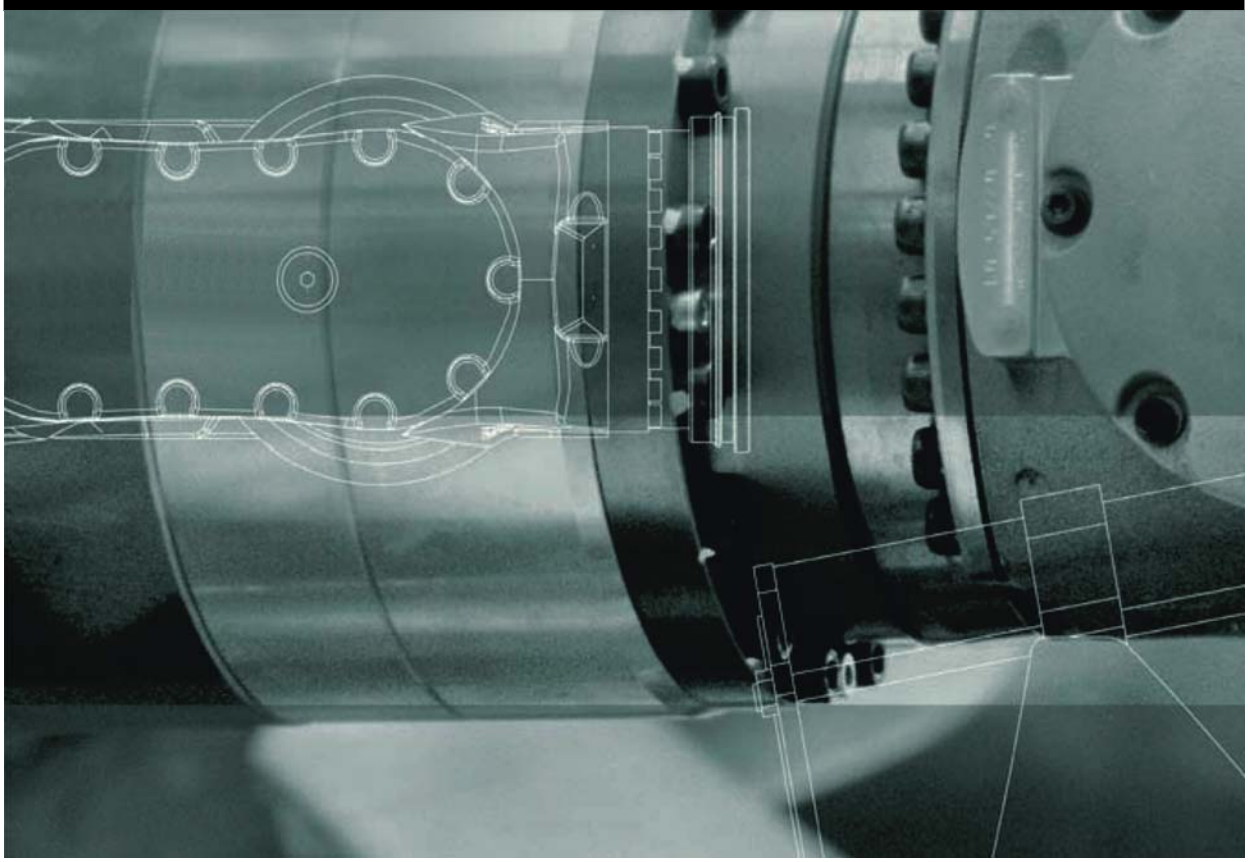
KUKA

KUKA System Software

KUKA Roboter GmbH

KUKA System Software 5.5

Operating and Programming Instructions for System Integrators



Issued: 28.10.2010

Version: KSS 5.5 SI V2 en

KUKA Roboter 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 5.5 SI en
Bookstructure:	KSS 5.5 SI V4.10
Label:	KSS 5.5 SI V2 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

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

Warnings marked with this pictogram are relevant to safety and **must** be observed.

Danger!
This warning means that death, severe physical injury or substantial material damage **will** occur, if no precautions are taken.

Warning!
This warning means that death, severe physical injury or substantial material damage **may** occur, if no precautions are taken.

Caution!
This warning means that minor physical injuries or minor material damage **may** occur, if no precautions are taken.

Notes

Notes marked with this pictogram contain tips to make your work easier or references to further information.

Tips to make your work easier or references 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 Overview of the industrial robot

The industrial robot consists of the following components:

- Manipulator
- Robot controller
- Teach pendant
- Connecting cables
- Software
- Options, accessories

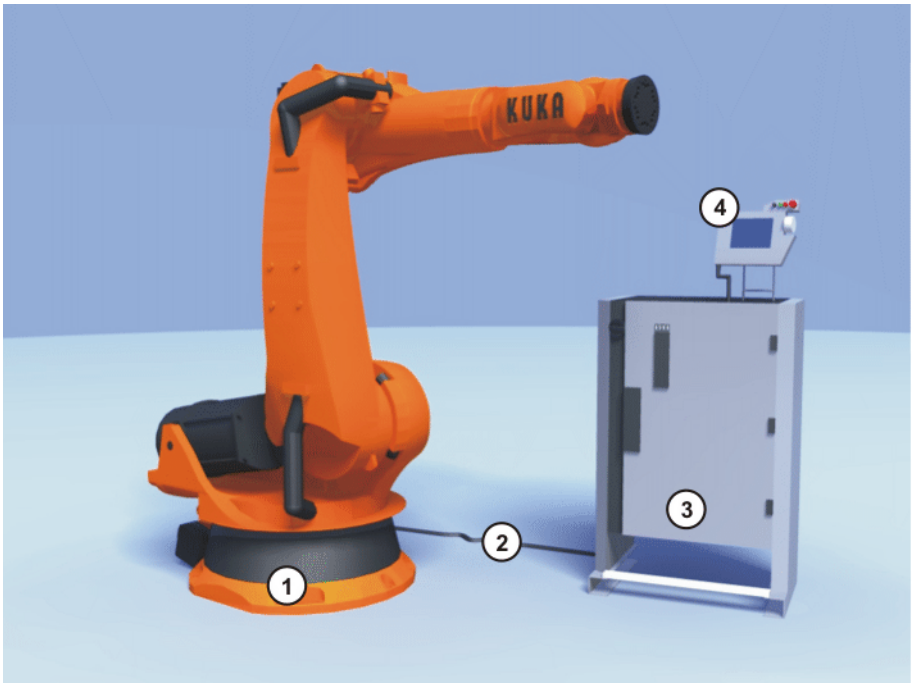


Fig. 2-1: Example of an industrial robot

1	Manipulator	3	Robot controller
2	Connecting cables	4	Teach pendant

2.2 Overview of the software components

Overview

The following software components are used:

- KUKA System Software 5.5
- Windows XP embedded 2.x incl. Service Pack 2

It is not possible to upgrade from Windows Service Pack 1 to Windows Service Pack 2.

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.

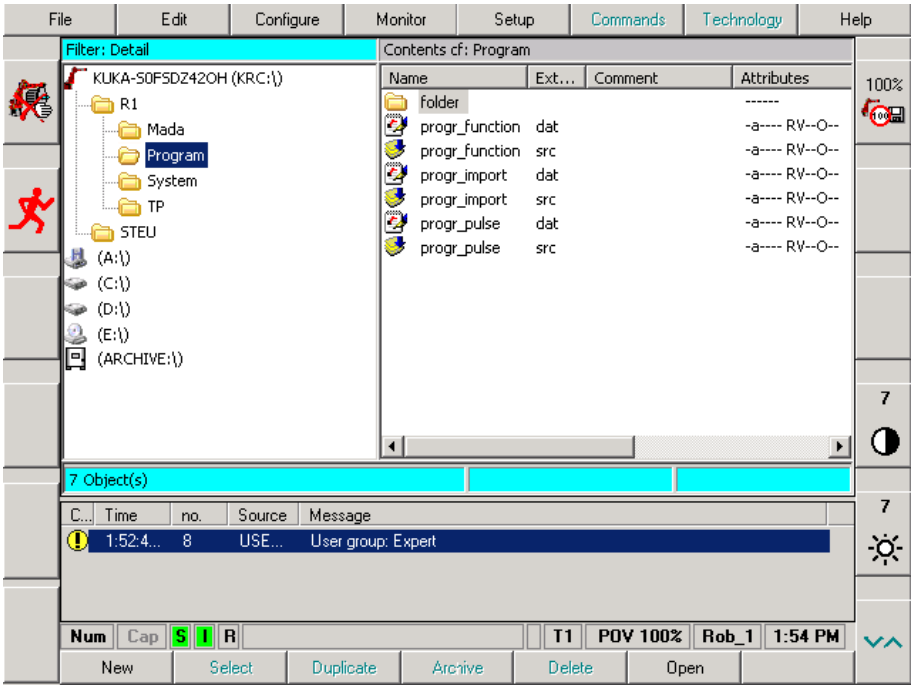


Fig. 2-2: KUKA.HMI user interface

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 component.

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.
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 instruction 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 ex-

ist. 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
Maintenance			x

Tasks	Operator	Programmer	System integrator
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.

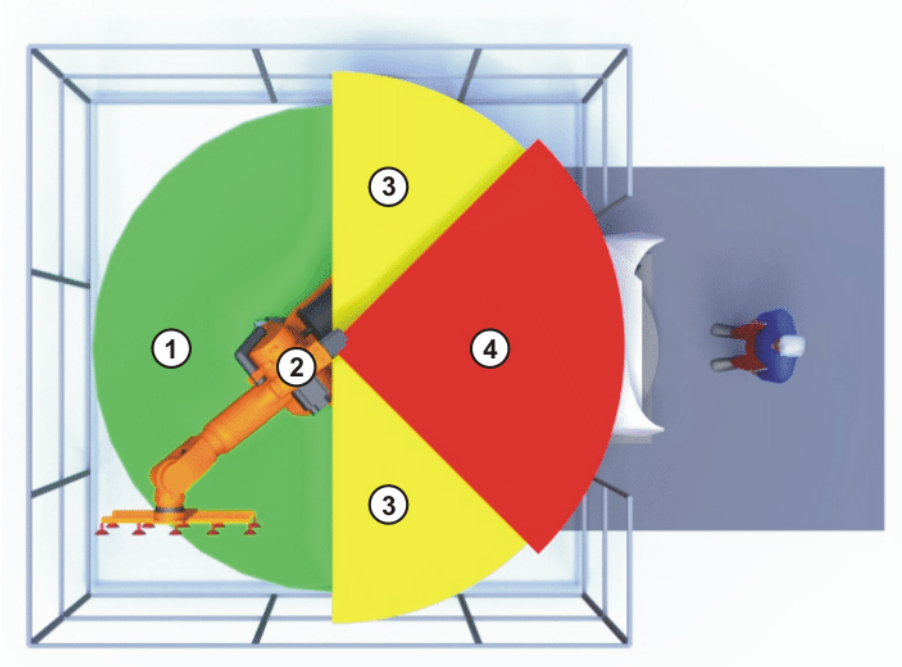


Fig. 3-1: Example of axis range A1

1	Workspace	3	Stopping distance
2	Manipulator	4	Safety zone

3.4 Triggers for stop reactions

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
- External enabling device
- Local safety stop via qualifying input
- RoboTeam: disabling of robots that have not been selected

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

Warning!
If these conditions are not met, KUKA Roboter GmbH must be contacted.

Danger!
In the absence of functional 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
- External enabling device
- Drives OFF
- Drives ON
- Operating modes
- Qualifying inputs

The ESC safety logic monitors the following outputs:

- Operating mode
- Drives ON
- 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.

