[], StrSource[]*)

Explanation of the syntax

Element	Description
Copy	Type: BOOL Variable for the return value. Return value: ■ The string variable was copied successfully: TRUE ■ The string variable was not copied: FALSE
StrDest[]	Type: CHAR array The character string is copied to this string variable. Since StrDest[ ] is an array of type CHAR, individual characters and constants are not permissible.
StrSource[]	Type: CHAR array The contents of this string variable are copied.

Example

```
DECL CHAR A[25], B[25]
DECL BOOL C
A[] = ""
B[] = "Example"
C = StrCopy(A[], B[])
```

Line	Description
5	A[ ] = “Example” C = TRUE

12 Submit interpreter

12.1 Function of the Submit interpreter

Function

2 tasks run in parallel on the robot controller:

- Robot interpreter

  The motion program runs in the robot interpreter.
- Submit interpreter

  A SUB program runs in the Submit interpreter.

  A SUB program can perform operator control or monitoring tasks. Examples: monitoring of safety equipment; monitoring of a cooling circuit.

  This means that no PLC is required for smaller applications, as the robot controller can perform such tasks by itself.

The Submit interpreter starts automatically when the robot controller is switched on. The program defined in the file KRC/STEU/MADA/$custom.dat is started. By default, this is SPS.SUB.

```
$PRO_I_O[]="/R1/SPS()"
```

The Submit interpreter can be stopped or deselected manually and can also be restarted. It is also possible to start a SUB program other than the one entered in $custom.dat.

(>>> 12.2 "Manually stopping or deselecting the Submit interpreter" Page 370)

(>>> 12.3 "Manually starting the Submit interpreter" Page 370)

SUB programs are always files with the extension *.SUB. The program SPS.SUB can be modified and new SUB programs can be created.

(>>> 12.4 "Modifying the program SPS.SUB" Page 370)

(>>> 12.5 "Creating a new SUB program" Page 371)

⚠ **WARNING** The Submit interpreter must not be used for time-critical applications! A PLC must be used in such cases. Reasons:

- The Submit interpreter shares system resources with the robot interpreter, which has the higher priority. The Submit interpreter is thus not executed at the robot controller’s interpolation cycle rate of 12 ms. Furthermore, the runtime of the Submit interpreter is irregular.
- The runtime of the Submit interpreter is influenced by the number of lines in the SUB program. Even comment lines and blank lines have an effect.

If a system file, e.g. $config.dat or $custom.dat, is modified in such a way that errors are introduced, the Submit interpreter is automatically deselected. Once the error in the system file has been rectified, the Submit interpreter must be reselected manually.

Display

The program SPS.SUB can be found in the “System” folder in the Navigator. SUB files are only visible in the Navigator in the user group “Expert”.

By default, the execution of a selected SUB program is not displayed. This can be changed using the system variable $INTERPRETER. The SUB program can only be displayed, however, if a motion program is selected at the same time.

Value of $INTERPRETER	Description
1	The selected motion program is displayed in the editor (default).
0	The selected SUB program is displayed in the editor.

12.2 Manually stopping or deselecting the Submit interpreter

Procedure

- Select the menu sequence **Configure** > **SUBMIT Interpreter** > **Stop** or **Cancel**.

Description

Command	Description
Stop	The Submit interpreter is stopped. When it is restarted, the SUB program is resumed at the point at which it was stopped.
Deselect	The Submit interpreter is deselected. A different SUB program can now be selected.

Once the Submit interpreter has been stopped or deselected, the corresponding icon in the status bar is red or gray.

Icon	Color	Description
S	Red	Submit interpreter has been stopped.
S	Gray	Submit interpreter is deselected.

12.3 Manually starting the Submit interpreter

Procedure

- Select the menu sequence **Configure** > **SUBMIT interpreter** > **Start/Select**.

If the Submit interpreter is deselected, the command **Start/Select** selects the SUB program defined in the file $custom.dat.

If the Submit interpreter has been stopped, the command **Start/Select** resumes the selected SUB program at the point at which it was stopped.

Alternative procedure

A SUB program can also be selected directly. Precondition: the Submit interpreter is deselected.

1. Select the program in the Navigator.
2. Press the **Select** softkey. The program starts automatically.

Description

Once the Submit interpreter has been started, the corresponding icon in the status bar is green.

Icon	Color	Description
S	Green	Submit interpreter is running.

12.4 Modifying the program SPS.SUB

Precondition

- The program SPS.SUB is not selected or has been stopped.
- User group “Expert”

Procedure

1. Select the program SPS.SUB in the Navigator and press the **Open** softkey.

2. Enter changes.
   - Enter initializations in the USER INIT Fold. This Fold is located in the INIT Fold.

```
USER INIT
  ; Please insert user defined initialization commands
```

   - Enter all other changes in the USER PLC Fold.

```
USER PLC
    ; Make your modifications here
```

3. Press the **Close** softkey. Respond to the request for confirmation asking whether the changes should be saved by pressing the **Yes** softkey.
4. The program SPS.SUB can now be started via the menu sequence **Configure** > **SUBMIT interpreter** > **Start**.

Description

Structure of the program SPS.SUB:

```
 1  DEF SPS()
 2
 3  DECLARATIONS
 4
 5  INI
 6
 7  LOOP
 8  ...
 9  GRIPPERTECH PLC
11
12  USER PLC
13
14  ENDLOOP
```

Line	Description
3	Declaration section
5	Initialization section. For statements that are only to be executed once after the system has booted.
7, 14	LOOP statement. For programs that are to run continuously in the background.
9	Fold with statements for a technology package (example). The Folds that are present depend on what technologies are installed.
12	Fold for user-specific adaptations

12.5 Creating a new SUB program

Precondition

- “Expert” user group

Procedure

1. In the Navigator, use the UP and DOWN arrow keys to select the folder in which the program is to be created.

   Closed folders can be opened by pressing the Enter key.
2. Move to the file list by pressing the RIGHT arrow button.
3. Press the **New** softkey.

   The **Template selection** window is opened.
4. Select the template **Submit** or **Expert Submit** and press the softkey **OK**.
5. Enter a name for the program and press the softkey **OK**.

Description

Newly created SUB programs contain no Folds for the installed technology packages.

These Folds are present by default in the program SPS.SUB. If the program SPS.SUB is to be replaced with a new SUB program, these Folds must be copied into the new program. The technology packages are otherwise no longer fully operational.

- The template **Submit** generates a SUB file with the following structure:

```
1  DECLARATIONS
2  INI
3
4  LOOP
5  USER PLC
6  ENDLOOP
7  USER SUBROUTINE
```

Line	Description
1	Declaration section
2	Initialization section. For statements that are only to be executed once after the system has booted.
4, 5, 6	LOOP statement containing the Fold USER PLC. The Fold is for programs that are to run continuously in the background.
7	For user-specific subroutines

- The template **Expert Submit** generates an empty SUB file. With this template, everything has to be programmed by the user.

Use a LOOP statement when programming. SUB programs without a LOOP statement are only executed once by the Submit interpreter. It is then automatically deselected.

12.6 Programming

KRL code

Almost all KRL instructions can be used in a SUB program. The following statements are not possible, however:

- Statements for robot motions

  Robot motions can only be interpreted by the robot interpreter. For this reason, SRC programs containing motion commands cannot be called as subprograms from a SUB program.
- Statements referring to robot motions

  These include BRAKE and all TRIGGER statements.

WAIT statements or wait loops stop the cycle.

System variables

The Submit interpreter has read-access to all system variables and write-access to many of them. Access works even if the system variables are being used in parallel by a motion program.

If a system variable to which the Submit interpreter does not have write-access is modified in a SUB program, an error message is generated when the program is started and the Submit interpreter stops.

System variables that are frequently required in SUB programs:

$MODE_OP = *Value*	
Value	Description
#T1	Robot controller is in T1 mode.
#T2	Robot controller is in T2 mode.
#AUT	Robot controller is in Automatic mode.
#EX	Robot controller is in Automatic External mode.
#INVALID	Robot controller has no defined state.

$OV_PRO = *Value*		
Element	Data type	Description
Value (%)	INT	Program override value

Example:

If the programmed velocity is not reached, output 2 is set to FALSE.

```
…
IF (($MODE_OP == #T1) OR ($OV_PRO < 100)) THEN
$OUT[2] = FALSE
ENDIF
…
```

Inputs/outputs

The Submit interpreter can access the inputs and outputs of the robot controller.

⚠ WARNING No check is made to see if the robot interpreter and Submit interpreter are accessing the same output simultaneously, as this may even be desired in certain cases.
The user must therefore carefully check the assignment of the outputs. Otherwise, unexpected output signals may be generated, e.g. in safety equipment. Death, serious physical injuries or major damage to property may result.

Subprograms

Other programs can be called as subprograms in a SUB program. The following are possible:

- Other SUB programs
- SRC programs without statements for robot motions

Example:

CELL.SRC can be called from the program SPS.SUB with a CWRITE statement and RUN. The call only takes effect in the case of a cold start.

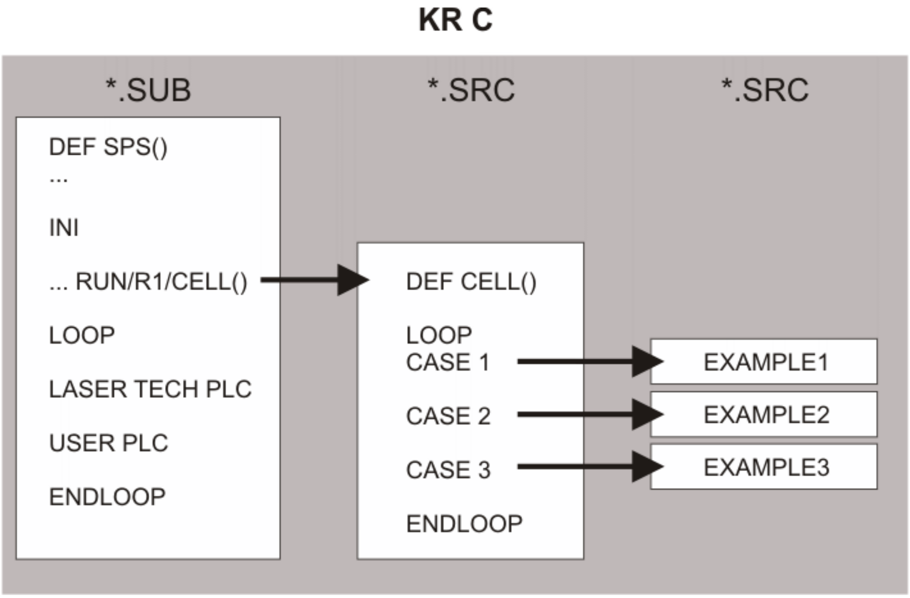


Fig. 12-1: SPS.SUB calls CELL.SRC

Further information about the program CELL.SRC can be found in this documentation.
(>>> 6.23.1 "Configuring CELL.SRC" Page 174)
Further information about CWRITE statements can be found in the Expert documentation CREAD/CWRITE.

Communication

The flags of the robot controller can be used to enable the exchange of binary information between a running motion program and a SUB program. A flag is set by the Submit interpreter and read by the robot interpreter.

13 Diagnosis

13.1 Logbook

13.1.1 Displaying the logbook

The operator actions on the KCP are automatically logged. The command **Logbook** displays the logbook.

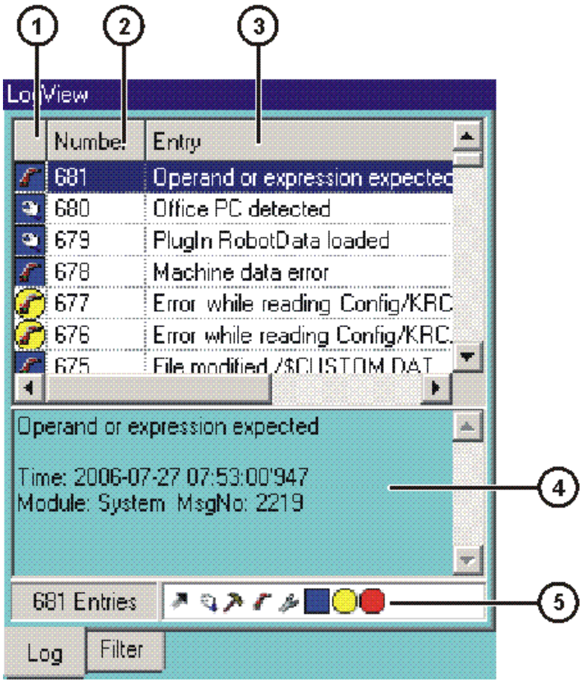
Procedure

- Select the menu sequence **Monitor** > **Diagnosis** > **Logbook** and select the desired log.

The following tabs are available:

- Log (>>> 13.1.2 "“Log” tab" Page 375)
- Filter (>>> 13.1.3 "“Filter” tab" Page 376)

13.1.2 “Log” tab



```
Number  Entry
681     Operand or expression expected
680     Office PC detected
679     PlugIn RobotData loaded
678     Machine data error
677     Error while reading Config/KRC
676     Error while reading Config/KRC.
675     File modified /$CUSTOM.DAT
```

```
Operand or expression expected

Time: 2006-07-27 07:53:00'947
Module: System  MsgNo: 2219
```

```
681 Entries
```

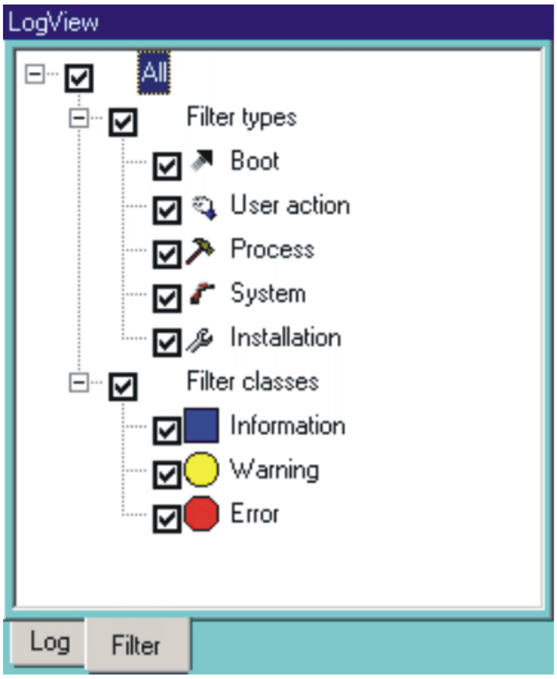
Fig. 13-1: Logbook, Log tab

Item	Description
1	Type of log event Example : Filter type "Note" + filter class "System" = note originated by the kernel system of the robot. The individual filter types and filter classes are listed on the **Filter** tab.
2	Log event number
3	Brief description of the log event
4	Detailed description of the selected log event
5	Indication of the active filter

The following softkeys are available:

Softkey	Description
Tab +	Toggles between the **Log** and **Filter** tabs.
Export	Exports the log data as a text file. Default path: C:\KRC\ROBOTER LOG\LOGBUCH.TXT
Refresh	Refreshes the log display.
Page +/ Page -	Scrolls up/down in the list of log events.

13.1.3 “Filter” tab



LogView

- ☑ All
  - ☑ Filter types
    - ☑ Boot
    - ☑ User action
    - ☑ Process
    - ☑ System
    - ☑ Installation
  - ☑ Filter classes
    - ☑ ■ Information
    - ☑ ● Warning
    - ☑ ● Error

Fig. 13-2: Logbook, Filter tab

The following softkeys are available:

Softkey	Description
Tab +	Toggles between the **Log** and **Filter** tabs.
Mark	Activates or deactivates the selected filter.

13.2 Displaying the caller stack

This function displays the data for the process pointer ($PRO_IP).