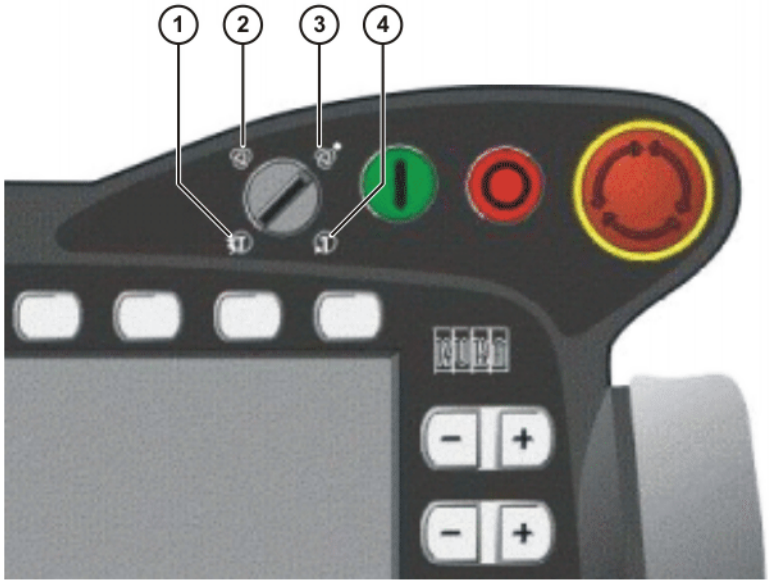


Fig. 3-2: Mode selector switch

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 possi-

ble. 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 on every operator panel and anywhere else it may be necessary to trigger an EMERGENCY STOP. The system integrator is responsible for ensuring this. 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

3.5.8 External enabling device

External enabling devices are required if it is necessary for more than one person to be in the danger zone of the industrial robot. They can be connected via the peripheral interface on the robot controller.

External enabling devices are not included in the scope of supply of the industrial robot.

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 Mechanical end stops

The axis ranges of main axes A1 to A3 and wrist axis A5 of the manipulator are limited by means of mechanical end stops with buffers.

Additional mechanical end stops can be installed on the external axes.

Warning!
If the manipulator or an external axis hits an obstruction or a buffer on the mechanical end stop or axis range limitation, this can result in material damage to the industrial robot. KUKA Roboter GmbH must be consulted before the industrial robot is put back into operation (>>> 15 "KUKA Service" Page 387). The affected buffer must be replaced with a new one before operation of the industrial robot is resumed. If a manipulator (or external axis) collides with a buffer at more than 250 mm/s, the manipulator (or external axis) must be exchanged or recommissioning must be carried out by KUKA Roboter GmbH.

3.6.4 Mechanical axis range limitation (optional)

Some manipulators can be fitted with mechanical axis range limitation in axes A 1 to A 3. The adjustable axis range limitation systems restrict the working range to the required minimum. This increases personal safety and protection of the system.

In the case of manipulators that are not designed to be fitted with mechanical axis range limitation, the workspace must be laid out in such a way that there is no danger to persons or material property, even in the absence of mechanical axis range limitation.

If this is not possible, the workspace must be limited by means of photoelectric barriers, photoelectric curtains or obstacles on the system side. There must be no shearing or crushing hazards at the loading and transfer areas.

This option is not available for all robot models. Information on specific robot models can be obtained from KUKA Roboter GmbH.

3.6.5 Axis range monitoring (optional)

Some manipulators can be fitted with dual-channel axis range monitoring systems in main axes A1 to A3. The positioner axes may be fitted with additional axis range monitoring systems. The safety zone for an axis can be adjusted and monitored using an axis range monitoring system. This increases personal safety and protection of the system.

This option is not available for all robot models. Information on specific robot models can be obtained from KUKA Roboter GmbH.

3.6.6 Release device (optional)

Description

The release device can be used to move the manipulator manually after an accident or malfunction. The release device can be used for the main axis drive motors and, depending on the robot variant, also for the wrist axis drive motors. It is only for use in exceptional circumstances and emergencies (e.g. for freeing people).

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

Procedure

1. Switch off the robot controller and secure it (e.g. with a padlock) to prevent unauthorized persons from switching it on again.
2. Remove the protective cap from the motor.
3. Push the release device onto the corresponding motor and move the axis in the desired direction.

   The directions are indicated with arrows on the motors. It is necessary to overcome the resistance of the mechanical motor brake and any other loads acting on the axis.

Warning!
Moving an axis with the release device can damage the motor brake. This can result in personal injury and material damage. After using the release device, the affected motor must be exchanged.

3.6.7 KCP coupler (optional)

The KCP coupler allows the KCP to be connected and disconnected with the robot controller running.

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.

Further information is contained in the operating instructions or installation instructions for the robot controller.

3.6.8 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:

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

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

3.6.9 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!

Warning!
The motors reach temperatures during operation which can cause burns to the skin. Contact should 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:

- E-STOP pushbutton, mode selector switch

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

  If relays are switched on at an output, they must be switched off at least once every 6 months in order to detect any malfunction.

Additional checks are required during start-up and recommissioning.

(>>> 3.8.4 "Start-up and recommissioning" Page 33)

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.

Warning!
If additional components (e.g. cables), that 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.

Caution!
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.

Interruptions/ cross-connections

Interruptions or cross-connections affecting safety functions and not detected by the robot controller or SafeRDC must either be precluded (e.g. by the construction) or detected by the customer (e.g. by means of a PLC or by testing the outputs).

Recommendation: design the construction in such a way as to preclude cross-connections. For this, observe the remarks in EN ISO 13849-2, tables D.5, D.6 and D.7.

Overview: possible cross-connections that are not detected by the robot controller or SafeRDC

Cross-connection	Possible in the case of …
Cross-connection to 0 V	■ ESC output Drives ON ■ ESC output E-STOP
Cross-connection to 24 V	■ ESC output Drives ON ■ ESC output E-STOP ■ ESC output Operating Mode ■ SafeRDC inputs
Cross-connection between the contacts of an output	■ ESC output Drives ON ■ ESC output E-STOP ■ ESC output Operating Mode
Cross-connection between the contacts of different outputs	
Cross-connection of an ESC output with an ESC input	
Cross-connection between the channels of different ESC inputs	ESC inputs
Cross-connection between 2 SafeRDC inputs	SafeRDC inputs
Cross-connection of a SafeRDC output with a SafeRDC input	SafeRDC outputs, SafeRDC inputs

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.

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.

Warning!
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 operation-

al 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.

Counterbalancing system

Some robot variants are equipped with a hydropneumatic, spring or gas cylinder counterbalancing system.

The hydropneumatic and gas cylinder counterbalancing systems are pressure equipment and, as such, are subject to obligatory equipment monitoring. Depending on the robot variant, the counterbalancing systems correspond to category 0, II or III, fluid group 2, of the Pressure Equipment Directive.

The user must comply with the applicable national laws, regulations and standards pertaining to pressure equipment.

Inspection intervals in Germany in accordance with Industrial Safety Order, Sections 14 and 15. Inspection by the user before commissioning at the installation site.

The following safety measures must be carried out when working on the counterbalancing system:

- The manipulator assemblies supported by the counterbalancing systems must be secured.
- Work on the counterbalancing systems must only be carried out by qualified personnel.

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

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
97/23/EC	Pressure Equipment Directive: Directive 97/23/EC of the European Parliament and of the Council of 29 May 1997 on the approximation of the laws of the Member States concerning pressure equipment	1997
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 for 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
EN ISO 10218-1	Industrial robots: Safety	2008
EN 614-1	Safety of machinery: Ergonomic design principles - Part 1: Terminology and general principles	2006

Name	Definition	Edition
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 56)
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 47)
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 42)
Numeric keypad	(>>> 4.1.3 "Numeric keypad" Page 43)
Softkeys	(>>> 4.2.1 "Status keys, menu keys, softkeys" Page 47)
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 47)
Menu keys	(>>> 4.2.1 "Status keys, menu keys, softkeys" Page 47)

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 50).
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 50).
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 50)

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	Rating plate	4	Enabling switch
2	Start key	5	Enabling switch
3	Enabling switch		

Description

Element	Description
Rating 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.

4.1.5 KCP coupler

The KCP coupler is an optional component of the KR C2 edition2005 robot controller. The KCP coupler allows the KCP to be connected and disconnected with the robot controller running.

If the robot is controlled via a PLC and the KCP is not used, it may be useful to uncouple the KCP. This is particularly the case in systems with large numbers of robots: The unused KCPs can be removed from the system, thereby improving the transparency of the system.

4.1.5.1 Visualization of the KCP coupler (option)

Description

If the robot controller is operated with a detachable KCP, the following system variables must be visualized:

- $T1 (T1 mode)
- $T2 (T2 mode)
- $EXT (External mode)
- $AUT (Automatic mode)
- $ALARM_STOP
- $PRO_ACT (program active)

The display can be configured using I/Os or a PLC. The system variables can be configured in the file: STEU/$MACHINE.DAT.

Warning!
If the KCP is disconnected, the system can no longer be deactivated by means of the EMERGENCY STOP button on the KCP. An external E-STOP must be connected to interface X11 to prevent personal injury and material damage.

4.1.5.2 Display and operator control elements of the KCP coupler (optional)

Overview

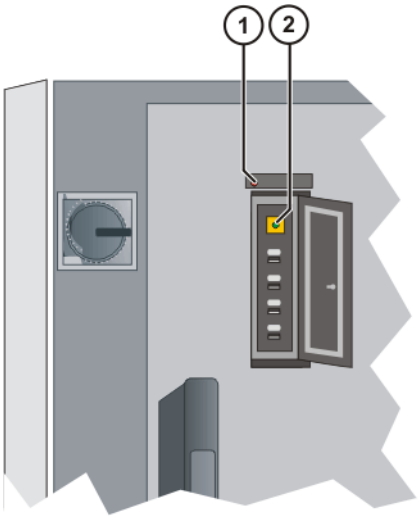


Fig. 4-5: KCP coupler LEDs and request button

1 Fault LED (red), KCP coupler
2 Request button with request LED (green)

4.1.5.3 Uncoupling the KCP

Procedure

1. Press the request button for at least 1 s.
   The green request LED flashes.
   The KCP is switched off (display goes dark).

Caution!
The KCP must not be disconnected without pressing the request button. If the KCP is disconnected without the request button being pressed, an EMERGENCY STOP is triggered.

2. Disconnect the KCP within 60 s.

Caution!
The KCP with EMERGENCY STOP is deactivated for the request time of 60 s. The EMERGENCY STOP on the KCP is not activated during this time.

3. Remove the KCP from the system.

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.

4.1.5.4 Coupling the KCP

Precondition

- The KCP variant to be coupled must be the same as that which was uncoupled.

Procedure

1. Set the operating mode on the KCP to the same operating mode as on the robot controller. The operating mode display depends on the specific application. (>>> 4.1.5.1 "Visualization of the KCP coupler (option)" Page 45)

If the KCP is connected with the wrong operating mode selected, the robot controller switches to the operating mode set on the KCP.

2. Couple the KCP to the robot controller.
   The request LED flashes quickly.
   Once coupling has been completed, the request LED lights up and the KCP display shows the user interface. The robot controller can once again be operated via the KCP.

4.2 KUKA.HMI user interface

4.2.1 Status keys, menu keys, softkeys

Overview

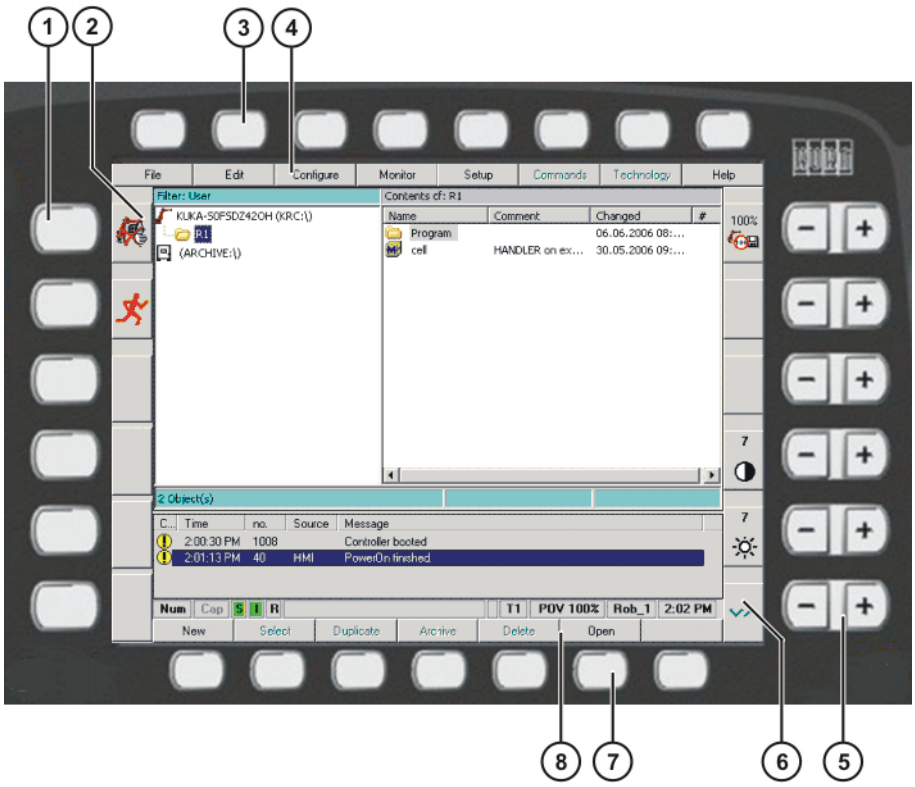


Fig. 4-6: Status keys, menu keys and softkeys in the user interface

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

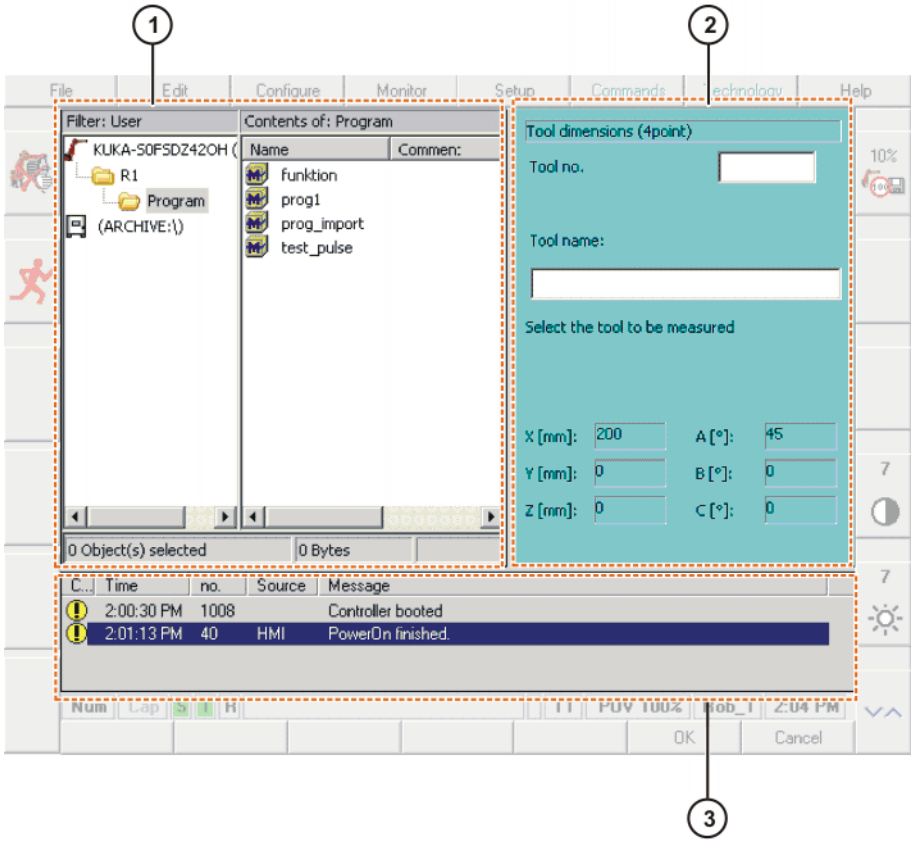


Fig. 4-7: Windows in the user interface

1 Main window
2 Option window
3 Message 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.

Fig. 4-8: Example of an input box

List box

A parameter can be selected from a list.

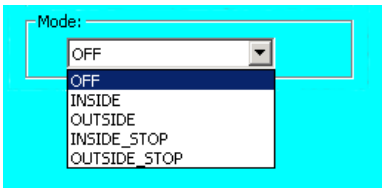


Fig. 4-9: Example of an opened list box

Check box

One or more options can be selected.

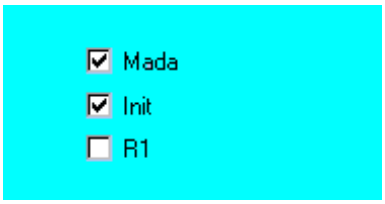


Fig. 4-10: Example of a check box

Option box

One option can be selected.

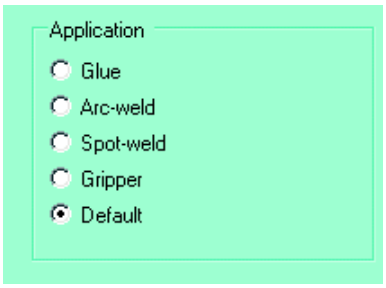


Fig. 4-11: Example of an option box

Slider

A value on a scale can be set.

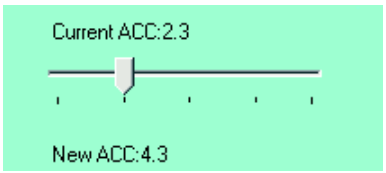


Fig. 4-12: Example of a slider

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.

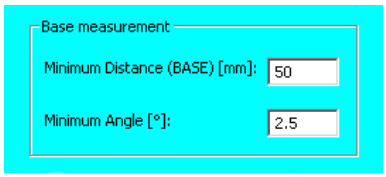


Fig. 4-13: Example of a group

4.2.4 Status bar

Overview

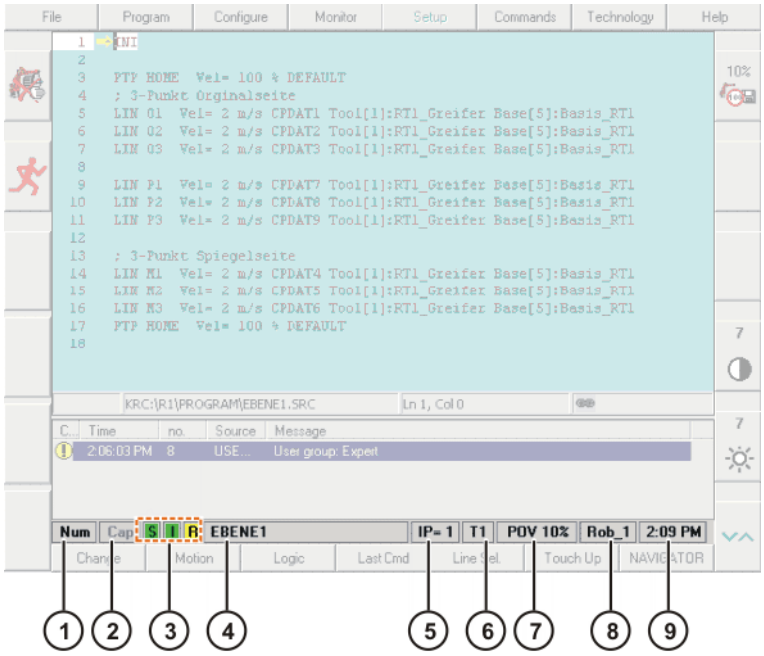


Fig. 4-14: Status bar in the user interface

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

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

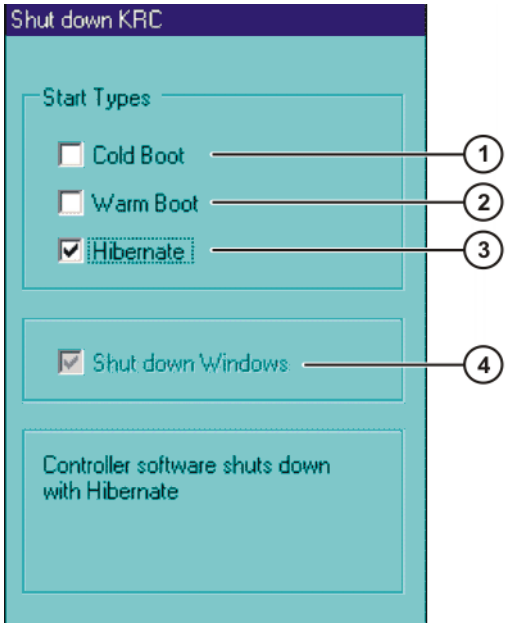


Fig. 4-15: Option window “Shut down KRC”

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

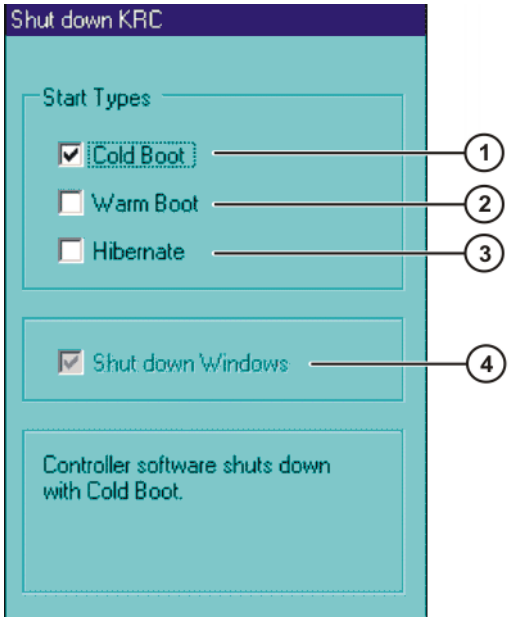


Fig. 4-16: Option window “Start types”

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
Danish	Polish
German	Portuguese
English	Russian
Finnish	Swedish
French	Spanish
Italian	Czech

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.

(>>> 6.6 "Changing the password" Page 119)

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.

(>>> 6.18 "Setting up a new user group and password" Page 151)

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.
(>>> 6.17 "Configuring the log-in" Page 147)

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.