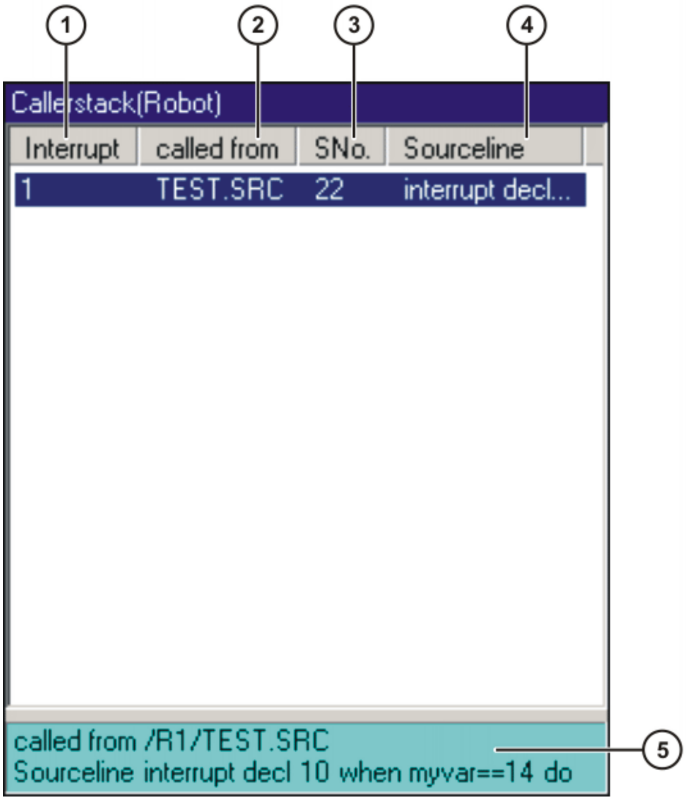
Precondition

- User group "Expert"
- Program is selected.

Procedure

- Select the menu sequence **Monitor** > **Diagnosis** > **Caller Stack**.

Description



Callerstack(Robot)

Interrupt	called from	SNo.	Sourceline
1	TEST.SRC	22	interrupt decl...

called from /R1/TEST.SRC
Sourceline interrupt decl 10 when myvar==14 do

Fig. 13-3: Caller Stack window

Item	Description
1	■ **None**: Call not initiated by interrupt ■ **[No.]**: Call initiated by interrupt with the number [No.]
2	This file contains the call.
3	The program line with this number contains the call. Preconditions in the program for the correct line to be determined using the number: ■ DEF line is displayed. ■ Detail view (ASCII mode) is activated. ■ All Folds are open.
4	Source line
5	Detailed information about the entry selected in the list

13.3 Displaying interrupts

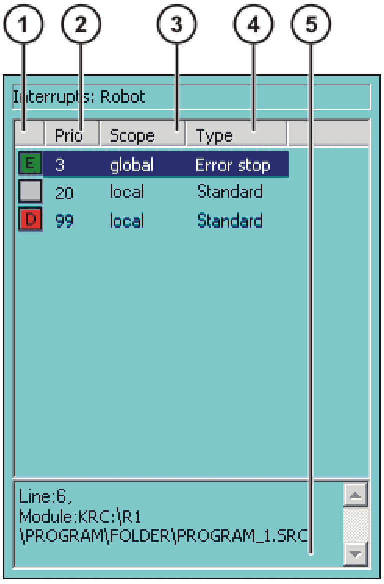
Precondition

- User group “Expert”

Procedure

- Select the menu sequence **Monitor** > **Diagnosis** > **Interrupts**.

Description



Interrupts: Robot

	Prio	Scope	Type
E	3	global	Error stop
□	20	local	Standard
D	99	local	Standard

Line:6,
Module:KRC:\R1
\PROGRAM\FOLDER\PROGRAM_1.SRC

Fig. 13-4: Interrupts

Item	Description
1	Status of the interrupt - E Interrupt ON or ENABLE - D Interrupt DISABLE - □ Interrupt OFF or not activated
2	Number/priority of the interrupt
3	Validity range of the interrupt: global or local
4	Type of interrupt, dependent on the defined event in the interrupt declaration - **Standard**: e.g. $IN[1…4096] - **Error stop**: $STOPMESS - **EMERGENCY STOP**: $ALARM_STOP - **Measurement (Fast Measurement)**: $MEAS_PULSE[1…5] - **Trigger**: Trigger subprogram
5	Module and program line of the interrupt declaration

The following softkeys are available:

Softkey	Description
Submit/Robot	Toggles between the displays for robot interrupts and Submit interrupts.
Refresh	Refreshes the display.

13.4 Oscilloscope

Overview

The oscilloscope is an important diagnostic tool during start-up of the industrial robot and during troubleshooting. It is also used for optimization of the machine data.

The oscilloscope can be used to record different variables with the program running, e.g. actual current, setpoint current, states of inputs and outputs, etc.

The recording (trace) can then be displayed. The variables are displayed as colored curves. 8 colors are available for this. If more than 8 variables have been recorded, individual variables can be removed from the display and others displayed.

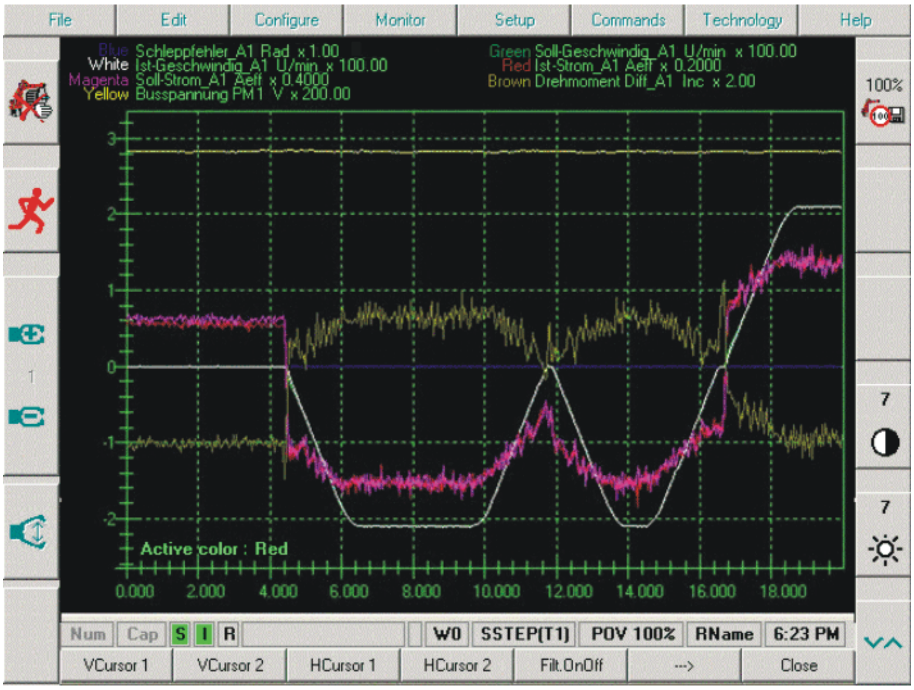

File
Edit
Configure
Monitor
Setup
Commands
Technology
Help
Blue Schleppfehler_A1 Rad x 1.00
White Ist-Geschwindig_A1 U/min x 100.00
Magenta Soll-Strom_A1 Aeff x 0.4000
Yellow Busspannung PM1 V x 200.00
Green Soll-Geschwindig_A1 U/min x 100.00
Red Ist-Strom_A1 Aeff x 0.2000
Brown Drehmoment Diff_A1 Inc x 2.00
3
2
1
0
-1
-2
Active color : Red
0.000
2.000
4.000
6.000
8.000
10.000
12.000
14.000
16.000
18.000
100%
7
7
Num
Cap
S
I
R
W0
SSTEP(T1)
POV 100%
RName
6:23 PM
VCursor 1
VCursor 2
HCursor 1
HCursor 2
Filt.OnOff
-->
Close

Fig. 13-5: Oscilloscope

13.4.1 Configuring and starting the oscilloscope

Description

During configuration, the data to be recorded by the oscilloscope are specified. The robot controller saves the trace recording as a TRC file in the directory C:\KRC\ROBOTER\TRACE.

Procedure

1. Select the menu sequence **Monitor** > **Diagnosis** > **Oscilloscope** > **Configure**.
2. Specify the basic data to be recorded.

   (>>> 13.4.1.1 "Main window" Page 380)

3. If DSE data are to be recorded:

   Press the **DSE Table** softkey and configure the DSE recording.

   (>>> 13.4.1.2 "DSE table" Page 382)

4. If I/O data are to be recorded:

   Press the **I/O Table** softkey and configure the I/O recording.

   (>>> 13.4.1.3 "I/O table" Page 383)

5. Press the **Main menu** softkey.
6. Optionally: Press the **Save** softkey to save the current configuration.
7. Start the KRL program.
8. Either: Press the **Start** softkey. The recording is started in accordance with the defined trigger.

   Or: Press the **Trigger** softkey. The recording starts immediately.

The **Trace Status** box jumps from #T_END to either #T_WAIT or #TRIGGERED.

The recording is ended when the **Trace Status** box displays the value #T_END again.

13.4.1.1 Main window

Description

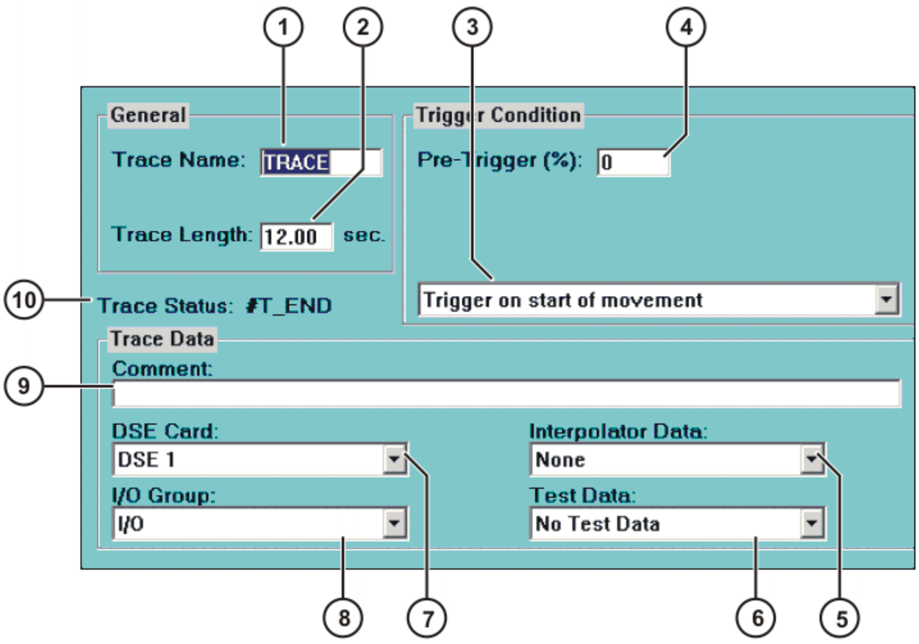


Fig. 13-6: Oscilloscope configuration – main window

Item	Description
1	Name of the TRC file (max. 7 characters). The robot controller appends a number to the end of the name, indicating what data have been recorded. ■ **1**: DSE data ■ **3**: I/O data ■ **4**: Interpolator data ■ **5**: Test data If, for example, DSE data and interpolator data have been recorded, the robot controller creates 2 TRC files: one with the number 1 and one with the number 4. **Note:** If a file of the same name already exists, it is overwritten by the new file without a request for confirmation!
2	Duration of the recording. Only whole numbers can be entered. Maximum value: 999 s.
3	The trigger controls the point in time at which the recording of data is started. (>>> "Trigger" Page 381) The oscilloscope actually starts recording data as soon as the **Start** or **Trigger** softkey is pressed. The trigger merely controls which time phase of the recording is then displayed in the TRC file.

Item	Description
4	The position of the time phase displayed in the TRC file relative to the trigger. The % value refers to the duration of the recording. Examples: - **0%**: The displayed time phase starts at the trigger. - **30%**: 30% of the displayed time phase comes before the trigger, 70% after the trigger. - **100%**: The displayed time phase ends at the trigger.
5	This box is only available in the user group “Expert”. The interpolator data include the Cartesian position of the robot (X, Y, Z, A, B, C), the path velocity and the geometric angles of axes A1 to A12. - **None**: No interpolator data are recorded. - **Command**: The setpoint interpolator data are recorded. - **Actual**: The actual interpolator data are recorded.
6	This function is intended for KUKA service personnel only. This box is only available in the user group “Expert”.
7	- **No DSE data**: No DSE data are recorded. - **DSE 1**: The DSE data of axes A1 to A8 are recorded. - **DSE 2**: The DSE data of axes A7 to A12 are recorded. DSE data are recorded every 2 ms. 500 data sets are generated per second.
8	- **No I/O data**: No I/O data are recorded. - **I/O**: Inputs and/or outputs are recorded, depending on the configuration in the I/O table. - **CYCFLAG**: All cyclical flags [1] … [32] which have been assigned a value are recorded. Specific cyclical flags cannot be selected. **Note:** Data are recorded every 12 ms. Approx. 85 data sets are generated per second.
9	A comment can be entered here.
10	Status of the oscilloscope - **#T_WAIT**: The oscilloscope is waiting for the trigger. - **#TRIGGERED**: The recording shall continue for the time defined by the trace length and trigger. - **#T_END**: The oscilloscope does not record any data.

Trigger

Trigger	Description
Trigger on I/O state	Additional boxes are displayed if this trigger is selected. An input, output or cyclical flag must be selected here, together with a state.
Start by user, recording until buffer is full	The recording must be started with the **Start** softkey. The recording stops after a defined duration. This trigger must be selected if the recording is to be started from a KRL program.

Trigger	Description
Cyclical recording until user stop	The recording starts when the **Start** softkey is pressed. The recording stops when the **Stop** softkey is pressed. The defined duration before **Stop** is pressed is then displayed in the TRC file. Example: A recording duration of 12 s is defined. The user presses **Start** and 30 s later **Stop**. The last 12 s before **Stop** was pressed are then displayed in the TRC file.
Trigger on error	Trigger = occurrence of a fault that causes the robot to stop. With this trigger, it is useful for the TRC file also to display data from before the trigger. Define in the main window, in the **Pre-Trigger (%)** box.
Trigger on motion start	The trace is started as soon as one or more axes move. The recording stops after a defined duration.
Trigger on clearing filter	This function is intended for KUKA service personnel only.
Trigger on DSE error	Trigger = fault in the drive circuit. With this trigger, it is useful for the TRC file also to display data from before the trigger. Define in the main window, in the **Pre-Trigger (%)** box.

Softkeys

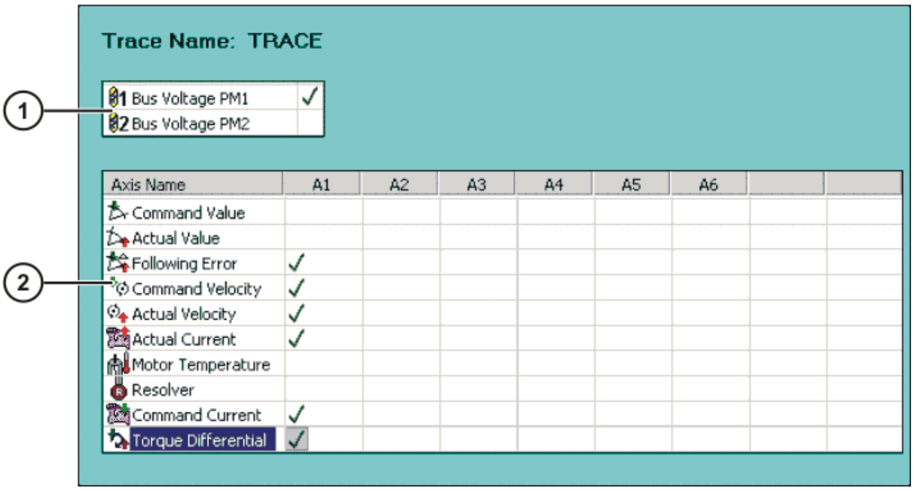
Softkey	Description
Display	Jumps to the oscilloscope display.
DSE Table	Displays the DSE table.
I/O Table	Displays the I/O table.
Trigger	Starts the recording immediately, irrespective of the trigger.
Start	This softkey is only displayed if no recording is in progress. Starts the recording in accordance with the trigger.
Stop	This softkey is only displayed if the recording is in progress. Stops the recording.
Save	Saves the current configuration. It is available again next time the oscilloscope function is opened.
Close	Closes the configuration window. The values are not saved.