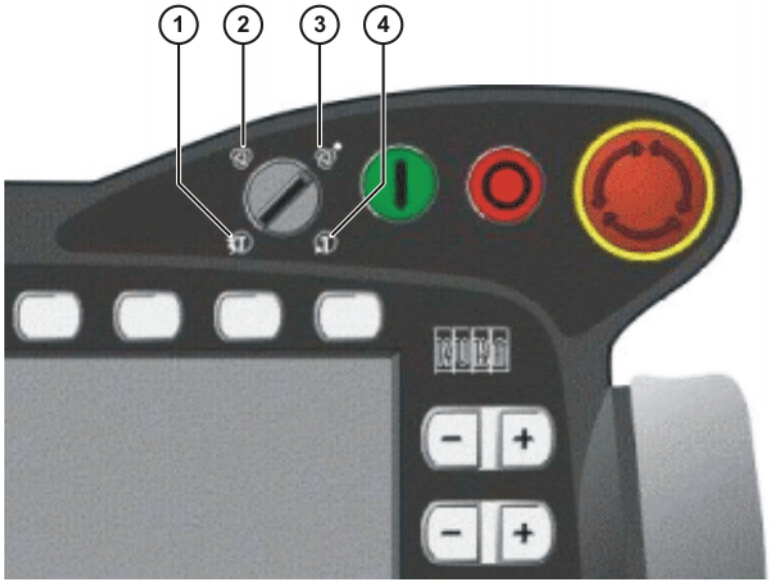


Fig. 4-17: Mode selector switch

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

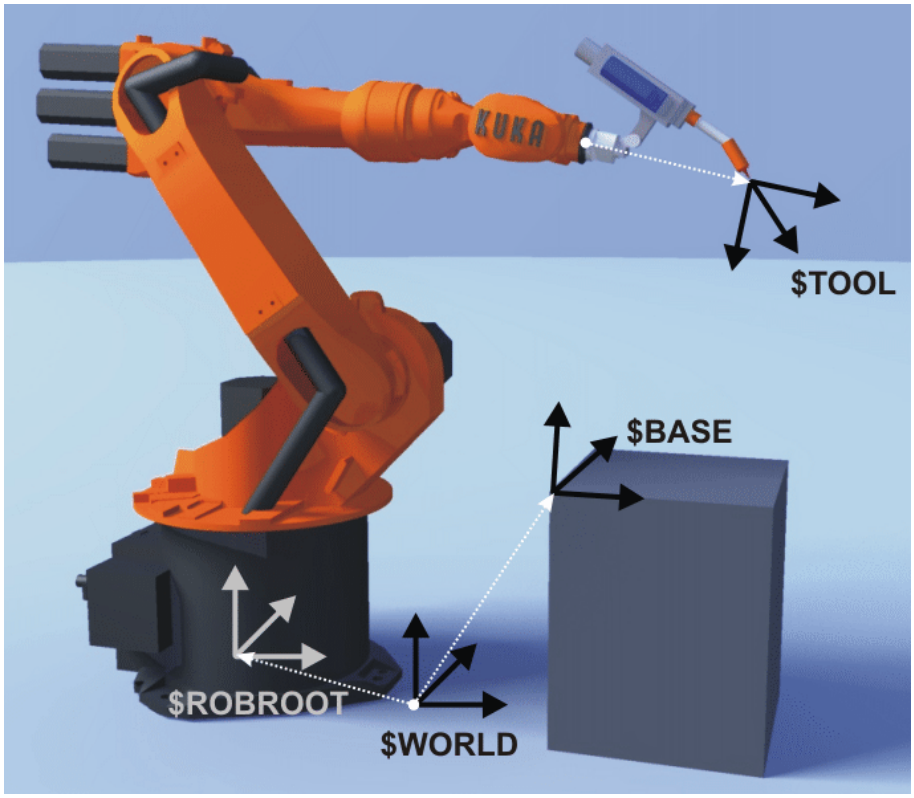


Fig. 4-18: Overview of coordinate systems

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.

(>>> 5.7.3 "Base calibration" Page 102)

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.

(>>> 5.7.1 "Tool calibration" Page 91)

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 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.

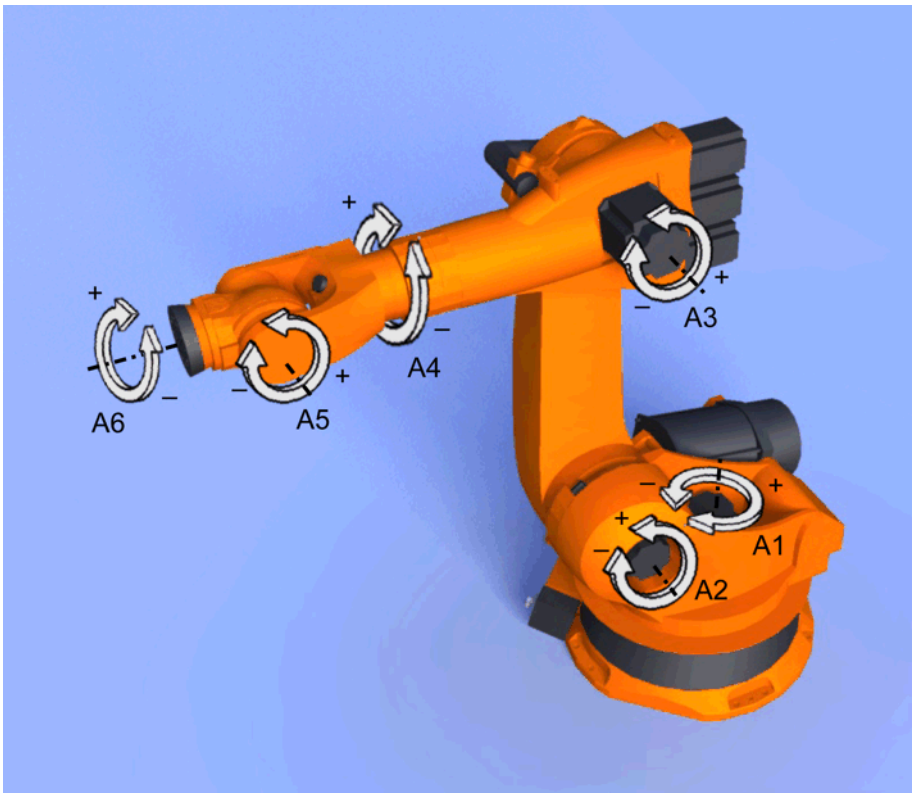


Fig. 4-19: Axis-specific jogging

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

- Jog keys
- Space Mouse

Overview

	Cartesian jogging	Axis-specific jogging
Jog keys	(>>> 4.14.4 "Cartesian jogging with the jog keys" Page 61)	(>>> 4.14.3 "Axis-specific jogging with the jog keys" Page 60)
Space Mouse	(>>> 4.14.7 "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 with the jog keys

Precondition

- Operating mode T1 or T2.

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. Set jog override.
4. Hold down the enabling switch.
5. 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.

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

4.14.4 Cartesian jogging with the jog keys

Precondition

- Tool and base have been selected.
  (>>> 4.14.2 "Selecting the tool and base" Page 60)
- Operating mode T1 or T2.

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. Set jog override.
4. Hold down the enabling switch.
5. 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.

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

4.14.5 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

Softkey	Description
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

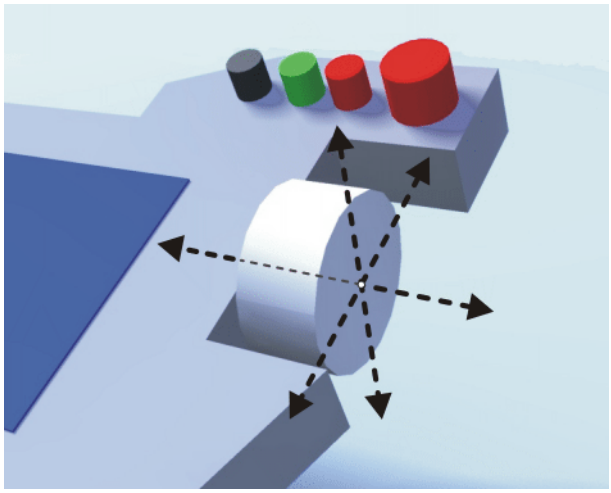


Fig. 4-20: Pushing and pulling the Space Mouse

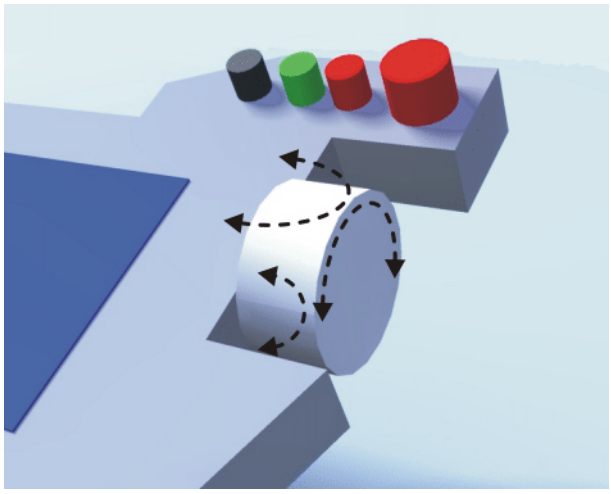


Fig. 4-21: Rotating and tilting the Space Mouse

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.

4.14.6 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.

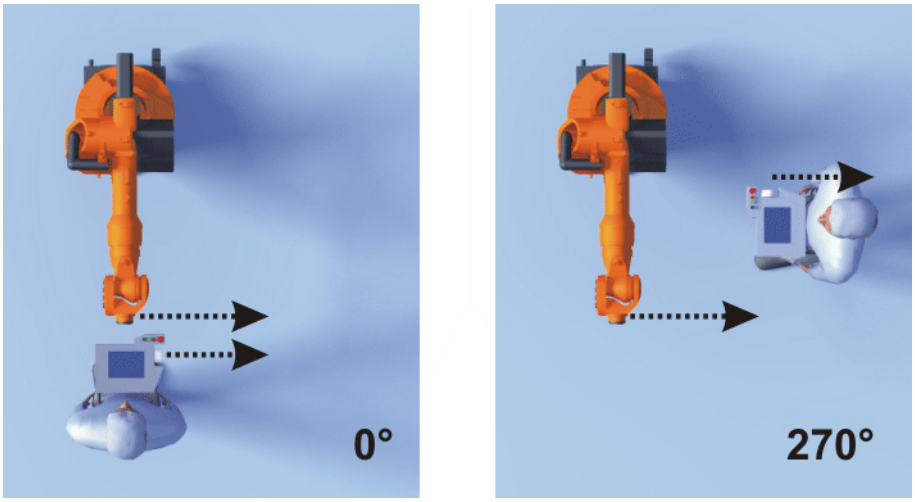


Fig. 4-22: Space Mouse: 0° and 270°

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.

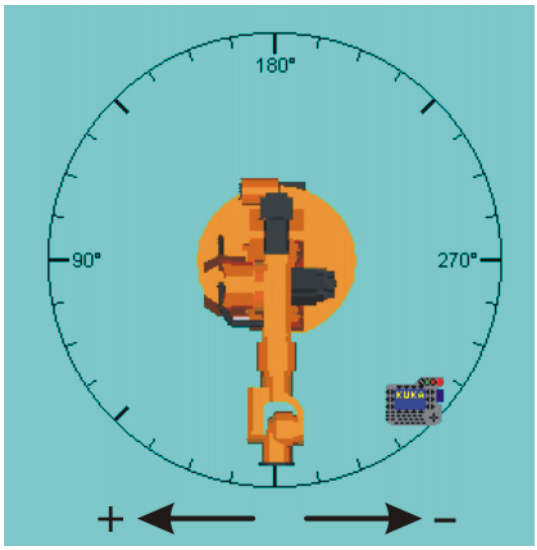


Fig. 4-23: Option window for aligning the Space Mouse

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°.

4.14.7 Cartesian jogging with the Space Mouse

Precondition

- Tool and base have been selected.
  (>>> 4.14.2 "Selecting the tool and base" Page 60)
- The Space Mouse is configured.
  (>>> 4.14.5 "Configuring the Space Mouse" Page 61)
- The alignment of the Space Mouse has been defined.
  (>>> 4.14.6 "Defining the alignment of the Space Mouse" Page 63)
- Operating mode T1 or T2.

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.

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

4.14.8 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
	Increment = 100 mm or 10°
	Increment = 10 mm or 3°
	Increment = 1 mm or 1°
	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 with the jog keys" Page 60)

   (>>> 4.14.4 "Cartesian jogging with the jog keys" Page 61)

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 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. (>>> 6.8 "Configuring workspaces" Page 122)

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.

4.16 Monitor functions

4.16.1 Displaying the actual position

Procedure

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

Description

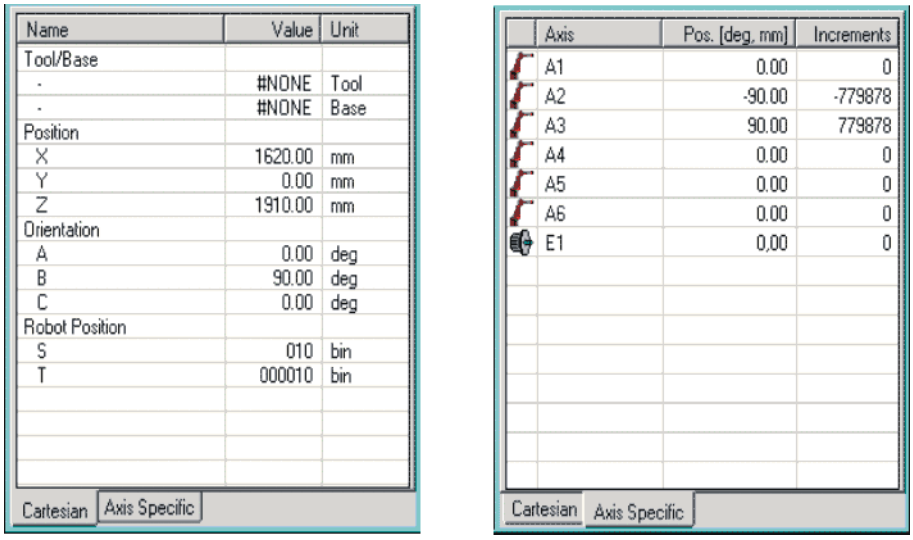


Fig. 4-24: Actual position: Cartesian and axis-specific

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.

4.16.2 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

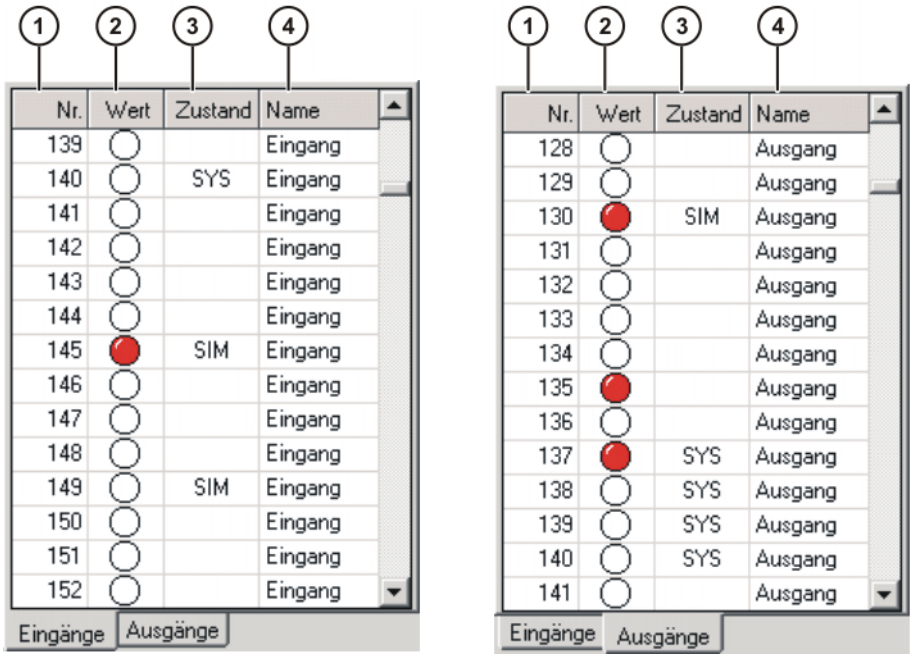


Fig. 4-25: Digital inputs/outputs

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.

4.16.3 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

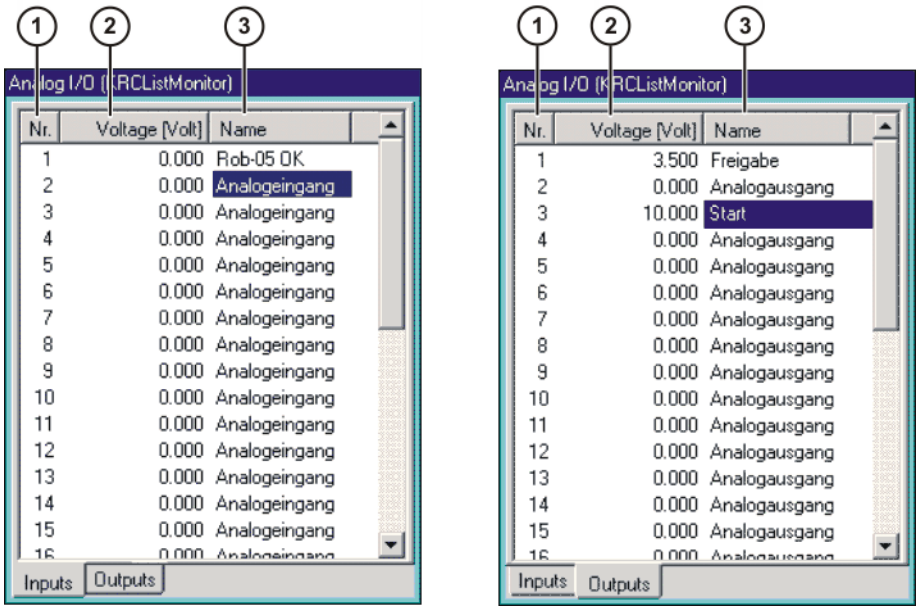


Fig. 4-26: Analog inputs/outputs

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.

4.16.4 Displaying inputs/outputs for Automatic External

Procedure

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

Description

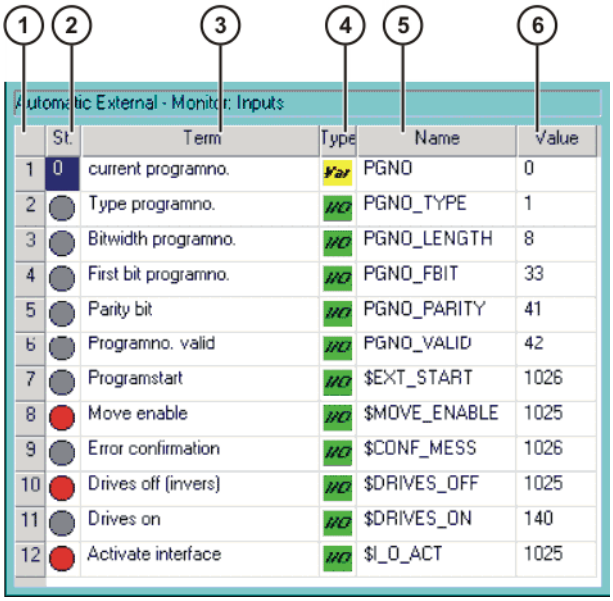


Fig. 4-27: Automatic External inputs (detail view)

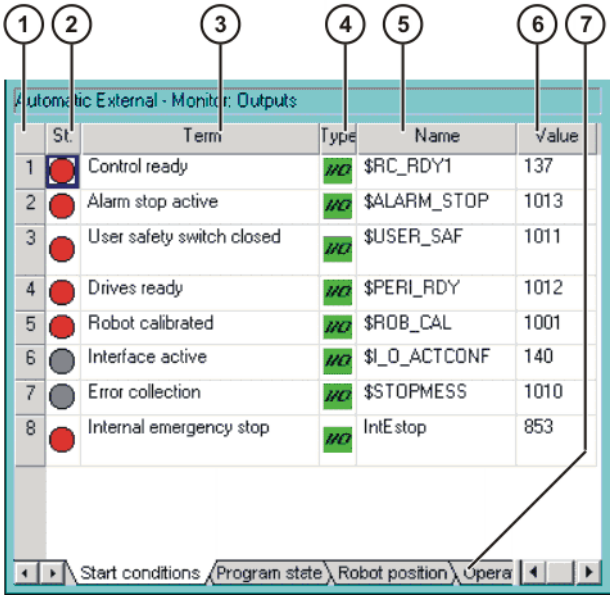


Fig. 4-28: Automatic External outputs (detail view)

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. (>>> 6.20.2 "Configuring Automatic External inputs/outputs" Page 155)
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.

4.16.5 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

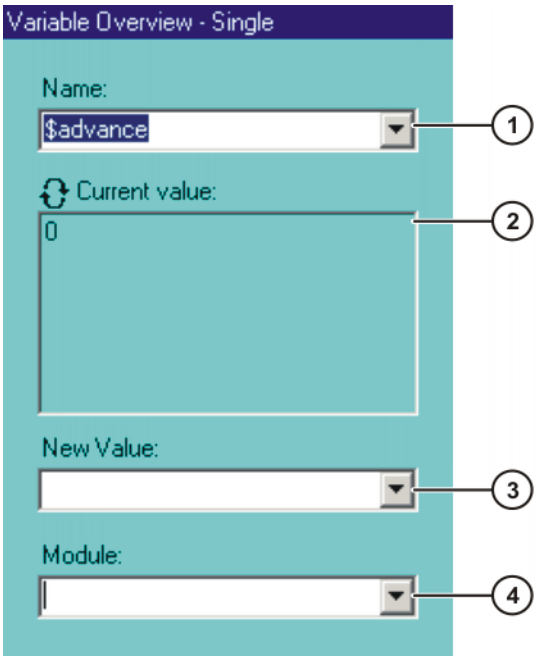


Fig. 4-29: Variable Overview - Single window

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.

4.16.6 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 Overview - Single** window is opened.
2. Enter the following in the **Name** box: =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.

4.16.7 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.

(>>> 6.4 "Configuring the variable overview" Page 117)

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

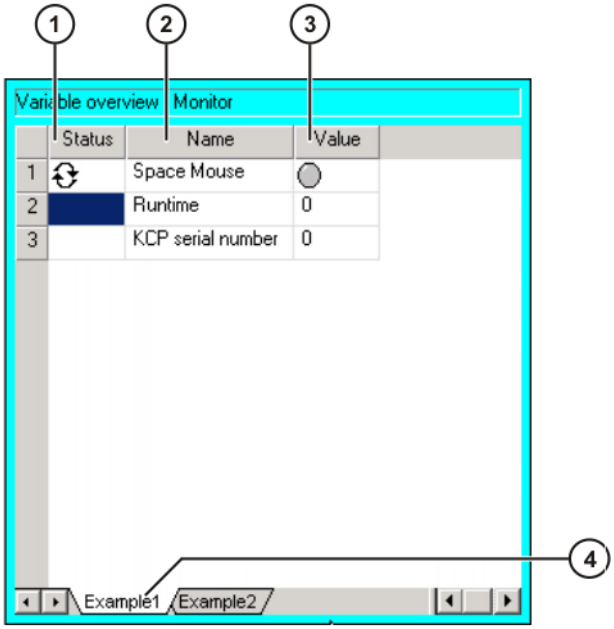


Fig. 4-30: Variable overview - Monitor window

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. (>>> 6.4 "Configuring the variable overview" Page 117) 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.

4.16.8 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.

4.16.9 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.

4.16.10 Displaying robot data

Procedure

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

Description

Data on RDC and hard drive are identical

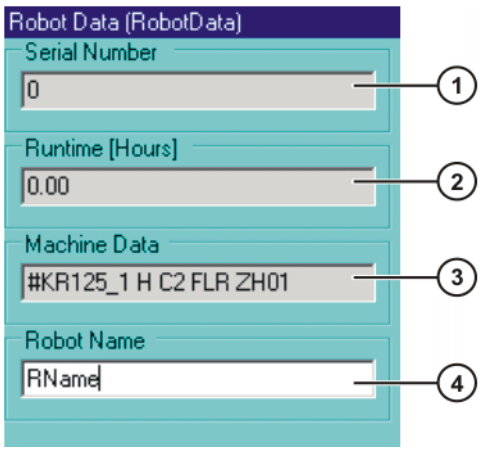


Fig. 4-31: Robot data – data on RDC and hard drive are identical

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 softkeys are available in the user group Expert:

Softkey	Description
Import PID	A tree structure containing the local drives is displayed. It is possible to select a directory and import a PID file. It is only necessary to import PID files manually in special cases. PID files are only relevant in the case of absolutely accurate robots.
Import MAM	This softkey is only available if the robot is of a type that uses a MAM file. A tree structure containing the local drives is displayed. It is possible to select a directory and import a MAM file. It is only necessary to import MAM files manually in special cases.

Data on RDC and hard drive are not identical

If the RDC or the hard drive has been exchanged, the data on the RDC and 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.) The data are also not identical in the case of initial start-up. This is indicated by a message.

The **Robot data** window indicates the different data.

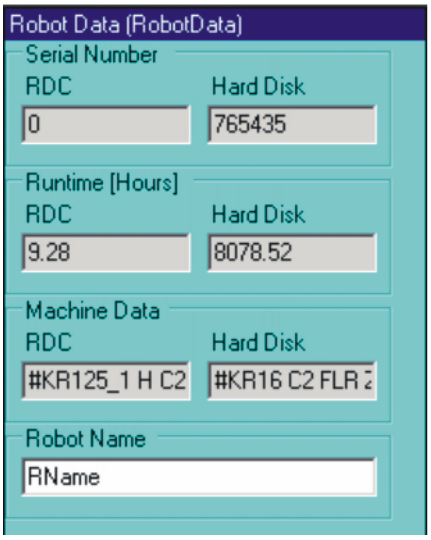


Fig. 4-32: Robot data – data on RDC and hard drive are not identical

The following softkeys are available in the user group Expert:

The **RDC** softkey must be pressed in the case of initial start-up.