13.4.1.2 DSE table

Description

Here you can select which DSE data are to be recorded.

A maximum of 21 variables can be selected. Reason: 21 DSE channels are available.



Trace Name: TRACE

1 Bus Voltage PM1	✓
2 Bus Voltage PM2	

Axis Name	A1	A2	A3	A4	A5	A6		
Command Value								
Actual Value								
Following Error	✓							
Command Velocity	✓							
Actual Velocity	✓							
Actual Current	✓							
Motor Temperature								
Resolver								
Command Current	✓							
Torque Differential	✓							

Fig. 13-7: Oscilloscope configuration – DSE table

Item	Description
1	Select which intermediate circuit voltage is to be recorded.
2	Select which data are to be recorded.

DSE data	Description
Bus Voltage PM1	KPS600
Bus Voltage PM1	KPS600 in the top-mounted cabinet (only relevant if top-mounted cabinet present)
Command Value	Command value from the interpolator per position control cycle
Actual Value	Actual value per position control cycle
Following Error	Difference between command position and actual position
Command Velocity	At the position controller output
Actual Velocity	Motor speed
Actual Current	Current sensors
Motor Temperature	In increments
Resolver	Encoder position
Command Current	At the speed controller output
Torque Differential	Value calculated from the command/actual current

Key commands

Key command	Action
Arrow keys	Switch from cell to cell.
Space bar	Select cell or cancel selection.

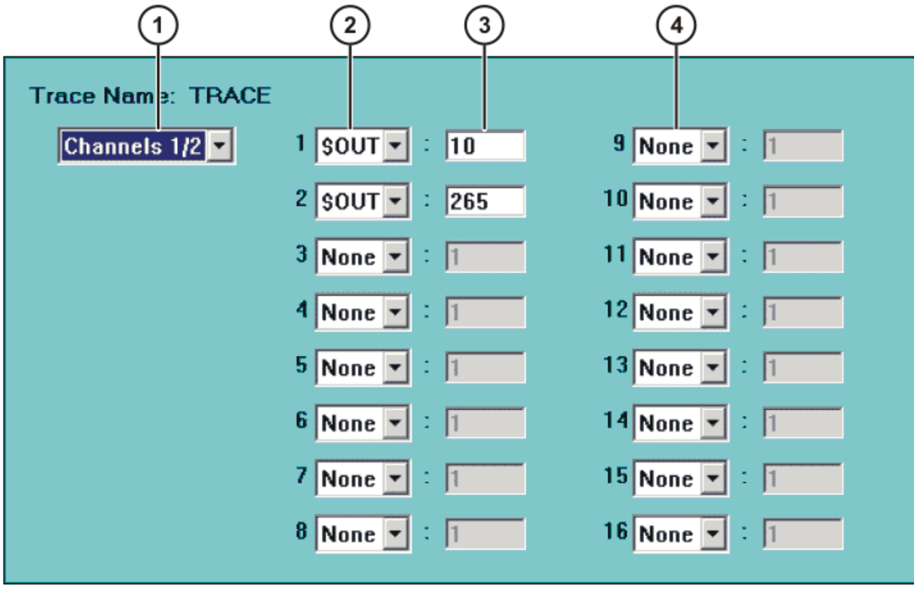
Softkeys

Softkey	Description
Main menu	Displays the main window.
I/O Table	Displays the I/O table.
Select Line	Selects the whole of the current line.
Clear Line	Removes all selections in the current line.
Clear All	Removes all selections in the lower table.

13.4.1.3 I/O table

Description Here you can select which inputs or outputs are to be recorded.

There are 4 channels available for the recording. A maximum of 8 inputs or outputs can be assigned to each channel. Each channel is subsequently displayed in a different color in the trace recording.

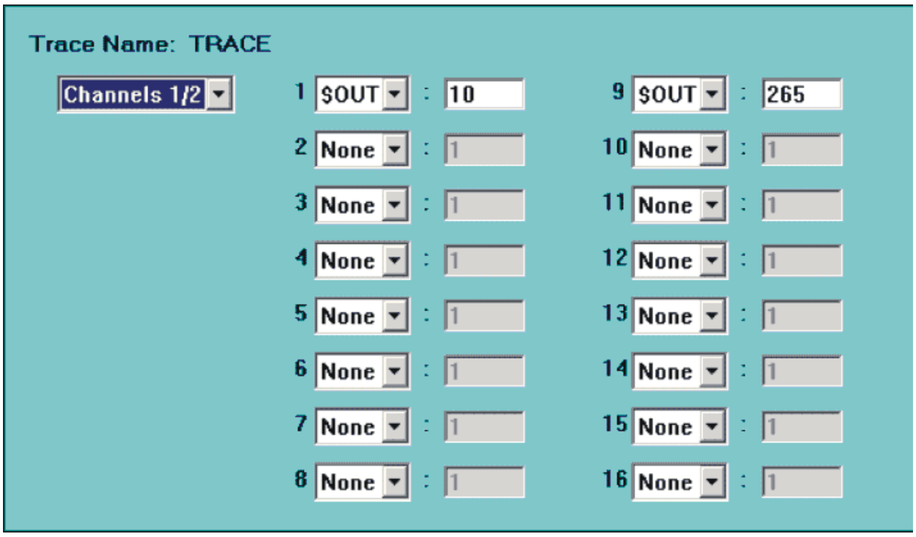


Trace Name: TRACE

Channels 1/2

1	$OUT	10	9	None	1
2	$OUT	265	10	None	1
3	None	1	11	None	1
4	None	1	12	None	1
5	None	1	13	None	1
6	None	1	14	None	1
7	None	1	15	None	1
8	None	1	16	None	1

Fig. 13-8: Example: 2 outputs are recorded, 1 channel = 1 color



Trace Name: TRACE

Channels 1/2

1	$OUT	10	9	$OUT	265
2	None	1	10	None	1
3	None	1	11	None	1
4	None	1	12	None	1
5	None	1	13	None	1
6	None	1	14	None	1
7	None	1	15	None	1
8	None	1	16	None	1

Fig. 13-9: Example: 2 outputs are recorded, 2 channels = 2 colors

Item	Description
1	Display the channels in the I/O table: ■ **Channels 1/2** ■ or **Channels 3/4**
2	Channel 1 Select whether an input or output is to be recorded: ■ **None** ■ **$IN** ■ **$OUT**
3	Enter the number of the input/output to be recorded.
4	Channel 2

Softkeys

Softkey	Description
Main menu	Displays the main window.
DSE Table	Displays the DSE table.

13.4.1.4 Starting the recording via a program

Description

It is also possible to start a recording via a program (instead of via the softkeys **Trigger** or **Start** in the configuration).

In this case, the trigger **User start, record until buffer is full** must be selected in the main window.

Example

```
 1  DEF myprogramm ( )
 2  INI
 3  $TRACE.NAME[]="CUR_"
 4
 5  PTP HOME Vel= 100 % DEFAULT
 6  LOOP
 7
 8  $TRACE.MODE=#T_START
 9  WAIT SEC 0.1
10  REPEAT
11  WAIT SEC 0.2
12  UNTIL $TRACE.STATE==#T_WAIT
13  WAIT SEC 0.5
14
15  PTP ...
16  ...
17
18  WAIT SEC 0.5
19  $TRACE.MODE=#T_STOP
20  WAIT SEC 0.1
21  REPEAT
22  WAIT SEC 0.2
23  UNTIL $TRACE.STATE==#T_END
24  WAIT SEC 0.1
25  ...
26  PTP HOME Vel= 100 % DEFAULT
27
28  ENDLOOP
29  END
```

Line	Description
3	Name of the TRC file A name must always be defined in the main window of the configuration. However, the name in the program overwrites the one from the main window. This makes it possible to start several different recordings (with different names) via several different programs, without constantly overwriting the TRC files.
8	Starts the recording in accordance with the trigger. (Corresponds to the **Start** softkey.)
9 … 13	Here it is assured that the oscilloscope reaches the state #T_WAIT before the robot motions begin.
15	The robot motions start.
19	Stops the recording (corresponds to the **Stop** softkey).
20 … 24	Here it is assured that the oscilloscope reaches the state #T_END before the robot moves to the HOME position and the loop starts again.

13.4.1.5 Configuring the oscilloscope – example 1

Description

The following variables are to be recorded:

- Command velocity of the drive of axis 1
- Actual velocity of the drive of axis 1
- Outputs 9 to 16 and 20 to 23

The recording should be called TRACE and have a duration of 12 s.

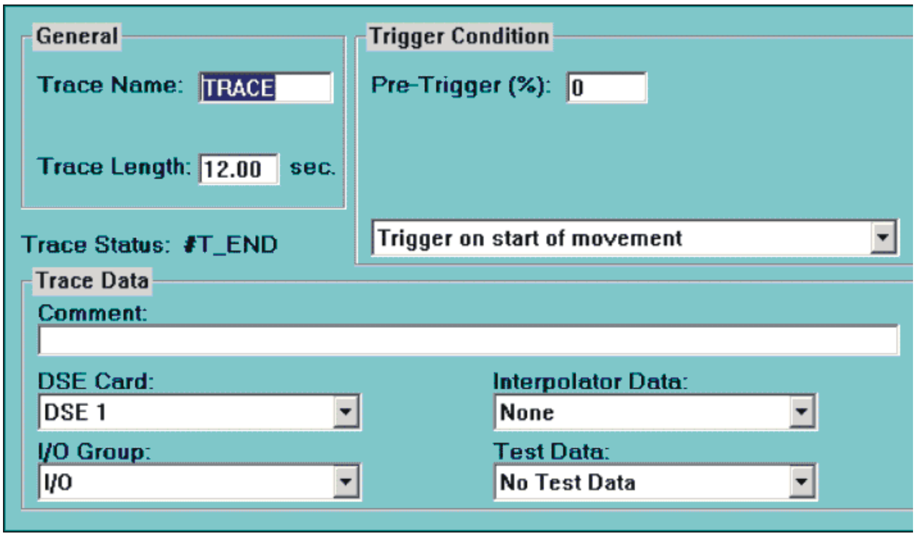
Procedure

1. Configure the **Main window**, **DSE table** and **I/O table**.
2. Start a KRL program.
3. Press the **Start** softkey. 2 files are generated:
   - TRACE1.trc (contains the DSE data)
   - TRACE3.trc (contains the I/O data)

   The files represent the time from the first robot motion to 12 s afterwards.

Main window

Make the following settings in the main window:



General

Trace Name: TRACE

Trace Length: 12.00 sec.

Trace Status: #T_END

Trigger Condition

Pre-Trigger (%): 0

Trigger on start of movement

Trace Data

Comment:

DSE Card: DSE 1

Interpolator Data: None

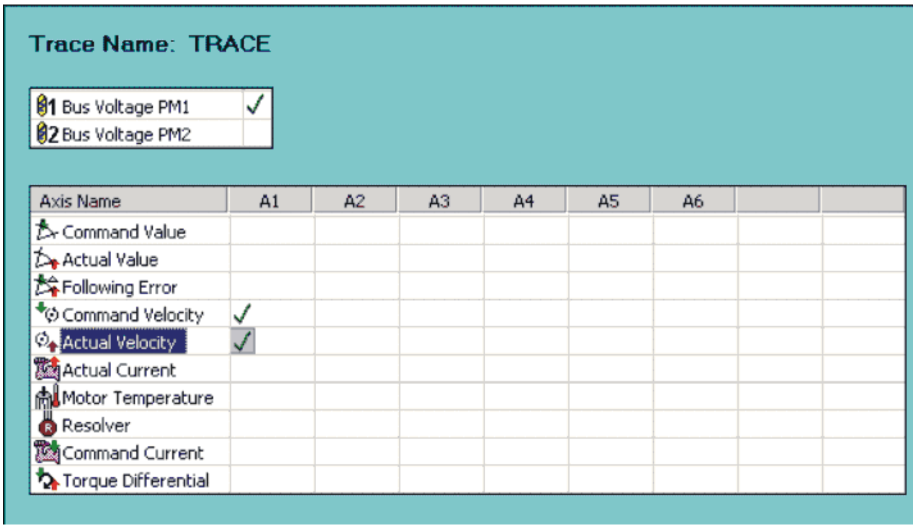
I/O Group: I/O

Test Data: No Test Data

Fig. 13-10: Example 1, main window

DSE table

Make the following settings in the DSE table:



Trace Name: TRACE

1 Bus Voltage PM1	✓
2 Bus Voltage PM2	

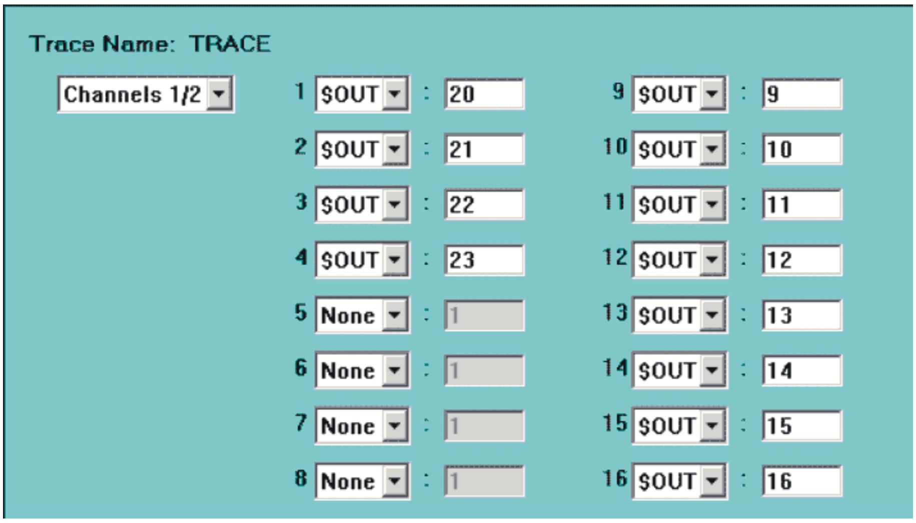
Axis Name	A1	A2	A3	A4	A5	A6
Command Value						
Actual Value						
Following Error						
Command Velocity	✓					
Actual Velocity	✓					
Actual Current						
Motor Temperature						
Resolver						
Command Current						
Torque Differential						

Fig. 13-11: Example 1, DSE table

I/O table

Make the following settings in the I/O table:

The outputs need not be specified in numerical order.



Trace Name: TRACE

Channels 1/2

1	$OUT	20	9	$OUT	9
2	$OUT	21	10	$OUT	10
3	$OUT	22	11	$OUT	11
4	$OUT	23	12	$OUT	12
5	None	1	13	$OUT	13
6	None	1	14	$OUT	14
7	None	1	15	$OUT	15
8	None	1	16	$OUT	16

Fig. 13-12: Example 1, I/O table

13.4.1.6 Configuring the oscilloscope – example 2

Description

Background: A KRL program calls subprograms in accordance with signals from the PLC. For unknown reasons, it also calls subprogram SP55.src; this is not desired.

Remedy: A free output, e.g. output 32, is set in subprogram SP55.src and reset when the subprogram is left. The recording is started in accordance with this output. The I/O communication of the KRL program with the PLC can then be analyzed using this recording.