Softkey	Description
Hard disk	Transfers the following files from the hard drive to the RDC: ■ PID file (if the robot is an absolutely accurate robot) ■ MAM file (if this robot type uses a MAM file) ■ Serial number ■ Machine data name ■ Operating hours
RDC	Transfers the following files from the RDC to the hard drive: ■ PID file (if the robot is an absolutely accurate robot) ■ MAM file (if this robot type uses a MAM file) ■ 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).

4.16.11 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

This is an overview of the most important steps during start-up. The precise sequence depends on the application, the manipulator type, the technology packages used and other customer-specific circumstances.
For this reason, the overview does not claim to be comprehensive.

This overview refers to the start-up of the industrial robot. The start-up of the overall system is not within the scope of this documentation.

Robot

Step	Description	Information
1	Carry out a visual inspection of the robot.	Detailed information is contained in the operating or assembly instructions for the robot, in the chapter “Start-up and recommissioning”.
2	Install the robot mounting base (mounting base, machine frame mounting or booster frame).	
3	Install the robot.	

Electrical system

Step	Description	Information
4	Carry out a visual inspection of the robot controller.	
5	Make sure that no condensation has formed in the robot controller.	
6	Install the robot controller.	Detailed information is contained in the operating or assembly instructions for the robot controller, in the chapter “Start-up and recommissioning”.
7	Connect the connecting cables.	
8	Connect the KCP.	
9	Establish the equipotential bonding between the robot and the robot controller.	
10	Connect the robot controller to the power supply.	
11	Reverse the battery discharge protection measures.	
12	Configure and connect interface X11. **Note:** If interface X11 has not been wired, the robot cannot be jogged.	
13	Switch on the robot controller.	
14	Check the direction of rotation of the fans.	
15	Check the safety equipment.	Detailed information is contained in the operating or assembly instructions for the robot controller, in the chapter “Safety”.
16	Configure the inputs/outputs between the robot controller and the periphery.	Detailed information can be found in the field bus documentation.

Software

Step	Description	Information
17	Check the machine data.	(>>> 5.2 "Checking the machine data" Page 78)
18	Transfer data from the RDC to the hard drive.	(>>> 5.3 "Transfer data from the RDC to the hard drive." Page 79)
19	Master the robot without a load.	(>>> 5.6.3.1 "First mastering with the EMT" Page 85)
20	Only for palletizing robots with 6 axes: Activate palletizing mode.	(>>> 5.5 "Activating palletizing mode" Page 80)
21	Mount the tool and master the robot with a load.	(>>> 5.6.3.2 "Teach offset" Page 85)
22	Check the software limit switches and adapt them if required.	
23	Calibrate tool. In the case of a fixed tool: calibrate external TCP.	(>>> 5.7.1 "Tool calibration" Page 91) (>>> 5.7.2 "Fixed tool calibration" Page 98)
24	Enter load data.	(>>> 5.8 "Load data" Page 111)
25	Calibrate base (optional). In the case of a fixed tool: calibrate workpiece (optional).	(>>> 5.7.3 "Base calibration" Page 102) (>>> 5.7.2 "Fixed tool calibration" Page 98)
26	If the robot is to be controlled from a higher-level controller: configure Automatic External interface.	(>>> 6.20 "Configuring Automatic External" Page 154)

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 robot. Furthermore, the long text names can be updated in application programs.
(>>> 5.9 "Transferring long text names" Page 113)

Accessories

Precondition: the robot is ready to move, i.e. the software start-up has been carried out up to and including the item “Master the robot without load”.

Description	Information
Optional: install axis range limitation systems. Adapt software limit switches.	Detailed information can be found in the axis range limitation documentation.
Optional: install and adjust axis range monitoring, taking the programming into consideration.	Detailed information can be found in the axis range monitoring documentation.
Optional: install and adjust external energy supply system, taking the programming into consideration.	Detailed information can be found in the energy supply system documentation.
Positionally accurate robot option: check data.	

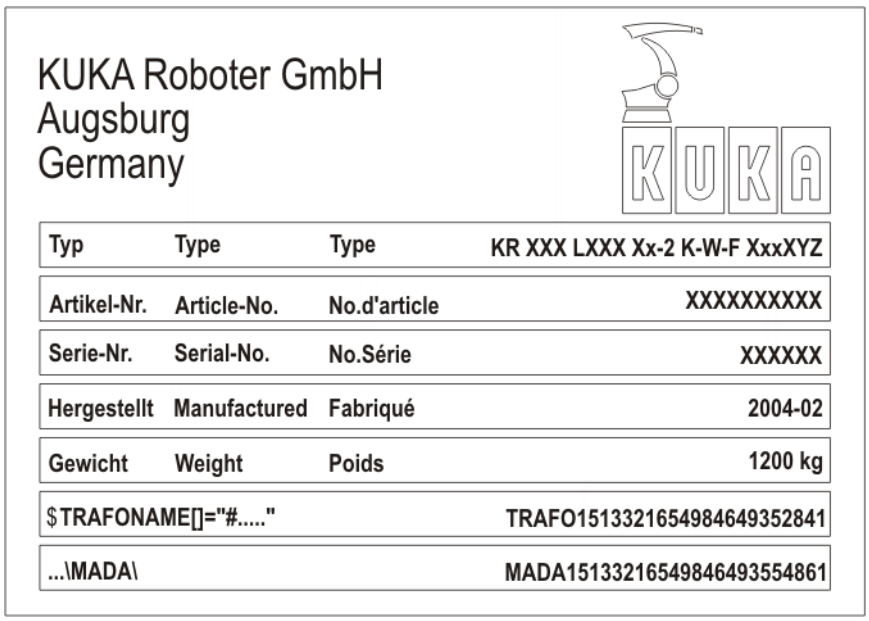
5.2 Checking the machine data

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 only the machine data from the CD with the KSS, that is also installed, are used.

Warning!
The 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 Transfer data from the RDC to the hard drive.

Description

In the case of initial start-up or if the hard drive has been exchanged, the data on the RDC and 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 from the RDC 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:
   - PID file (if the robot is an absolutely accurate robot)
   - MAM file (if this robot type uses a MAM file)
   - 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 Transferring data from the hard drive to the RDC (after exchange of RDC)

Description

If the RDC has been exchanged, this is indicated by the following message when the robot controller and KSS have been started: "*Data of RDC and Hard Disk inconsistent! Check robot data!*". The data must now be transferred from the hard drive to the RDC.

Precondition

- Expert user group

Procedure

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

   The following data are transferred:

   - PID file (if the robot is an absolutely accurate robot)
   - MAM file (if this robot type uses a MAM file)
   - Serial number
   - Machine data name
   - Operating hours

5.5 Activating palletizing mode

Description

Only relevant for palletizing robots with 6 axes!

In the case of palletizing robots with 6 axes, palletizing mode is deactivated by default and must be activated. When palletizing mode is active, A4 is locked at 0° and the mounting flange is parallel to the floor.

Precondition

- The robot is mastered.
- There is no load on the robot; i.e. there is no tool, workpiece or supplementary load mounted.

Procedure

- Activate palletizing mode in the program as follows:

```
$PAL_MODE = TRUE
```

Alternative procedure

1. Set $PAL_MODE to TRUE via the variable correction function.
2. The following message is displayed: *Palletizing mode: Move axis A4 [direction] into position*.

   Move A4 in the direction specified in the message [+/-].
3. Once A4 has reached its position, the following message is displayed: *Palletizing mode: Move axis A5 [direction] into position*.

   Move A5 in the direction specified in the message [+/-].

Restrictions

- After every cold restart of the robot controller, $PAL_MODE is automatically set to FALSE.

Recommendation: Integrate $PAL_MODE = TRUE into the initialization section of all programs for the palletizing robot.

- In the case of robots with palletizing mode active, payload determination with KUKA.LoadDetect is not possible.

Caution!
In the case of robots with palletizing mode active, payload determination with KUKA.LoadDetect must not be carried out. Physical injuries or damage to property may result.

- If palletizing mode is active, the robot cannot be mastered. If mastering is nonetheless required, proceed as follows:
  a. Remove all loads from the robot.
  b. Set $PAL_MODE to FALSE via the variable correction function.
  c. Master the robot.
  d. Set $PAL_MODE to TRUE.
     (Not necessary if $PAL_MODE = TRUE is in the initialization section of all programs for the palletizing robot.)
  e. Move the robot to the palletizing position.
  f. Re-attach all loads to the robot.

5.6 Mastering

Overview

Every robot must be mastered. Only if the robot has been mastered can it move to programmed positions and be moved using Cartesian coordinates. During mastering, the mechanical position and the electronic position of the robot are aligned. For this purpose, the robot is moved to a defined mechanical position, the mastering position. The encoder value for each axis is then saved.

The mastering position is similar, but not identical, for all robots. The exact positions may even vary between individual robots of a single robot type.

Fig. 5-2: Mastering position – approximate position

A robot must be mastered in the following cases:

Case	Comments
During commissioning	- - -
After maintenance work during which the robot loses its mastering, e.g. exchange of motor or RDC	(>>> 5.6.6 "Reference mastering" Page 89)
When the robot has been moved without the robot controller (e.g. with the release device)	- - -
After exchanging a gear unit	Before carrying out a new mastering procedure, the old mastering data must first be deleted! Mastering data are deleted by manually unmastering the axes. (>>> 5.6.8 "Manually unmastering axes" Page 90)
After an impact with an end stop at more than 250 mm/s	
After a collision	

The mastering is automatically saved in the following cases:

- Operating mode T1 or T2: the brakes of all axes are applied.
- Operating mode AUT or AUT EXT:
  - The program is stopped by pressing the STOP key.
  - EMERGENCY STOP button is pressed.
  - Drives are switched off.

The saved data are checked. If they do not match the last backup, the flash of the RDC is defective. In this case, an error message is displayed. Additionally, the output of the system variable $RDC_FLASH_DEFECT is set if an output has been configured. The system variable can be found in the file: KRC:\STEU:\MADA:\$machine.dat. No output is configured as standard.

```
SIGNAL $RDC_FLASH_DEFECT FALSE
```

5.6.1 Mastering methods

Overview

A robot can be mastered in the following ways:

- With the EMT (electronic measuring tool)
  (>>> 5.6.3 "Mastering with the EMT" Page 84)
- With the dial gauge
  (>>> 5.6.4 "Mastering with the dial gauge" Page 87)

The axes must be moved to the pre-mastering position before every mastering operation.

EMT mastering is recommended.

Another method is “reference mastering”. It is only used for mastering the robot after certain maintenance tasks.

(>>> 5.6.6 "Reference mastering" Page 89)

5.6.2 Moving axes to the pre-mastering position

Description

Each axis is moved so that the mastering marks line up.