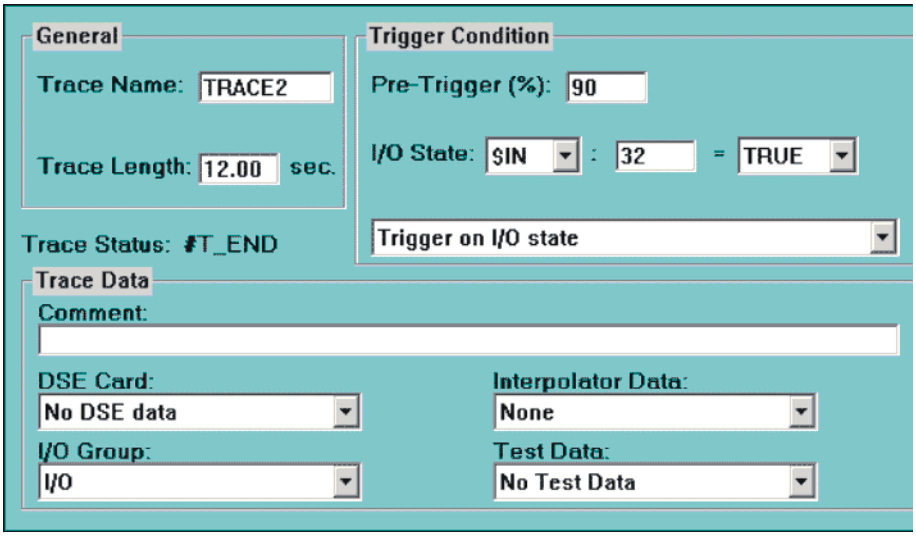
The affected robot controller inputs/outputs are inputs 1 to 16 and outputs 1 to 16. The recording should be called TRACE2 and have a duration of 12 s.

Procedure

1. Configure the **Main window** and **I/O table**.
2. Start a KRL program.
3. Press the **Start** softkey. 1 file is generated:
   - TRACE23.trc (contains the I/O data)

Main window

Make the following settings in the main window:



General

Trace Name: TRACE2

Trace Length: 12.00 sec.

Trace Status: #T_END

Trigger Condition

Pre-Trigger (%): 90

I/O State: $IN : 32 = TRUE

Trigger on I/O state

Trace Data

Comment:

DSE Card: No DSE data

Interpolator Data: None

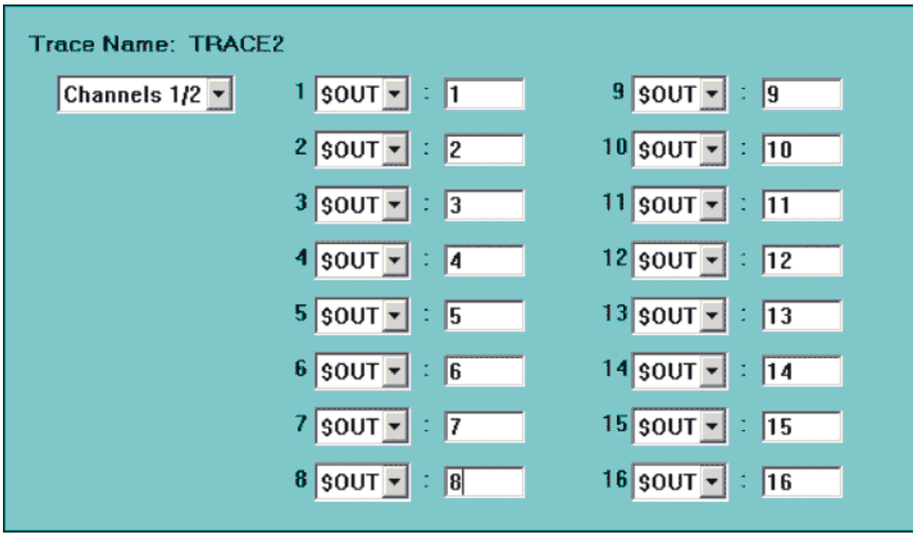
I/O Group: I/O

Test Data: No Test Data

Fig. 13-13: Example 2, main window

The value in the **Pre-Trigger** box is set to *90* or a similarly high value, as the important thing for fault analysis is what happens before output 32 is set. The inputs and outputs are then recorded for 10.8 s before this event and for 1.2 s after it.

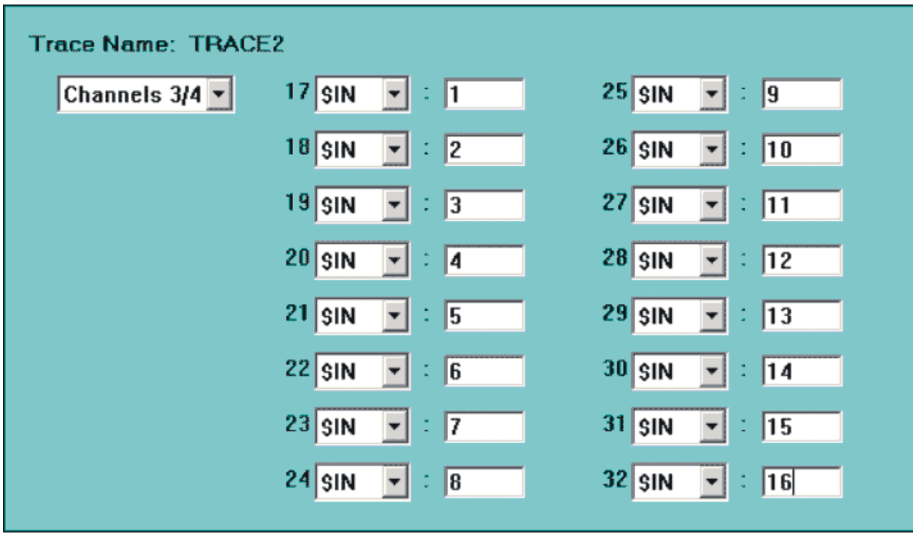
I/O table Make the following settings in the I/O table:



Trace Name: TRACE2

Channels 1/2	1	$OUT	1	9	$OUT	9
	2	$OUT	2	10	$OUT	10
	3	$OUT	3	11	$OUT	11
	4	$OUT	4	12	$OUT	12
	5	$OUT	5	13	$OUT	13
	6	$OUT	6	14	$OUT	14
	7	$OUT	7	15	$OUT	15
	8	$OUT	8	16	$OUT	16

Fig. 13-14: Example 2, I/O table (outputs)



Trace Name: TRACE2

Channels 3/4	17	$IN	1	25	$IN	9
	18	$IN	2	26	$IN	10
	19	$IN	3	27	$IN	11
	20	$IN	4	28	$IN	12
	21	$IN	5	29	$IN	13
	22	$IN	6	30	$IN	14
	23	$IN	7	31	$IN	15
	24	$IN	8	32	$IN	16

Fig. 13-15: Example 2, I/O table (inputs)

13.4.1.7 Configuring the oscilloscope – example 3

Description Background: A KRL program always stops at night because an error occurs in the I/O communication with the PLC. It is unclear whether the PLC program or the KRL program is causing the error.

Remedy: Start the oscilloscope from the KRL program, before the affected section of the program. The old TRC file is overwritten by the new one every time the program is correctly executed. If the program is interrupted by the error, however, the last TRC file is retained and the I/O communication can be analyzed with the aid of the recorded data.

The recording should be called TRACE3.

Precondition

- "Expert" user group

Procedure

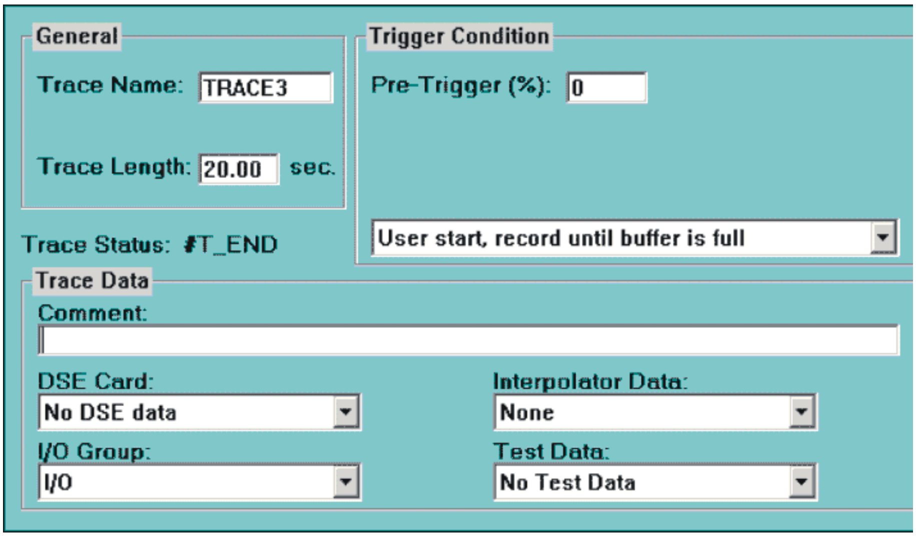
1. Configure the **Main window** and **I/O table**.
2. Insert instructions into the KRL program for starting and ending the oscilloscope.

   (>>> 13.4.1.4 "Starting the recording via a program" Page 385)
3. Restart the KRL program.

> Once the error has been located, remove the instructions from the KRL program again!

Main window

Make the following settings in the main window:



Trace Name: TRACE3

Trace Length: 20.00 sec.

Trace Status: #T_END

Pre-Trigger (%): 0

User start, record until buffer is full

Comment:

DSE Card: No DSE data

Interpolator Data: None

I/O Group: I/O

Test Data: No Test Data

Fig. 13-16: Example 3, main window

The length of the trace must be selected in such a way that the relevant I/O communication between the robot controller and the PLC is recorded.

I/O table

Set all affected inputs/outputs in the I/O table.

13.4.2 Displaying recorded data

Description

This procedure can be used to select and display a recording (=TRC file) located in the directory C:\KRC\ROBOTER\TRACE.

The robot controller appends a number to the end of the name of every TRC file, indicating what data have been recorded.

- **1**: DSE data
- **3**: I/O data
- **4**: Interpolator data
- **5**: Test data

Procedure

1. Select the menu sequence **Monitor** > **Diagnosis** > **Oscilloscope** > **Display**.
2. The TRC files are displayed. Select the desired file and press **OK**. The trace is displayed.
3. The LEFT and RIGHT ARROW keys can be used to scroll along the X axis.

   The UP and DOWN ARROW keys can be used to scroll along the Y axis.

4. The ESC key can be used to close the oscilloscope.

13.4.2.1 User interface

Recording

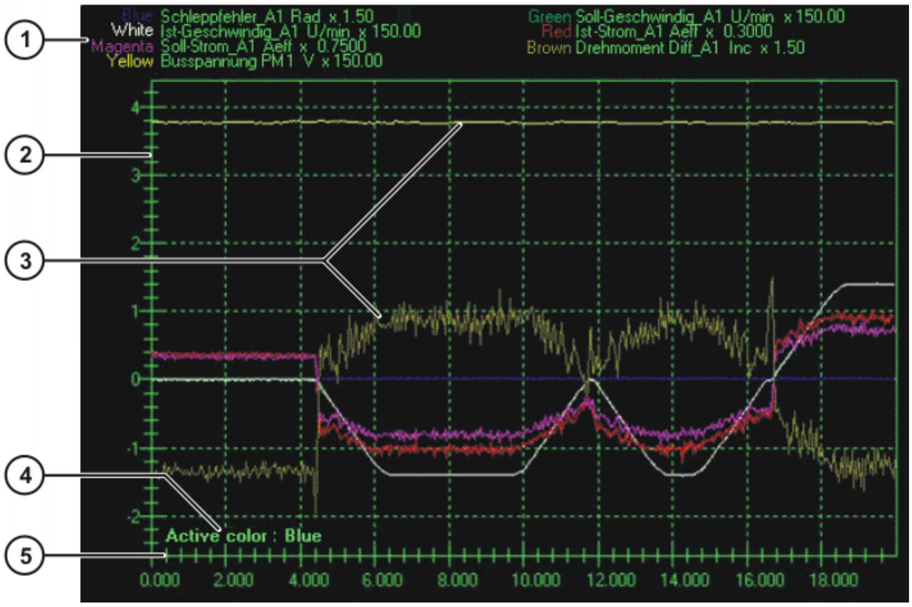

1
2
3
4
5
Blue Schleppfehler_A1 Rad x 1.50
White Ist-Geschwindig_A1 U/min x 150.00
Magenta Soll-Strom_A1 Aeff x 0.7500
Yellow Busspannung PM1 V x 150.00
Green Soll-Geschwindig_A1 U/min x 150.00
Red Ist-Strom_A1 Aeff x 0.3000
Brown Drehmoment Diff_A1 Inc x 1.50
4
3
2
1
0
-1
-2
Active color : Blue
0.000
2.000
4.000
6.000
8.000
10.000
12.000
14.000
16.000
18.000

Fig. 13-17: Example of a trace recording

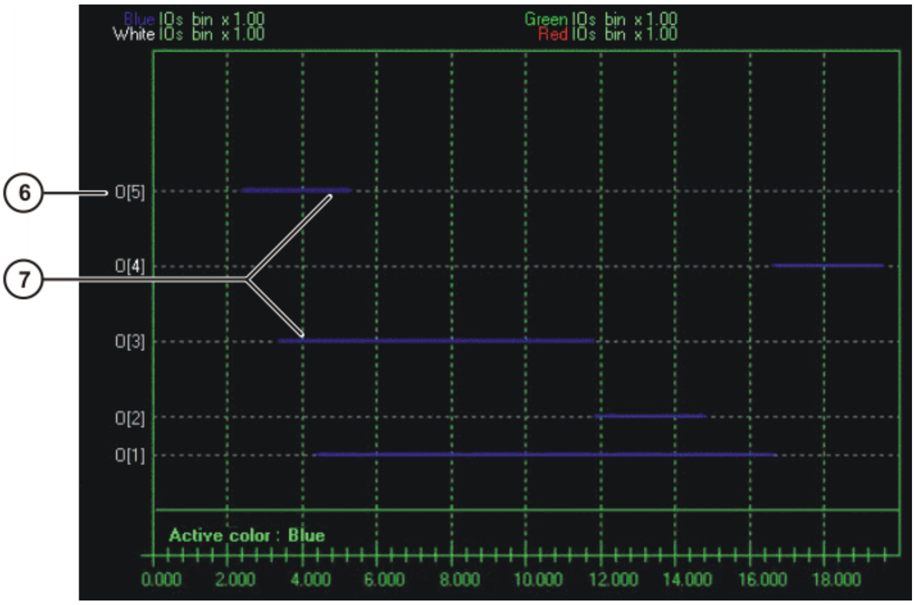


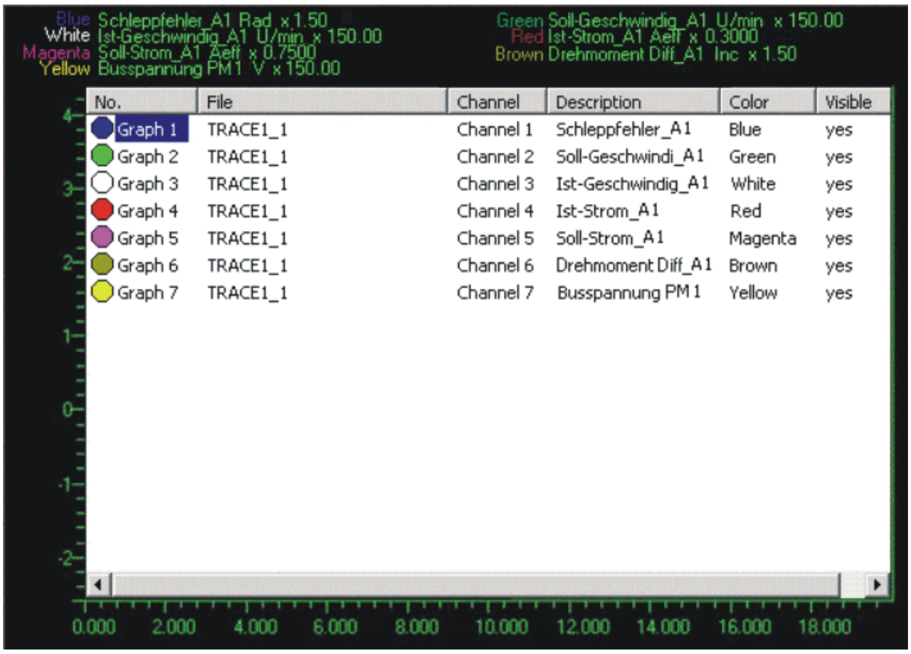
Fig. 13-18: Example of a trace recording of outputs

1	Characteristics of the curves	5	X axis
2	Y axis	6	Output
3	Curves	7	Curves for inputs/outputs
4	Active color/curve		

Element	Description
Characteristics of the curves	From left to right: color, designation, unit, scale
Y axis	The value of the curve on the Y axis multiplied by the scale gives the value of the curve at a specific point in time. If inputs and outputs have been recorded, the inputs and outputs are labeled on the Y axis with numbers. ■ **I**[*x*]: input no. *x* ■ **O**[*x*]: output no. *x*
Curves for inputs/outputs	Inputs and outputs are only displayed if they were TRUE. If an input or output was FALSE, a dotted gray line is displayed. The color of inputs/outputs depends on which channels they were assigned to in the configuration. If they were all assigned to the same channel, they will all have the same color.
Active color/curve	Active color/curve
X axis	Time; unit: seconds

Info window

The Info window displays information about the curves. The Info window can be opened and closed by means of the **I** key.



Blue Schleppfehler_A1 Rad x 1.50
White Ist-Geschwindig_A1 U/min x 150.00
Magenta Soll-Strom_A1 Aeff x 0.7500
Yellow Busspannung PM1 V x 150.00
Green Soll-Geschwindig_A1 U/min x 150.00
Red Ist-Strom_A1 Aeff x 0.3000
Brown Drehmoment Diff_A1 Inc x 1.50

No.	File	Channel	Description	Color	Visible
Graph 1	TRACE1_1	Channel 1	Schleppfehler_A1	Blue	yes
Graph 2	TRACE1_1	Channel 2	Soll-Geschwindi_A1	Green	yes
Graph 3	TRACE1_1	Channel 3	Ist-Geschwindig_A1	White	yes
Graph 4	TRACE1_1	Channel 4	Ist-Strom_A1	Red	yes
Graph 5	TRACE1_1	Channel 5	Soll-Strom_A1	Magenta	yes
Graph 6	TRACE1_1	Channel 6	Drehmoment Diff_A1	Brown	yes
Graph 7	TRACE1_1	Channel 7	Busspannung PM1	Yellow	yes

Fig. 13-19: Info window

Column	Description
No.	Number and color of the curve
File	File from which the curve data have been read
Channel	Channel to which the curve is assigned
Description	Variable represented by the curve
Color	Name of color
Visible	**Yes**: curve is visible. **No**: curve is invisible.

Operator control

The user interface can be operated by means of key commands and with softkeys. For most actions, both a key command and a softkey are available.

Key commands:

Key	Description	Corresponds to softkey
ESC	Closes the oscilloscope.	**Cancel**; **Close**
UP ARROW/ DOWN ARROW	Scrolling along the X axis.	- - - -
LEFT ARROW/ RIGHT ARROW	Scrolling along the Y axis.	- - - -
D	Switches the filter on or off. (>>> 13.4.2.7 "Filtering the display" Page 397)	**Filt. on/off**
E	Activates or deactivates the RMS function. (>>> 13.4.2.8 "Determining the r.m.s. value" Page 398)	**RMS**
I	Displays or hides the Info window.	**Info**
K	Activates a different curve. (Curves/channels which have not been assigned a color cannot be activated.) **Note:** After pressing **K**, always press the **Unzoom** softkey. This reinitializes the display so that the correct information is shown.	**Channel**
M	Enlarges the active curve.	- - - -
N	Shrinks the active curve.	- - - -
Q	Switches to the window where a TRC file can be selected.	- - - -
S	Saves the trace as a BMP file in the directory C:\KRC\ROBOTER\TRACE. (>>> 13.4.2.10 "Saving the display as a BMP file" Page 402)	**Save**
Z	Activates the Zoom function. (>>> 13.4.2.6 "Enlarging the display" Page 396)	**Zoom**
U	Deactivates the Zoom function.	**Unzoom**
C	Keys for the cursor functions (>>> 13.4.2.9 "Cursor functions" Page 400)	**VCursor 1**
V		**VCursor 2**
H		**HCursor 1**
J		**HCursor 2**
W	Switches to the oscilloscope configuration.	**Config**

> **V** key or **VCursor 2** softkey only works if **C** or **VCursor 1** has already been pressed.
> **J** key or **HCursor 2** softkey only works if **H** or **HCursor 1** has already been pressed.

Softkeys in the window with the TRC files:

Softkey	Description
Config	Switches to the oscilloscope configuration.
TraceFile2	This softkey is used to select a TRC file that is to be compared with another one. (>>> 13.4.2.2 "Superimposing traces" Page 394)
OK	Confirms the selection of a TRC file.
Cancel	Closes the oscilloscope.

If a trace is opened, 4 softkey bars are available.

Softkey bar 1:

Softkey	Description
Info	Displays or hides the Info window.
Blue; **Green**; **White**; **Red**	Displays or hides the curve with this color. If the Info window is open: assigns this color to the selected curve. If the selected curve already has this color, the curve is hidden.
-->	Displays softkey bar 2.
Close	Closes the oscilloscope.

Softkey bar 2:

Softkey	Description
Magenta; **Brown**; **Yellow**; **Cyan**	Displays or hides the curve with this color. If the Info window is open: assigns this color to the selected curve. If the selected curve already has this color, the curve is hidden.
Channel	Activates a different curve. (Curves/channels which have not been assigned a color cannot be activated.) **Note:** After pressing **Channel**, always press the **Unzoom** softkey. This reinitializes the display so that the correct information is shown.
-->	Displays softkey bar 3.
Close	Closes the oscilloscope.

Softkey bar 3:

Softkey	Description
Zoom	Activates the Zoom function. (>>> 13.4.2.6 "Enlarging the display" Page 396)
Unzoom	Deactivates the Zoom function.

Softkey	Description
Save	Saves the current oscilloscope display as a BMP file in the directory C:\KRC\ROBOTER\TRACE. (>>> 13.4.2.10 "Saving the display as a BMP file" Page 402)
RMS	Activates or deactivates the RMS function. (>>> 13.4.2.8 "Determining the r.m.s. value" Page 398)
Print	Prints the display. (>>> 13.4.2.11 "Printing the display" Page 402)
-->	Displays softkey bar 4.
Close	Closes the oscilloscope.

Softkey bar 4:

Softkey	Description
VCursor 1	Softkeys for the cursor functions (>>> 13.4.2.9 "Cursor functions" Page 400)
VCursor 2	
HCursor 1	
HCursor 2	
Filt. on/off	Switches the filter on or off. (>>> 13.4.2.7 "Filtering the display" Page 397)
-->	Displays softkey bar 1.
Close	Closes the oscilloscope.

13.4.2.2 Superimposing traces

Description

This function can be used to superimpose 2 traces. This makes it possible to compare them with one another.

The traces are displayed superimposed. The information in the upper area of the window always refers to the second file that was selected.

Procedure

1. Select the menu sequence **Monitor** > **Diagnosis** > **Oscilloscope** > **Display**.
2. The TRC files are displayed. Select a file and press the **TraceFile2** softkey. The trace is displayed.
3. Select the file that is to be compared with the first file and press the **OK** softkey.

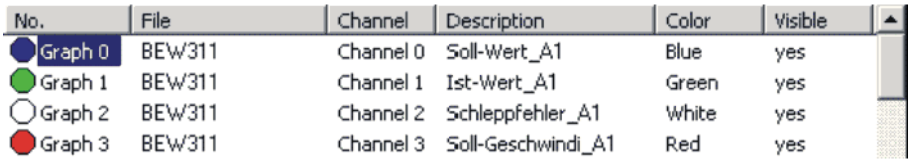
13.4.2.3 Activating and deactivating curves

Procedure

1. Press the softkey with the color of the curve. The curve is removed from the display.
2. To reactivate the curve, press the softkey again.

Alternative procedure

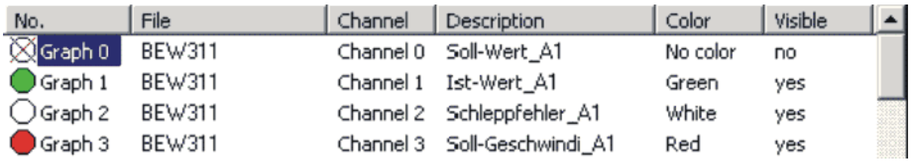
1. Press **I**. The Info window is opened.
2. Select the curve that is to be deactivated (e.g. Blue).



No.	File	Channel	Description	Color	Visible
🔵 Graph 0	BEW311	Channel 0	Soll-Wert_A1	Blue	yes
🟢 Graph 1	BEW311	Channel 1	Ist-Wert_A1	Green	yes
○ Graph 2	BEW311	Channel 2	Schleppfehler_A1	White	yes
🔴 Graph 3	BEW311	Channel 3	Soll-Geschwindi_A1	Red	yes

Fig. 13-20: Curve is selected

3. Press Enter. The curve is removed from the display.



No.	File	Channel	Description	Color	Visible
⊗ Graph 0	BEW311	Channel 0	Soll-Wert_A1	No color	no
🟢 Graph 1	BEW311	Channel 1	Ist-Wert_A1	Green	yes
○ Graph 2	BEW311	Channel 2	Schleppfehler_A1	White	yes
🔴 Graph 3	BEW311	Channel 3	Soll-Geschwindi_A1	Red	yes

Fig. 13-21: Curve is deactivated

4. Press **I** again. The Info window is closed.
5. To reactivate the curve, repeat steps 1 to 4.

13.4.2.4 Changing colors

Description

The colors of the curves can be changed. This is necessary, for example, if more than 8 channels are assigned. In this case, not all channels can be displayed at the same time, as only 8 colors are available.

Procedure

Assign a different color to one curve (e.g. Blue).

1. Press **I**. The Info window is opened.
2. Select the curve that is to be displayed in blue and press the **Blue** softkey.

   The curve is now blue. The curve that was previously blue now has no color and is no longer visible.
3. Press **I** again. The Info window is closed.

Switching colors (e.g. Green and Red):

1. Press **I**. The Info window is opened.
2. Select the green curve and press the **Green** softkey. The curve now has no color.
3. Select the red curve and press the **Green** softkey. The curve that was red is now green.
4. Select the curve that was previously green (now colorless) and press the **Red** softkey. The curve is now red.
5. Press **I** again. The Info window is closed.

13.4.2.5 Scaling curves

Description

Using this function, it is also possible to make curves more visible which only have a low amplitude or which are hidden by other curves.

The current scale of the current curve is indicated in the upper area of the screen.

Procedure

1. Press **K** until the desired curve is active. The active curve is always indicated in the bottom left-hand corner.
2. Press **N** repeatedly to increase the amplitude gradually.

Press **M** repeatedly to reduce the amplitude gradually.

13.4.2.6 Enlarging the display

Description This function can be used to select and enlarge any area of the trace.

Procedure

1. Press **Z**. A white cross appears in the center of the trace. The cross represents the corner of a window.
2. Shift the cross as required using the arrow keys.
3. Press Enter. Drag with the arrow keys to form a window.

   Example: Use the UP ARROW to drag the window upwards from the cross. Then use the LEFT ARROW to extend the window across to the left from the cross.
4. Press Enter. The contents of the window are enlarged.
5. The enlargement can be undone by pressing **U**.

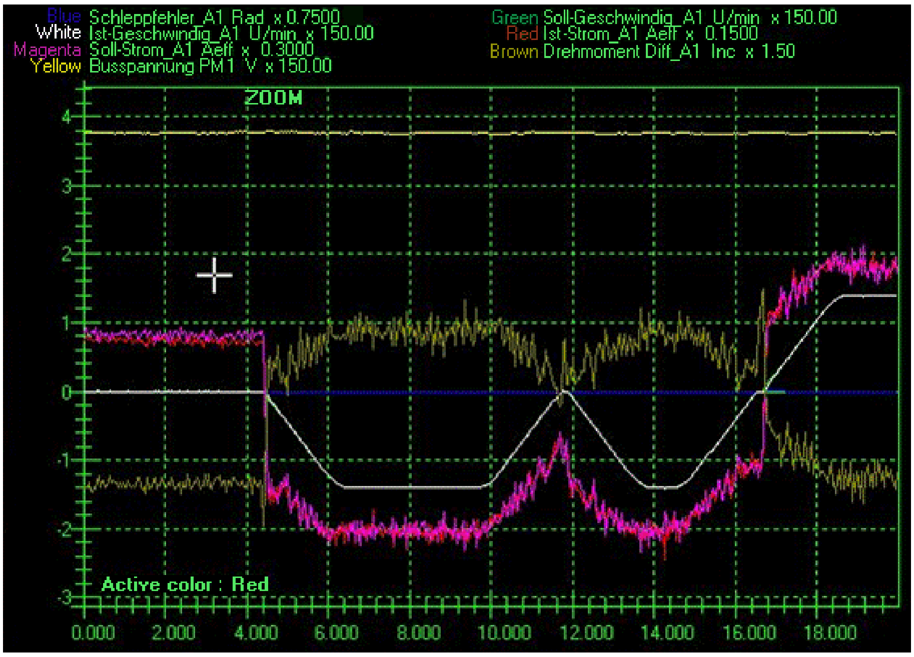

Blue Schleppfehler_A1 Rad x 0.7500
White Ist-Geschwindig_A1 U/min x 150.00
Magenta Soll-Strom_A1 Aeff x 0.3000
Yellow Busspannung PM1 V x 150.00
Green Soll-Geschwindig_A1 U/min x 150.00
Red Ist-Strom_A1 Aeff x 0.1500
Brown Drehmoment Diff_A1 Inc x 1.50
ZOOM
4
3
2
1
0
-1
-2
-3
Active color : Red
0.000
2.000
4.000
6.000
8.000
10.000
12.000
14.000
16.000
18.000

Fig. 13-22: Cross is displayed

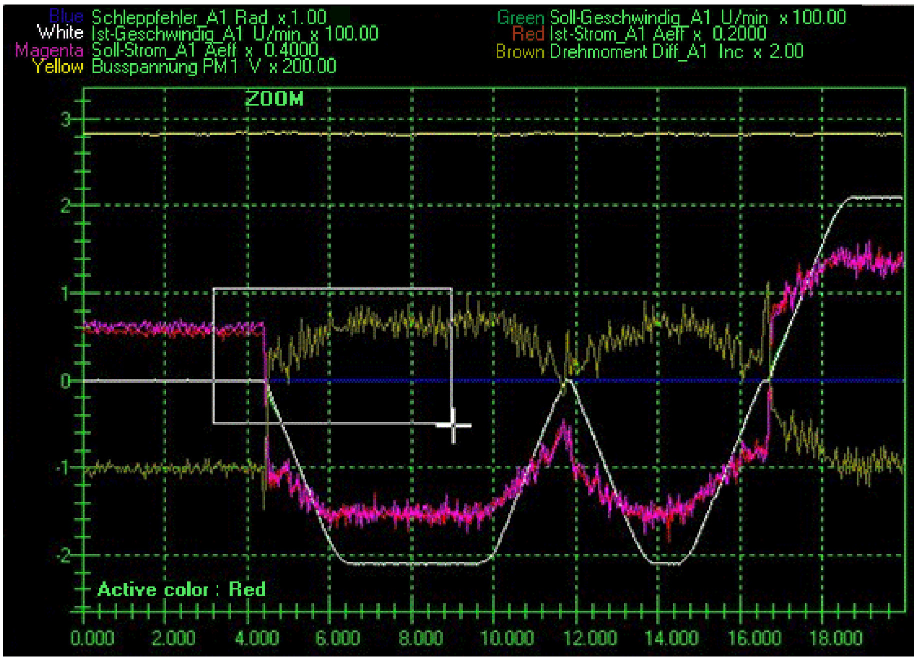

Blue Schleppfehler_A1 Rad x 1.00
White Ist-Geschwindig_A1 U/min x 100.00
Magenta Soll-Strom_A1 Aeff x 0.4000
Yellow Busspannung PM1 V x 200.00
Green Soll-Geschwindig_A1 U/min x 100.00
Red Ist-Strom_A1 Aeff x 0.2000
Brown Drehmoment Diff_A1 Inc x 2.00
ZOOM
3
2
1
0
-1
-2
Active color : Red
0.000
2.000
4.000
6.000
8.000
10.000
12.000
14.000
16.000
18.000