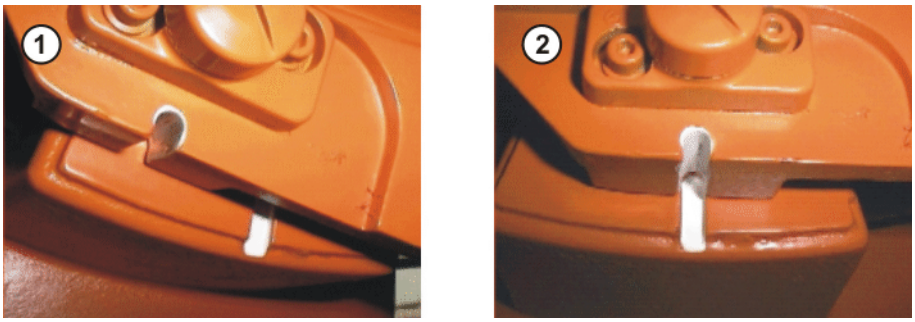


Fig. 5-3: Moving an axis to the pre-mastering position

The mastering marks are situated in the following positions on the robot:

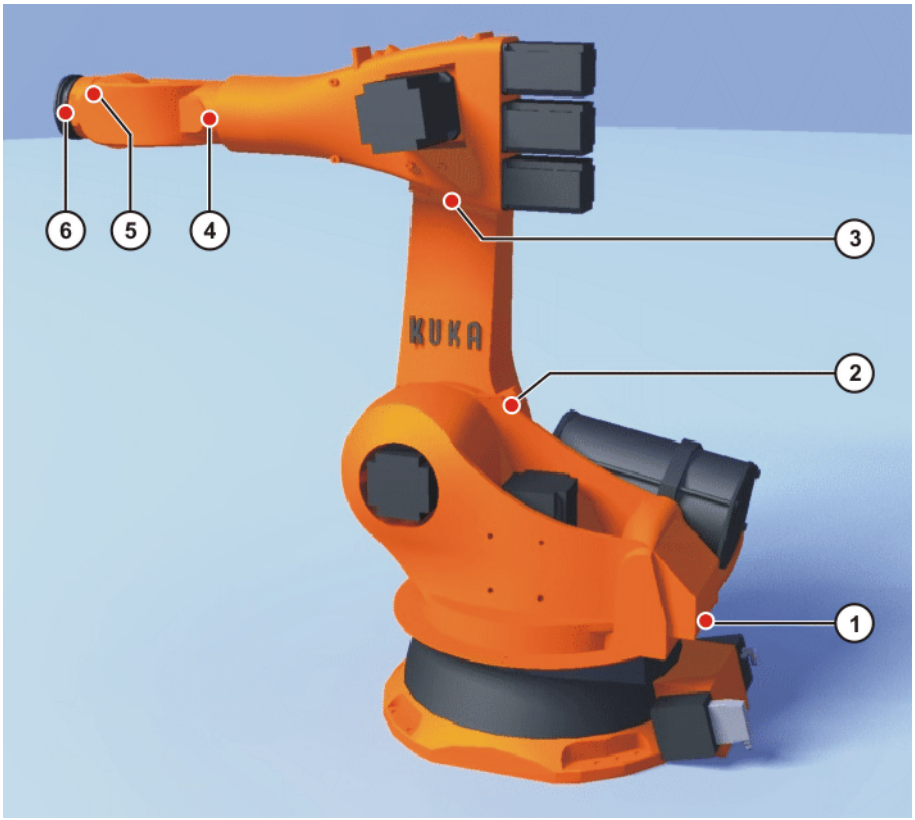


Fig. 5-4: Mastering marks on the robot

Depending on the specific robot model, the positions of the mastering marks may deviate slightly from those illustrated.

Precondition

- Operating mode T1

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. Hold down the enabling switch.
4. Axes 1 to 6 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.
5. Move each axis, starting from axis 1 and working upwards, so that the mastering marks line up.

If A4 and A6 are moved to the pre-mastering position, ensure that the energy supply system – if present – is in its correct position and not rotated through 360°.

5.6.3 Mastering with the EMT

Overview

In EMT mastering, the axis is automatically moved by the robot controller to the mastering position. Mastering is carried out first without and then with a load. It is possible to save mastering data for different loads.

Fig. 5-5: EMT – electronic measuring tool

EMT mastering consists of the following steps:

Step	Description
1	**First mastering** (>>> 5.6.3.1 "First mastering with the EMT" Page 85) First mastering is carried out without a load.
2	**Teach offset** (>>> 5.6.3.2 "Teach offset" Page 85) “Teach offset” is carried out with a load. The difference from the first mastering is saved.
3	If required: **Master load with offset** (>>> 5.6.3.3 "Master load with offset" Page 86) “Load mastering with offset” is carried out with a load for which an offset has already been taught. Area of application: ■ Checking first mastering ■ Restoring first mastering if it has been lost (e.g. following exchange of motor or collision). Since an offset that has been taught is retained, even if mastering is lost, the robot controller can calculate the first mastering.

5.6.3.1 First mastering with the EMT

Precondition

- There is no load on the robot; i.e. there is no tool, workpiece or supplementary load mounted.
- All axes are in the pre-mastering position.
- No program is selected.
- Operating mode T1

Procedure

Caution!
The EMT must always be screwed onto the gauge cartridge without the signal cable attached; only then may the signal cable be attached. When removing the EMT, always remove the signal cable from the EMT first, then remove the EMT from the gauge cartridge. Otherwise, the signal cable could be damaged.
After mastering, remove the signal cable from connection X32. Failure to do so may result in interference signals or damage.

1. Select the menu **Setup** > **Master** > **EMT** > **With load correction** > **First mastering**.
   A window opens. All axes to be mastered are displayed. The axis with the lowest number is highlighted.
2. Remove the protective cap of the gauge cartridge on the axis highlighted in the window. Screw the EMT onto gauge cartridge. Then attach signal cable to EMT and plug into connector X32 on the base frame junction box.
3. Press **Master**.
4. Press an enabling switch and the Start key.
   When the EMT detects the lowest point of the reference notch, the mastering position is reached. The robot stops automatically. The values are saved. The axis is no longer displayed in the window.
5. Remove the signal cable from the EMT. Then remove the EMT from the gauge cartridge and replace the protective cap.
6. Repeat steps 2 to 5 for all axes to be mastered.
7. Remove signal cable from connection X32.
8. Exit the window by means of **Close**.

5.6.3.2 Teach offset

Description

"Teach offset" is carried out with a load. The difference from the first mastering is saved.

If the robot is operated with different loads, "Teach offset" must be carried out for every load. In the case of grippers used for picking up heavy workpieces, "Teach offset" must be carried out for the gripper both with and without the workpiece.

Precondition

- Same ambient conditions (temperature, etc.) as for first mastering.
- The load is mounted on the robot.
- All axes are in the pre-mastering position.
- No program is selected.
- Operating mode T1

Procedure

Caution!
The EMT must always be screwed onto the gauge cartridge without the signal cable attached; only then may the signal cable be attached. When removing the EMT, always remove the signal cable from the EMT first, then remove the EMT from the gauge cartridge. Otherwise, the signal cable could be damaged.
After mastering, remove the signal cable from connection X32. Failure to do so may result in interference signals or damage.

1. Select the menu **Setup** > **Master** > **EMT** > **With load correction** > **Teach offset**.
2. Enter tool number. Confirm with **Tool OK**.
   A window opens. All axes for which the tool has not yet been taught are displayed. The axis with the lowest number is highlighted.
3. Remove the protective cap of the gauge cartridge on the axis highlighted in the window. Screw the EMT onto gauge cartridge. Then attach signal cable to EMT and plug into connector X32 on the base frame junction box.
4. Press **Teach**.
5. Press an enabling switch and the Start key.
   When the EMT detects the lowest point of the reference notch, the mastering position is reached. The robot stops automatically. A window opens. The deviation of this axis from the first mastering is indicated in degrees and increments.
6. Confirm with **OK**. The axis is no longer displayed in the window.
7. Remove the signal cable from the EMT. Then remove the EMT from the gauge cartridge and replace the protective cap.
8. Repeat steps 3 to 7 for all axes to be mastered.
9. Remove signal cable from connection X32.
10. Exit the window by means of **Close**.

5.6.3.3 Master load with offset

Description

Area of application:

- Checking first mastering
- Restoring first mastering if it has been lost (e.g. following exchange of motor or collision). Since an offset that has been taught is retained, even if mastering is lost, the robot controller can calculate the first mastering.

An axis can only be checked if all axes with lower numbers have been mastered.

Precondition

- Same ambient conditions (temperature, etc.) as for first mastering.
- A load for which “Teach offset” has been carried out is mounted on the robot.
- All axes are in the pre-mastering position.
- No program is selected.
- Operating mode T1

Procedure

Caution!
The EMT must always be screwed onto the gauge cartridge without the signal cable attached; only then may the signal cable be attached. When removing the EMT, always remove the signal cable from the EMT first, then remove the EMT from the gauge cartridge. Otherwise, the signal cable could be damaged.
After mastering, remove the signal cable from connection X32. Failure to do so may result in interference signals or damage.

1. Select the menu **Setup** > **Master** > **EMT** > **With load correction** > **Master load** > **With offset**.
2. Enter tool number. Confirm with **Tool OK**.
   A window opens. All axes for which an offset has been taught with this tool are displayed. The axis with the lowest number is highlighted.
3. Remove the protective cap of the gauge cartridge on the axis highlighted in the window. Mount EMT on gauge cartridge. Then attach signal cable to EMT and plug into connector X32 on the base frame junction box.
4. Press **Check**.
5. Hold down an enabling switch and press the Start key.
   When the EMT detects the lowest point of the reference notch, the mastering position is reached. The robot stops automatically. The difference from “Teach offset” is displayed.
6. If required, press **Save** to save the values. The old mastering values are deleted.
   To restore a lost first mastering, always save the values.

Axes A4, A5 and A6 are mechanically coupled. This means:
If the values for A4 are deleted, the values for A5 and A6 are also deleted.
If the values for A5 are deleted, the values for A6 are also deleted.

7. Remove the signal cable from the EMT. Then remove the EMT from the gauge cartridge and replace the protective cap.
8. Repeat steps 3 to 7 for all axes to be mastered.
9. Remove signal cable from connection X32.
10. Exit the window by means of **Close**.

5.6.4 Mastering with the dial gauge

Description

In dial mastering, the axis is moved manually by the user to the mastering position. Mastering is always carried out with a load. It is not possible to save mastering data for different loads.

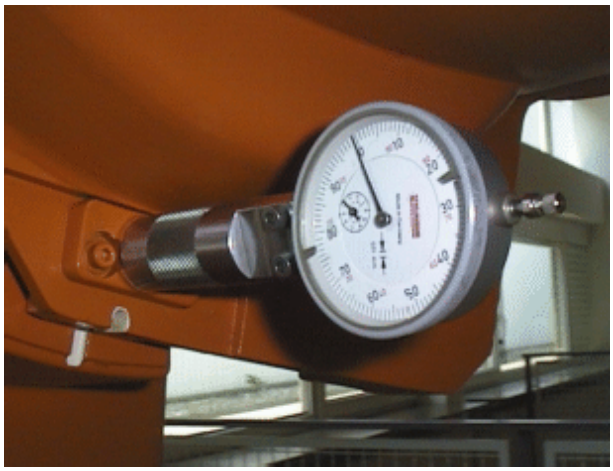


Fig. 5-6: Dial gauge

Precondition

- The load is mounted on the robot.
- All axes are in the pre-mastering position.
- Axis-specific jogging with the jog keys is selected.
  (>>> 4.14.3 "Axis-specific jogging with the jog keys" Page 60)
- No program is selected.
- Operating mode T1

Procedure

1. Select the menu sequence **Setup** > **Master** > **Dial**.
   A window opens. All axes that have not been mastered are displayed. The axis that must be mastered first is selected.
2. Remove the protective cap from the gauge cartridge on this axis and mount the dial gauge on the gauge cartridge.
   Using the Allen key, loosen the screws on the neck of the dial gauge. Turn the dial so that it can be viewed easily. Push the pin of the dial gauge in as far as the stop.
   Using the Allen key, tighten the screws on the neck of the dial gauge.
3. Reduce jog override to 1%.
4. Jog axis from “+” to “-”. At the lowest position of the reference notch, recognizable by the change in direction of the pointer, set the dial gauge to 0.
   If the axis inadvertently overshoots the lowest position, jog the axis backwards and forwards until the lowest position is reached. It is immaterial whether the axis is moved from “+” to “-” or from “-” to “+”.
5. Move the axis back to the pre-mastering position.
6. Move the axis from “+” to “-” until the pointer is about 5-10 scale divisions before zero.
7. Switch to incremental jogging.
8. Move the axis from “+” to “-” until zero is reached.

If the axis overshoots zero, repeat steps 5 to 8.

9. Press **Master**. The axis that has been mastered is removed from the window.
10. Remove the dial gauge from the gauge cartridge and replace the protective cap.
11. Switch back from incremental jogging to the normal jog mode.
12. Repeat steps 2 to 11 for all axes to be mastered.
13. Exit the window by means of **Close**.