Fig. 13-23: Window has been stretched

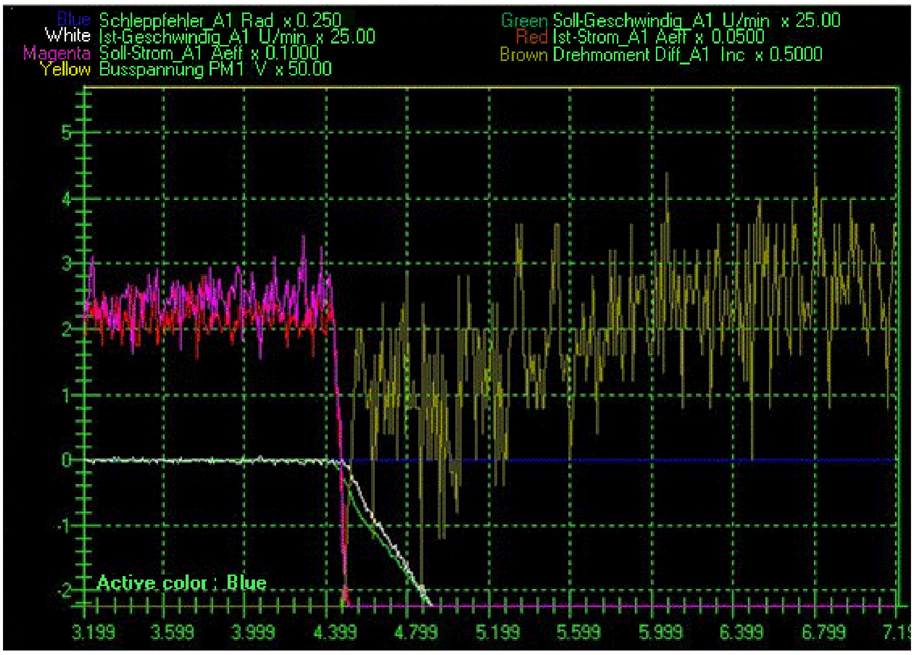

Blue Schleppfehler_A1 Rad x 0.250
White Ist-Geschwindig_A1 U/min x 25.00
Magenta Soll-Strom_A1 Aeff x 0.1000
Yellow Busspannung PM1 V x 50.00
Green Soll-Geschwindig_A1 U/min x 25.00
Red Ist-Strom_A1 Aeff x 0.0500
Brown Drehmoment Diff_A1 Inc x 0.5000
5
4
3
2
1
0
-1
-2
Active color : Blue
3.199
3.599
3.999
4.399
4.799
5.199
5.599
5.999
6.399
6.799
7.1

Fig. 13-24: Contents of the window are enlarged

13.4.2.7 Filtering the display

Description

The display is filtered by default. This ensures clear presentation of the curves. In order to obtain an exact display of the recorded data for the purpose of analysis, the filter is switched off.

Procedure

1. Press **D**. The filter is deactivated.
2. The filter can be reactivated by pressing **D** again.

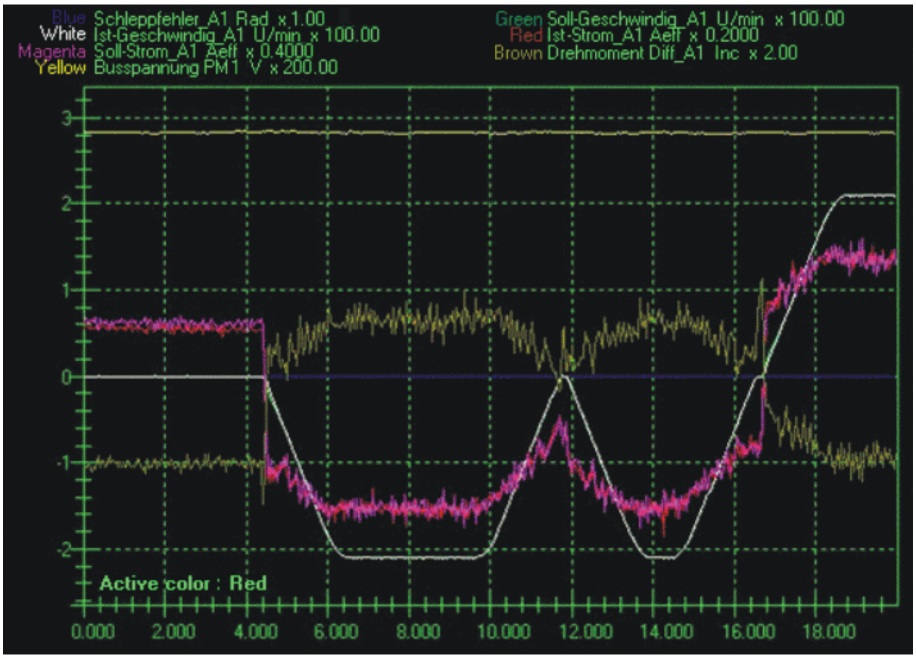


Fig. 13-25: Filter on

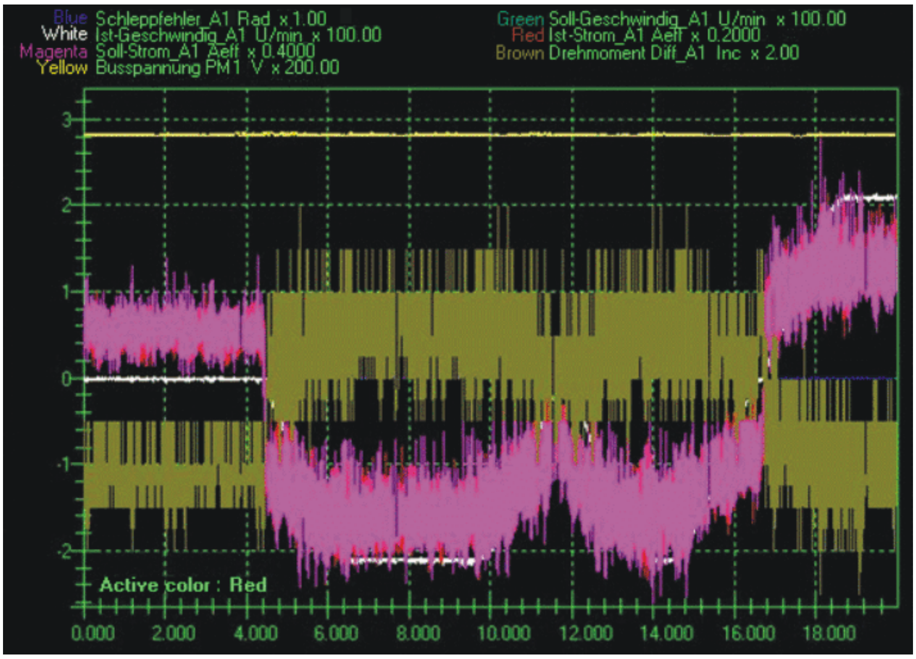


Fig. 13-26: Filter off

13.4.2.8 Determining the r.m.s. value

Description

This function can be used to determine the r.m.s. value for a particular interval. In electrical engineering, the r.m.s. value is the root-mean-square value of a signal that changes over time.

Procedure

1. Press **K** repeatedly until the desired curve is active. The active curve is always indicated in the bottom left-hand corner.
2. Press **E**. A vertical white line is displayed.

   The line marks the start of the interval on the X axis. The LEFT and RIGHT ARROW keys can be used to shift the line.

3. Press Enter. The LEFT and RIGHT ARROW keys can then be used to stretch the line to form a window. The right-hand edge of the window marks the end of the interval on the X axis.
4. Press the Enter key again. The result of the calculation is displayed in the window. (If the text is concealed by other curves, hide these curves.)
5. To end the function, press **E** again.

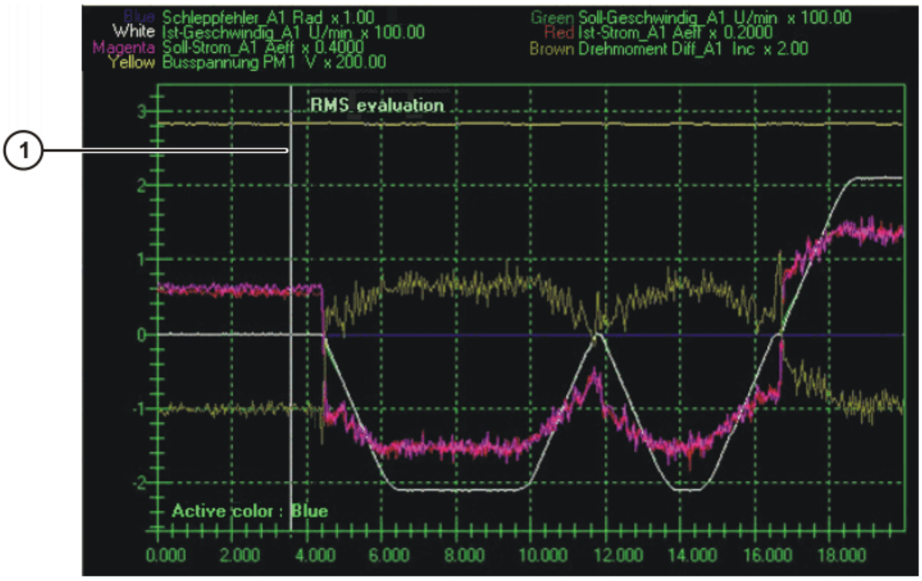

Blue Schleppfehler_A1 Rad x 1.00
White Ist-Geschwindig_A1 U/min x 100.00
Magenta Soll-Strom_A1 Aeff x 0.4000
Yellow Busspannung PM1 V x 200.00
Green Soll-Geschwindig_A1 U/min x 100.00
Red Ist-Strom_A1 Aeff x 0.2000
Brown Drehmoment Diff_A1 Inc x 2.00
RMS evaluation
Active color : Blue
1
3
2
1
0
-1
-2
0.000
2.000
4.000
6.000
8.000
10.000
12.000
14.000
16.000
18.000

Fig. 13-27: Vertical line is displayed

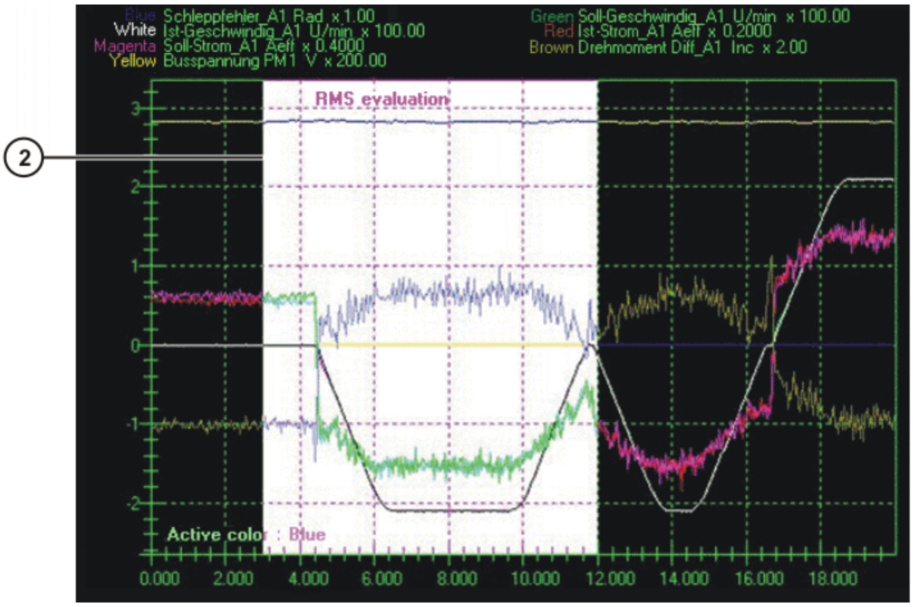

Blue Schleppfehler_A1 Rad x 1.00
White Ist-Geschwindig_A1 U/min x 100.00
Magenta Soll-Strom_A1 Aeff x 0.4000
Yellow Busspannung PM1 V x 200.00
Green Soll-Geschwindig_A1 U/min x 100.00
Red Ist-Strom_A1 Aeff x 0.2000
Brown Drehmoment Diff_A1 Inc x 2.00
RMS evaluation
Active color : Blue
2
3
2
1
0
-1
-2
0.000
2.000
4.000
6.000
8.000
10.000
12.000
14.000
16.000
18.000

Fig. 13-28: Line has been stretched to form a window

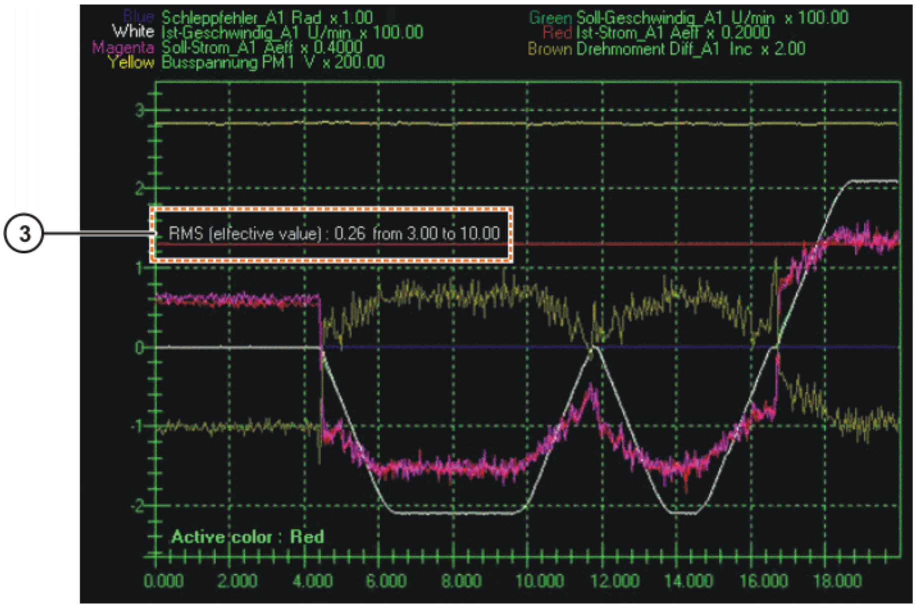

Blue Schleppfehler_A1 Rad x 1.00
White Ist-Geschwindig_A1 U/min x 100.00
Magenta Soll-Strom_A1 Aeff x 0.4000
Yellow Busspannung PM1 V x 200.00
Green Soll-Geschwindig_A1 U/min x 100.00
Red Ist-Strom_A1 Aeff x 0.2000
Brown Drehmoment Diff_A1 Inc x 2.00
3
RMS (effective value) : 0.26 from 3.00 to 10.00
3
2
1
0
-1
-2
Active color : Red
0.000
2.000
4.000
6.000
8.000
10.000
12.000
14.000
16.000
18.000

Fig. 13-29: Result is displayed

Item	Description
1	Vertical line
2	Window
3	Result

13.4.2.9 Cursor functions

Description

This function can be used to determine the value that a curve represented at a specific time. For this purpose, a vertical line is displayed, intersecting the desired time value on the X axis. The differential value between two points in time can also be determined.

If the trace represents inputs and outputs, the bit pattern of the input or output group is displayed as the value.

Additionally, a horizontal line can be used to mark a specific value on the Y axis. The difference between two values can also be determined.

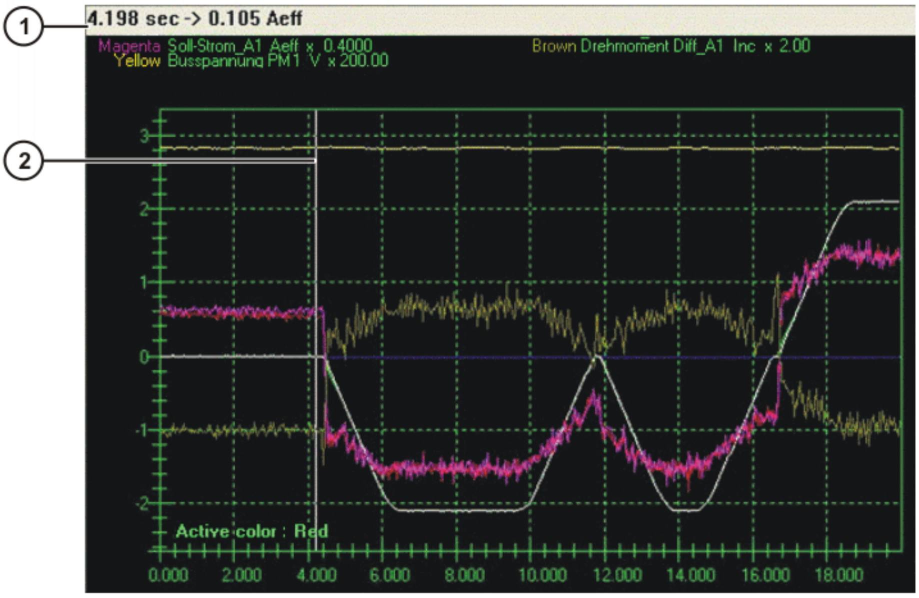
If the trace represents inputs and outputs, the horizontal lines are not suitable.

Procedure

1. Press **K** repeatedly until the desired curve is active. The active curve is always indicated in the bottom left-hand corner.
2. Press **C**. A vertical line is displayed. It marks a point in time on the X axis. The time and the corresponding measured value are indicated in the top left-hand corner.

   The LEFT and RIGHT ARROW keys can be used to shift the line.
3. Press **V**. A second vertical line is displayed. In addition to the value of the first line, the value of the second line is also displayed, together with the difference.
4. Press **H**. A horizontal line is displayed. It marks a value on the Y axis. The value is displayed in the top left-hand corner.

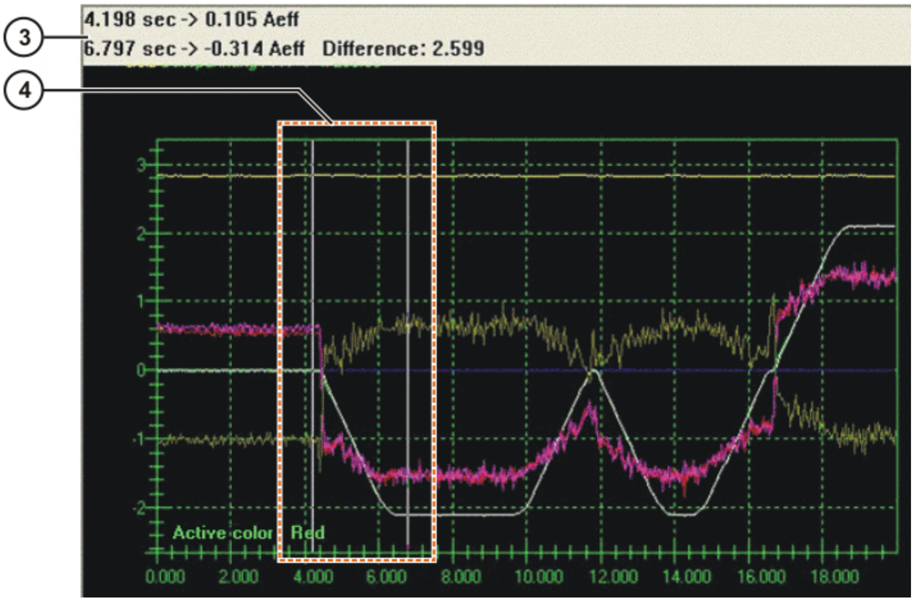
   The UP and DOWN ARROW keys can be used to shift the line.
5. Press **J**. A second horizontal line is displayed. In addition to the value of the first line, the value of the second line is also displayed, together with the difference.
6. The lines can be deactivated by pressing the keys again.



4.198 sec -> 0.105 Aeff

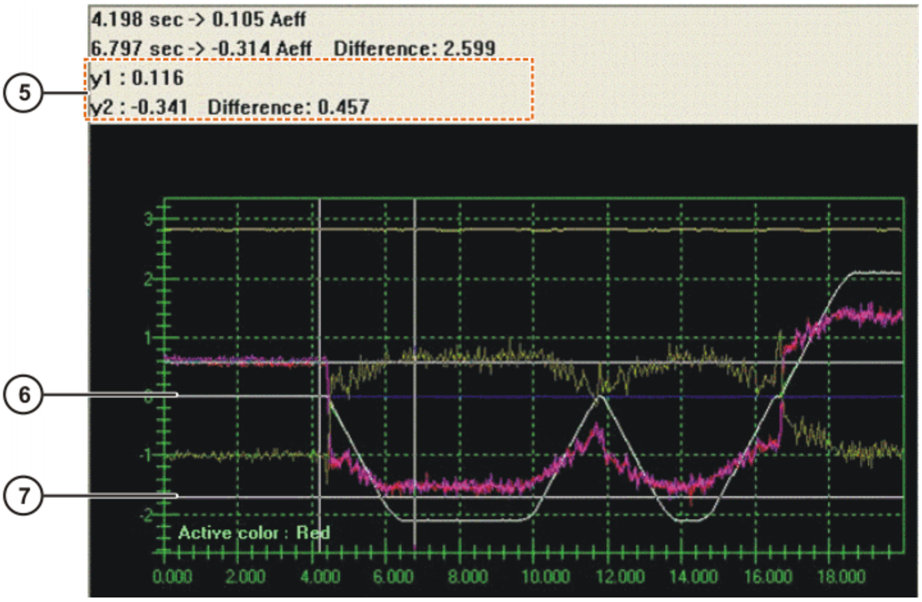
1
2
Magenta Soll-Strom_A1 Aeff x 0.4000
Yellow Busspannung PM1 V x 200.00
Brown Drehmoment Diff_A1 Inc x 2.00
3
2
1
0
-1
-2
Active color : Red
0.000
2.000
4.000
6.000
8.000
10.000
12.000
14.000
16.000
18.000

Fig. 13-30: First vertical line



4.198 sec -> 0.105 Aeff
6.797 sec -> -0.314 Aeff   Difference: 2.599

Fig. 13-31: First and second vertical lines



4.198 sec -> 0.105 Aeff
6.797 sec -> -0.314 Aeff   Difference: 2.599
y1 : 0.116
y2 : -0.341   Difference: 0.457

Fig. 13-32: First and second horizontal lines

Item	Description
1	Position and measured value of the first vertical line
2	First vertical line
3	Position and measured value of the first and second vertical lines and the difference
4	First and second vertical lines
5	Position and measured value of the first and second horizontal lines and the difference
6	First horizontal line
7	Second horizontal line

13.4.2.10 Saving the display as a BMP file

Description

This function saves the trace as a BMP file in the directory C:\KRC\ROBOTER\TRACE. All current settings, e.g. enlarged curves, are saved.

The BMP file has the same name as the TRC file. If 2 superimposed traces are saved, the name is composed of the names of both TRC files.

If a file of the same name already exists, it is overwritten by the new file without a request for confirmation!

Procedure

- Press **S**.

13.4.2.11 Printing the display

Procedure

1. Set the printer to landscape format.
2. Press the **-->** softkey repeatedly until the **Print** softkey is displayed.
3. Press the **Print** softkey. The display is printed.

14 Installation

The robot controller is supplied with a Windows operating system and an operational version of the KUKA System Software (KSS). Therefore, no installation is required during initial start-up.

Installation becomes necessary, for example, in the event of the hard drive being damaged and exchanged.

> ⚠ The robot controller may only be operated using the software provided with the controller by KUKA.
> KUKA Roboter GmbH must be consulted if different software is to be used. (>>> 16 "KUKA Service" Page 415)

14.1 Overview of the software components

Overview

The following software components are used:

- KUKA System Software 5.6 lr
- Windows XP embedded 2.0 incl. Service Pack 2

  or Windows XP embedded 2.2 incl. Service Pack 3
- VxWin RT V5.0

14.2 Installation overview

Step	Description
1	Install Windows: 1. Only required if there is no CD-ROM drive on the control cabinet: Adapt BIOS settings for USB CD/DVD drive. (>>> 14.2.1 "Adapting BIOS settings for USB CD/DVD drive" Page 403) 2. Install Windows. (>>> 14.2.2 "Installing Windows" Page 404) 3. If required, change the Windows language. (>>> 14.2.3 "Changing the Windows language" Page 404)
2	Install the KUKA System Software. (>>> 14.2.4 "Installing the KUKA System Software" Page 405)
3	Install technology packages. (>>> 14.3 "Installing additional software (via KUKA.HMI)" Page 406)
4	If applicable: load archive. (>>> 8.8.3 "Restoring data" Page 241)

14.2.1 Adapting BIOS settings for USB CD/DVD drive

This procedure is only required if there is no CD-ROM drive on the control cabinet.

Precondition

- Bootable USB CD-ROM/DVD drive
- External keyboard

Procedure

1. Connect USB CD/DVD drive.
2. After switching on the controller, press the F12 key while it is booting. The BIOS Boot Menu is displayed.
3. Select **CD-ROM Drive** as Boot Device.
4. Press the Enter key.

14.2.2 Installing Windows

NOTICE When Windows is installed, the hard drive is formatted. Data on the hard drive are deleted.

Preconditions

- KUKA Windows CD-ROM
- External keyboard
- Only required if there is no CD-ROM drive on the control cabinet:
  - Bootable USB CD-ROM/DVD drive
  - BIOS settings have been adapted.

Procedure

1. Insert KUKA Windows CD-ROM
2. Boot the robot controller.
   - If no operating system has yet been installed, the installation starts automatically.
   - If an operating system has already been installed, the prompt **Press any key to boot from ...** appears following the BIOS sequence.

     Press any key. The installation starts and the existing operating system is overwritten.

   2 partitions are set up. The first partition (drive C:\) has a defined size of 5 GB. The remaining capacity is allocated to the second partition (drive D:\). Both partitions are formatted. The operating system is loaded. The system is rebooted.
3. The KUKA drivers are installed. Depending on the system, user entries may be required, e.g. language selection or entry of a computer name. The system is then restarted.
4. Remove CD-ROM from drive.

14.2.3 Changing the Windows language

Description

By default, the Windows user interface can be displayed in German or English. Additional languages can be installed.

The following additional languages are available:

Chinese (simplified)	Dutch
Chinese (traditional)	Polish
Danish	Portuguese
Finnish	Russian
French	Swedish
Greek	Spanish
Italian	Czech
Japanese	Turkish
Korean	Hungarian

All languages other than German and English can be uninstalled again as required.

Precondition

- Windows XP embedded Languages CD

Procedure

1. Start the CD-ROM.
2. Open the folder with the desired language. Start the program SETUP.EXE. Follow the instructions in the installation wizard.
3. Open the Windows Start menu (CTRL+ESC)
4. Select **Settings** > **Control Panel** > **Regional and Language Options**. The **Regional and Language Options** window is opened.
5. Select the **Languages** tab. In the **Language used in menus and dialogs** box, select the previously installed language. Press **OK**.
6. Reboot the robot controller.

14.2.4 Installing the KUKA System Software

Precondition

- Windows XP embedded 2.0 incl. Service Pack 2 is installed.

  Or Windows XP embedded 2.2 incl. Service Pack 3 is installed.

> **NOTICE** Windows Service Packs must not be installed subsequently, as the system will not be operable.

- 256 MB RAM
- KSS CD-ROM
- Only required if there is no CD-ROM drive on the control cabinet:

  Bootable USB CD-ROM/DVD drive

Procedure

1. If a KSS version is already installed:

   Reboot the robot controller. As soon as the Windows screen is displayed, press the CTRL key and hold it down. During the loading procedure, a message referring to the Ikarus virus scanner will appear briefly. After this, keep the CTRL key held down for approximately another 15 seconds, then release it.

   This prevents the KSS from being started automatically.
2. Press the Windows **Start** button and select **Run...**. Navigate to the KSS CD-ROM with **Browse…** and start the program SETUP.EXE.
3. Select the desired language. Press the Enter key.
4. Information about the setup and copyright are displayed. Press the Enter key.
5. Select **Copy CD-ROM**. This window is not displayed if installing from the hard drive or server. Press the Enter key.
6. Select **Install Ikarus virus scanner**. Press the Enter key.
7. Select the desired customer version. Press the Enter key.
8. Only if a KSS version is already installed: It is possible to select which data are to be retained from the existing installation.

   Select data. Press the Enter key.
9. Select which bus drivers are to be installed. Press the Enter key.
10. The system suggests a robot type. If the type suggested does not correspond to the type used, select **Browse** with the TAB key and press Enter.

    In the MaDa directory, select the control cabinet used and the corresponding type. Press the Enter key.

> ℹ If, under point 8, the data from an existing KSS installation have been retained, the system does not suggest a robot type, but retains the previous one.

11. A summary of the setup settings is displayed. Press the Enter key. The installation starts.

12. A message indicates completion of the installation. Remove CD-ROM from drive.
13. Select **Yes, restart computer**. Press the Enter key.

14.3 Installing additional software (via KUKA.HMI)

This function can be used to install additional software, e.g. technology packages. New programs and updates can be installed. The software can be installed from CD or from another path. The system checks whether the additional software is relevant for the KSS. If not, the system rejects the installation. If a software package that the system has rejected is nonetheless to be installed, KUKA Roboter GmbH must be contacted.

(>>> 16.2 "KUKA Customer Support" Page 415)

This function can also be used to uninstall additional software. Multiple additional programs can be installed or uninstalled one after the other.

This function is also used to select the path for a KSS update from the network. This function cannot be used, however, to install the KSS update.

Precondition

- CD-ROM with additional software
- Only required if there is no CD-ROM drive on the control cabinet:

  Bootable USB CD-ROM/DVD drive
- User group “Expert”

Procedure

1. Select the menu sequence **Setup** > **Install Additional Software**.
2. Press the **New SW** softkey. If a software package on the CD-ROM in the drive is not yet displayed, press the **Refresh** softkey.
3. Select the software to be installed and press the softkey **Install**. Answer the request for confirmation with **Yes**. The files are copied onto the hard drive.
4. If another additional software package is to be installed, repeat step 3.
5. Depending on the specific additional software, it may be necessary to reboot the controller. In this case, a corresponding prompt will be displayed. Confirm with **OK** and restart the robot controller. The installation is resumed and completed.

Description

The following softkeys are available:

Softkey	Description
New SW	All programs available for installation are displayed. Programs on the CD-ROM in the drive are also displayed.
Back	Additional software already installed is displayed.
Refresh	Refreshes the display, e.g. after a CD-ROM has been inserted.
Install	Shows additional softkeys: ■ **Yes**: The selected software is installed. If it is necessary to reboot the controller, this is indicated by a message. ■ **No**: The software is not installed.

Softkey	Description
Config.	This softkey is only displayed if the **New SW** softkey has been pressed. Paths for the installation of additional software or KSS updates can be selected and saved here. Shows additional softkeys: ■ **Browse**: A new path can be selected. ■ **Apply**: Saves the displayed paths.
De-install	Shows additional softkeys: ■ **Yes**: The selected software is uninstalled. ■ **No**: The software is not uninstalled.

14.4 KSS update (via KUKA.HMI)

Description

This function can be used to install KSS updates, e.g. from KSS 5.6.1 lr to KSS 5.6.2 lr.

It is advisable to archive all relevant data before updating a software package. If necessary, the old version can be restored in this way. It is also advisable to archive the new version after carrying out the update.

> This function cannot be used to install new versions, e.g. from KSS 5.3 lr to KSS 5.6 lr. KUKA Roboter GmbH must be consulted before a new version is installed.
> (>>> 16 "KUKA Service" Page 415)
> This function cannot be used to install updates of additional software, such as technology packages.
> (>>> 14.3 "Installing additional software (via KUKA.HMI)" Page 406)

> ⚠ **WARNING** If machine data are reloaded after an update, the version of the machine data must correspond exactly to the KSS version. This is ensured if the machine data supplied together with the KSS release are used.
> The robot must not be moved if incorrect machine data are loaded. Personal injuries or damage to property may result in this case.

Overview

There are 2 ways of installing a KSS update:

- From CD-ROM
  (>>> 14.4.2 "KSS update from CD-ROM" Page 409)
- From the network
  (>>> 14.4.3 "KSS update from the network" Page 409)

14.4.1 Accepting user data during a KSS update

User-defined variables, initializations, etc., are only retained in a KSS update if they are defined in the following program sections.

Programs created by the user:

In the Fold USER EXT (a sub-Fold of EXTERNAL DECLARATIONS):

```
EXTERNAL DECLARATIONS
...
USER EXT
;Make here your modifications
```

System\$config.dat:

In the Fold USER GLOBALS:

```
USER GLOBALS
;*******************************************
;Make your modifications -ONLY- here
;*******************************************
;==================================
; Userdefined Types
;==================================

;==================================
; Userdefined Externals
;==================================

;==================================
; Userdefined Variables
;==================================
```

System\ir_stopm.src:

For declarations: in the Fold USER DECL (a sub-Fold of DECLARATIONS)

```
FOLD DECLARATIONS
  FOLD USER DECL
  ; Please insert user defined declarations

  ;ENDFOLD (USER DECL)
```

In the case of a stop: in the Fold USER STOP

```
;FOLD USER STOP
;Make your modifications here

;ENDFOLD (USER STOP)
```

For restart: in the Fold USER RESTART

```
;FOLD USER RESTART
;Make your modifications here

;ENDFOLD (USER RESTART)
```

sps.sub:

For declarations: in the Fold USER DECL (a sub-Fold of DECLARATIONS)

```
;FOLD DECLARATIONS
...
;FOLD USER DECL
; Please insert user defined declarations

;ENDFOLD (USER DECL)
```

For initialization: in the Fold USER INIT (a sub-Fold of INI)

```
;FOLD INI
...
;FOLD USER INIT
; Please insert user defined initialization commands

;ENDFOLD (USER INIT)
```

For a cyclical call: in the Fold USER PLC

```
;FOLD USER PLC
;Make your modifications here

;ENDFOLD (USER PLC)
```

For subprograms: in the Fold USER SUBROUTINE (a sub-Fold of the Fold ;%{H})

```
;FOLD ;%{H})
...
;FOLD USER SUBROUTINE
; Integrate your user defined subroutines

;ENDFOLD (USER SUBROUTINE)
```

14.4.2 KSS update from CD-ROM

Precondition

- Only required if there is no CD-ROM drive on the control cabinet:

  A USB CD/DVD drive is connected.

Procedure

1. Select the menu sequence **Setup** > **Software Update** > **Automatic**.
2. A request for confirmation is displayed, asking if the update should be carried out. Confirm by pressing the **Yes** softkey.
3. A message is displayed, indicating that a cold start will be forced next time the system is booted. Switch the controller off.
4. Wait until the computer has shut down completely. Then switch the controller back on.
5. Once the update has been completed, the computer is automatically shut down and rebooted.

   Only now may the user remove the installation CD-ROM from the drive or remove the USB CD/DVD drive.

14.4.3 KSS update from the network

Description

With this kind of update, the installation data are copied from the network to the local drive D:\. If there is a copy of a KSS CD-ROM on D:\, it will be overwritten, but not until the installation data have been completely transferred.

Installation is started on completion of the copying operation.

Precondition

- “Expert” user group

Preparation

Configure the network path from which the update installation is to be carried out:

1. Select the menu sequence **Setup** > **Install Additional Software**.
2. Press the **New SW** softkey.
3. Press the **Config.** softkey.
4. Select the **Installation path for KRC update via the network** box. Press the **Browse** softkey.
5. Select the desired network path. Press the **Apply** softkey.
6. The selected path is displayed in the **Installation path for KRC update via the network** box. Press the **Apply** softkey.
7. Press the **Close** softkey.

> It is only necessary to configure the network path once. It remains saved for subsequent updates.

Procedure

1. Select the menu sequence **Setup** > **Software Update** > **Net**.

2. A request for confirmation is displayed, asking if the update should be carried out. Confirm by pressing the **Yes** softkey.

   Depending on the network utilization, the procedure may take up to 15 min.
3. A message is displayed, indicating that a cold start will be forced next time the system is booted. Switch the controller off.
4. Wait until the computer has shut down completely. Then switch the controller back on.
5. Once the update has been completed, the computer is automatically shut down and rebooted.

15 Messages

15.1 System messages

Details about the system messages can be found in the online help.

(>>> 4.2.5 "Calling online help" Page 47)

15.2 Messages, LWR

Message	Cause	Effect	Remedy
Gravitation Compensation is active	Gravitation compensation is active.	The robot cannot be moved using the jog keys or in program mode.	Select a different controller.
Validate Loaddata %1	Load data are implausible.	Controllers based on torque controller cannot be activated.	Check load data and offset of the torque sensors. If required: ■ Select a different robot position, e.g. move from the mastering position. ■ Enter correct load data. ■ Reset torque sensors.
Loaddata incorrect %1	Load data do not match measured torque.	Controller is deactivated.	Check load data and mastering. If required: ■ Enter correct load data. ■ Master the robot.
Cartesian deviation from commanded trajectory exceeds limits	Parameters of the Cartesian stiffness controller are unfavorable or incorrectly set.	The robot stops and can no longer be moved.	Check and modify parameter of the Cartesian stiffness controller.
Quitt Cartesian Deviation		Robot cannot be moved.	Acknowledge the message.
Cartesian path deviation cannot be calculated	The Cartesian model is not initialized ($BASE, $TOOL).	Robot cannot be moved.	Initialize $BASE and $TOOL. OR Select position controller.
Power limitation ON	Power limitation to 80 watts is active.	———	———
Power limitation OFF	Power limitation to 80 watts is not active.	———	———

Message	Cause	Effect	Remedy
Power limitation has released !	The robot has exceeded the power limitation to 80 watts.	Drives are briefly switched off.	———
HandCmd reduced %1	The stiffness values are very low and the robot cannot follow the motion.	The desired motion cannot be executed. Jog key or Space Mouse is deactivated.	■ Cartesian jogging: change the coordinate system. Must match $STIFFNESS.FRAMETYPE. ■ Move the robot using the displayed jog keys. ■ Select a suitable motion type, e.g. axis-specific jogging in the case of axis-specific stiffness.

15.3 Automatic External error messages

No.	Message text	Cause
P00:1	PGNO_TYPE incorrect value permissible values (1,2,3)	The data type for the program number was entered incorrectly.
P00:2	PGNO_LENGTH incorrect value Range of values 1 ≤ PGNO_LENGTH ≤ 16	The selected program number length in bits was too high.
P00:3	PGNO_LENGTH incorrect value permissible values (4,8,12,16)	If BCD format was selected for reading the program number, a corresponding number of bits must also be set.
P00:4	PGNO_FBIT incorrect value not in the $IN range	The value “0” or a non-existent input was specified for the first bit of the program number.
P00:7	PGNO_REQ incorrect value not in the $OUT range	The value “0” or a non-existent output was specified for the output via which the program number is to be requested.
P00:10	Transmission error incorrect parity	Discrepancy detected when checking parity. A transmission error must have occurred.
P00:11	Transmission error incorrect program number	The higher-level controller has transferred a program number for which there is no CASE branch in the file CELL.SRC.
P00:12	Transmission error incorrect BCD encoding	The attempt to read the program number in BCD format led to an invalid result.

No.	Message text	Cause
P00:13	Incorrect operating mode	The I/O interface output has not been activated, i.e. the system variable $I_O_ACTCONF currently has the value FALSE. This can have the following causes: ■ The mode selector switch is not in the “Automatic External” position. ■ The signal $I_O_ACT currently has the value FALSE.
P00:14	Move to Home position in operating mode T1	The robot has not reached the HOME position.
P00:15	Incorrect program number	More than one input set with “1 of n”.

16 KUKA Service

16.1 Requesting support

Introduction

The KUKA Roboter GmbH documentation offers information on operation and provides assistance with troubleshooting. For further assistance, please contact your local KUKA subsidiary.

Information

The following information is required for processing a support request:

- Model and serial number of the robot
- Model and serial number of the controller
- Model and serial number of the linear unit (if applicable)
- Version of the KUKA System Software
- Optional software or modifications
- Archive of the software
- Application used
- Any external axes used
- Description of the problem, duration and frequency of the fault

16.2 KUKA Customer Support

Availability

KUKA Customer Support is available in many countries. Please do not hesitate to contact us if you have any questions.

Argentina

Ruben Costantini S.A. (Agency)
Luis Angel Huergo 13 20
Parque Industrial
2400 San Francisco (CBA)
Argentina
Tel. +54 3564 421033
Fax +54 3564 428877
ventas@costantini-sa.com

Australia

Headland Machinery Pty. Ltd.
Victoria (Head Office & Showroom)
95 Highbury Road
Burwood
Victoria 31 25
Australia
Tel. +61 3 9244-3500
Fax +61 3 9244-3501
vic@headland.com.au
www.headland.com.au

Belgium

KUKA Automatisering + Robots N.V.
Centrum Zuid 1031
3530 Houthalen
Belgium
Tel. +32 11 516160
Fax +32 11 526794
info@kuka.be
www.kuka.be

Brazil

KUKA Roboter do Brasil Ltda.
Avenida Franz Liszt, 80
Parque Novo Mundo
Jd. Guançã
CEP 02151 900 São Paulo
SP Brazil
Tel. +55 11 69844900
Fax +55 11 62017883
info@kuka-roboter.com.br

Chile

Robotec S.A. (Agency)
Santiago de Chile
Chile
Tel. +56 2 331-5951
Fax +56 2 331-5952
robotec@robotec.cl
www.robotec.cl

China

KUKA Automation Equipment (Shanghai) Co., Ltd.
Songjiang Industrial Zone
No. 388 Minshen Road
201612 Shanghai
China
Tel. +86 21 6787-1808
Fax +86 21 6787-1805
info@kuka-sha.com.cn
www.kuka.cn

Germany

KUKA Roboter GmbH
Zugspitzstr. 140
86165 Augsburg
Germany
Tel. +49 821 797-4000
Fax +49 821 797-1616
info@kuka-roboter.de
www.kuka-roboter.de

France

KUKA Automatisme + Robotique SAS
Techvallée
6, Avenue du Parc
91140 Villebon S/Yvette
France
Tel. +33 1 6931660-0
Fax +33 1 6931660-1
commercial@kuka.fr
www.kuka.fr

India

KUKA Robotics India Pvt. Ltd.
Office Number-7, German Centre,
Level 12, Building No. - 9B
DLF Cyber City Phase III
122 002 Gurgaon
Haryana
India
Tel. +91 124 4635774
Fax +91 124 4635773
info@kuka.in
www.kuka.in

Italy

KUKA Roboter Italia S.p.A.
Via Pavia 9/a - int.6
10098 Rivoli (TO)
Italy
Tel. +39 011 959-5013
Fax +39 011 959-5141
kuka@kuka.it
www.kuka.it

Japan

KUKA Robotics Japan K.K.
Daiba Garden City Building 1F
2-3-5 Daiba, Minato-ku
Tokyo
135-0091
Japan
Tel. +81 3 6380-7311
Fax +81 3 6380-7312
info@kuka.co.jp

Korea

KUKA Robotics Korea Co. Ltd.
RIT Center 306, Gyeonggi Technopark
1271-11 Sa 3-dong, Sangnok-gu
Ansan City, Gyeonggi Do
426-901
Korea
Tel. +82 31 501-1451
Fax +82 31 501-1461
info@kukakorea.com

Malaysia

KUKA Robot Automation Sdn Bhd
South East Asia Regional Office
No. 24, Jalan TPP 1/10
Taman Industri Puchong
47100 Puchong
Selangor
Malaysia
Tel. +60 3 8061-0613 or -0614
Fax +60 3 8061-7386
info@kuka.com.my

Mexico

KUKA de Mexico S. de R.L. de C.V.
Rio San Joaquin #339, Local 5
Colonia Pensil Sur
C.P. 11490 Mexico D.F.
Mexico
Tel. +52 55 5203-8407
Fax +52 55 5203-8148
info@kuka.com.mx

Norway

KUKA Sveiseanlegg + Roboter
Bryggeveien 9
2821 Gjövik
Norway
Tel. +47 61 133422
Fax +47 61 186200
geir.ulsrud@kuka.no

Austria

KUKA Roboter Austria GmbH
Vertriebsbüro Österreich
Regensburger Strasse 9/1
4020 Linz
Austria
Tel. +43 732 784752
Fax +43 732 793880
office@kuka-roboter.at
www.kuka-roboter.at

Poland

KUKA Roboter Austria GmbH
Spółka z ograniczoną odpowiedzialnością
Oddział w Polsce
Ul. Porcelanowa 10
40-246 Katowice
Poland
Tel. +48 327 30 32 13 or -14
Fax +48 327 30 32 26
ServicePL@kuka-roboter.de

Portugal

KUKA Sistemas de Automatización S.A.
Rua do Alto da Guerra n° 50
Armazém 04
2910 011 Setúbal
Portugal
Tel. +351 265 729780
Fax +351 265 729782
kuka@mail.telepac.pt

Russia

OOO KUKA Robotics Rus
Webnaja ul. 8A
107143 Moskau
Russia
Tel. +7 495 781-31-20
Fax +7 495 781-31-19
kuka-robotics.ru

Sweden

KUKA Svetsanläggningar + Robotar AB
A. Odhners gata 15
421 30 Västra Frölunda
Sweden
Tel. +46 31 7266-200
Fax +46 31 7266-201
info@kuka.se

Switzerland

KUKA Roboter Schweiz AG
Industriestr. 9
5432 Neuenhof
Switzerland
Tel. +41 44 74490-90
Fax +41 44 74490-91
info@kuka-roboter.ch
www.kuka-roboter.ch

Spain

KUKA Robots IBÉRICA, S.A.
Pol. Industrial
Torrent de la Pastera
Carrer del Bages s/n
08800 Vilanova i la Geltrú (Barcelona)
Spain
Tel. +34 93 8142-353
Fax +34 93 8142-950
Comercial@kuka-e.com
www.kuka-e.com

South Africa

Jendamark Automation LTD (Agency)
76a York Road
North End
6000 Port Elizabeth
South Africa
Tel. +27 41 391 4700
Fax +27 41 373 3869
www.jendamark.co.za

Taiwan

KUKA Robot Automation Taiwan Co., Ltd.
No. 249 Pujong Road
Jungli City, Taoyuan County 320
Taiwan, R. O. C.
Tel. +886 3 4331988
Fax +886 3 4331948
info@kuka.com.tw
www.kuka.com.tw

Thailand

KUKA Robot Automation (M)SdnBhd
Thailand Office
c/o Maccall System Co. Ltd.
49/9-10 Soi Kingkaew 30 Kingkaew Road
Tt. Rachatheva, A. Bangpli
Samutprakarn
10540 Thailand
Tel. +66 2 7502737
Fax +66 2 6612355
atika@ji-net.com
www.kuka-roboter.de

Czech Republic

KUKA Roboter Austria GmbH
Organisation Tschechien und Slowakei
Sezemická 2757/2
193 00 Praha
Horní Počernice
Czech Republic
Tel. +420 22 62 12 27 2
Fax +420 22 62 12 27 0
support@kuka.cz

Hungary

KUKA Robotics Hungaria Kft.
Fö út 140
2335 Taksony
Hungary
Tel. +36 24 501609
Fax +36 24 477031
info@kuka-robotics.hu

USA

KUKA Robotics Corp.
22500 Key Drive
Clinton Township
48036
Michigan
USA
Tel. +1 866 8735852
Fax +1 586 5692087
info@kukarobotics.com
www.kukarobotics.com

UK

KUKA Automation + Robotics
Hereward Rise
Halesowen
B62 8AN
UK
Tel. +44 121 585-0800
Fax +44 121 585-0900
sales@kuka.co.uk

Index