5.6.5 Mastering external axes

Description

- KUKA external axes can be mastered using either the EMT or the dial gauge.
- Non-KUKA external axes can be mastered using the dial gauge. If mastering with the EMT is desired, the external axis must be fitted with gauge cartridges.

Procedure

- The procedure for mastering external axes is the same as that for mastering robot axes. Alongside the robot axes, the configured external axes now also appear in the axis selection window.

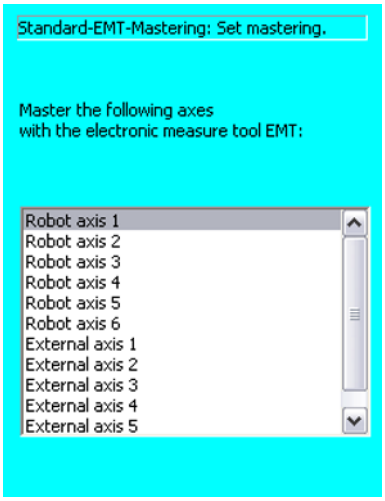


Fig. 5-7: Selection list of axes to be mastered

Mastering in the case of industrial robots with more than 2 external axes: if the system contains more than 8 axes, it may be necessary to connect the signal cable of the EMT to the second RDC.

5.6.6 Reference mastering

Reference mastering must not be used when commissioning the robot.

Description

Reference mastering is suitable if maintenance work is due on a correctly mastered robot and it is to be expected that the robot will lose its mastering. Examples:

- Exchange of RDC
- Exchange of motor

The robot is moved to the $MAMES position before the maintenance work is commenced. Afterwards, the axis values of this system variable are reassigned to the robot by means of reference mastering. The state of the robot is then the same as before the loss of mastering. Taught offsets are retained. No EMT or dial gauge is required.

In the case of reference mastering, it is irrelevant whether or not there is a load mounted on the robot. Reference mastering can also be used for external axes.

Preparation

- Move the robot to the $MAMES position before commencing the maintenance work. To do so, program a point PTP $MAMES and move the robot to it. This is only possible in the user group “Expert”!

Warning!
The robot must not move to the default HOME position instead of to $MAMES. $MAMES may be, but is not always, identical to the default HOME position. Only in the $MAMES position will the robot be correctly mastered by means of reference mastering. If the robot is reference mastered at any position other than $MAMES, this may result in physical injury and material damage.

Precondition

- No program is selected.
- The position of the robot was not changed during the maintenance work.
- If the RDC has been exchanged: the robot data have been transferred from the hard drive to the RDC (this can only be done in the user group “Expert”!)
  (>>> 5.4 "Transferring data from the hard drive to the RDC (after exchange of RDC)" Page 80)

Procedure

1. Select the menu sequence **Setup** > **Master** > **Reference**.
   The option window **Reference-Mastering** is opened. All axes that have not been mastered are displayed. The axis that must be mastered first is selected.
2. Press the **Master** softkey. The selected axis is mastered and removed from the option window.
3. Repeat step 2 for all axes to be mastered.
4. Exit the option window **Reference-Mastering** by pressing the **Close** softkey.

5.6.7 Saving the mastering

Procedure

- Select the menu sequence **Setup** > **Master** > **Save current data**.

Description

Save current data saves all mastering data to the RDC.

In earlier KSS versions, this manual backup prevented loss of the mastering data if the robot controller could not be shut down properly, e.g. due to a defective battery.

The mastering is now backed up automatically. Manual backup is no longer necessary. The command is still available in the user interface, however.

5.6.8 Manually unmastering axes

Description

The mastering values of the individual axes can be deleted. The axes do not move during unmastering.

Axes A4, A5 and A6 are mechanically coupled. This means:
If the values for A4 are deleted, the values for A5 and A6 are also deleted.
If the values for A5 are deleted, the values for A6 are also deleted.

Warning!
The software limit switches of an unmastered robot are deactivated. The robot can hit the end stop buffers, thus damaging the robot and making it necessary to exchange the buffers. An unmastered robot must not be jogged, if at all avoidable. If it must be jogged, the jog override must be reduced as far as possible.

Precondition

- No program is selected.

Procedure

1. Select the menu sequence **Setup** > **Unmaster**. A window opens.
2. Select the axis to be unmastered.
3. Press **UnMaster**. The mastering data of the axis are deleted.
4. Exit the window by means of **Close**.

5.7 Calibration

5.7.1 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.

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.
(>>> 5.7.2 "Fixed tool calibration" Page 98)

Advantages of the 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

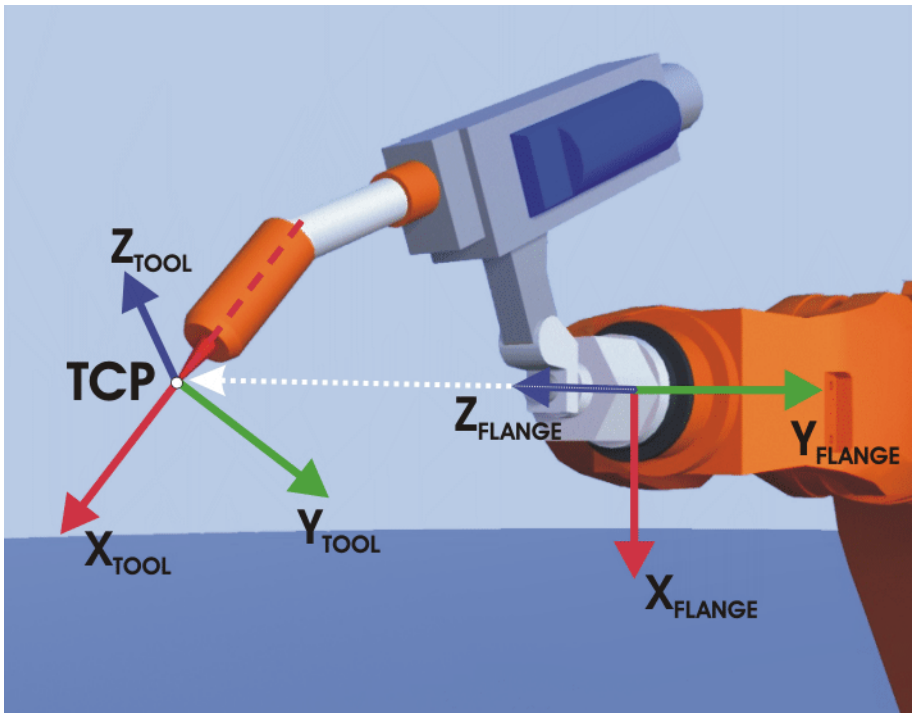


Fig. 5-8: TCP calibration principle

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 ■ XYZ Reference
2	**Definition of the orientation of the TOOL coordinate system** The following methods are available: ■ ABC World ■ ABC 2-Point

If the calibration data are already known, they can be entered directly.

5.7.1.1 TCP calibration: XYZ 4-Point method

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.

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 **Measure**. Confirm with **Continue**.

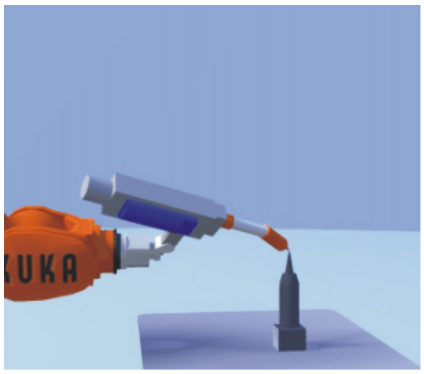


Fig. 5-9: Move to reference point

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

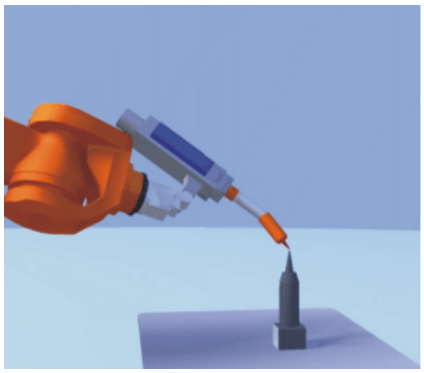


Fig. 5-10: Move to point from different direction

5. Repeat step 4 twice.

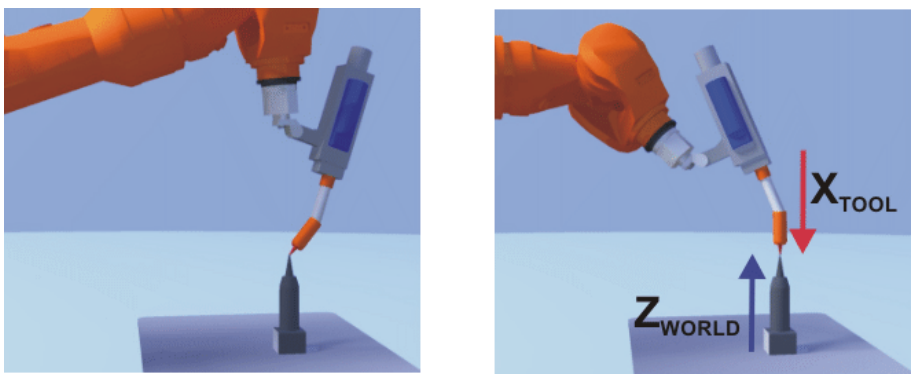


Fig. 5-11: Move to point from 3rd and 4th direction

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.

(>>> 5.8.3 "Entering payload data" Page 112)

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.

(>>> 5.7.1.3 "Defining the orientation: ABC 2-Point method" Page 95)

(>>> 5.7.1.4 "Defining the orientation: ABC World method" Page 96)

5.7.1.2 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**.

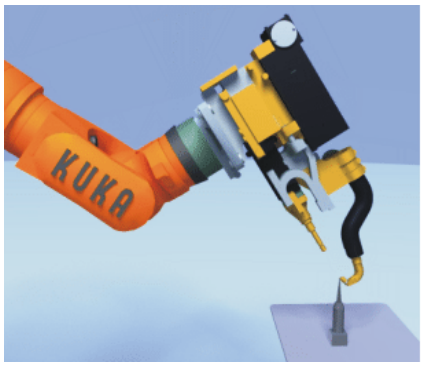


Fig. 5-12: Move to point with calibrated tool

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. 5-13: 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.
   (>>> 5.8.3 "Entering payload data" Page 112)
   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.
   (>>> 5.7.1.3 "Defining the orientation: ABC 2-Point method" Page 95)
   (>>> 5.7.1.4 "Defining the orientation: ABC World method" Page 96)

5.7.1.3 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

Procedure

1. Select the menu **Setup** > **Measure** > **Tool** > **ABC 2-Point**.
2. Enter the number of the mounted tool. Confirm with **Continue**.
3. Move the TCP to any reference point. Press **Measure**. Confirm with **Continue**.

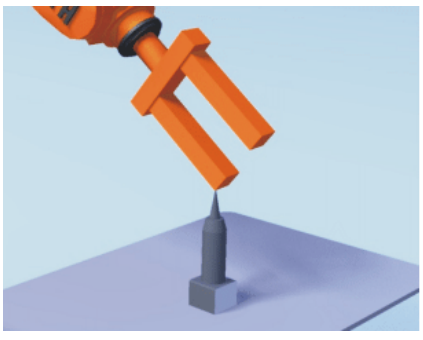


Fig. 5-14: Move to reference point

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 **Measure**. Confirm with **Continue**.

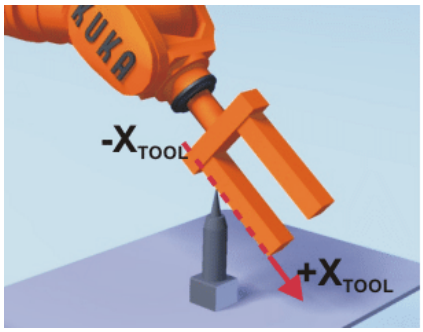


Fig. 5-15: Reference point on the X-axis

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

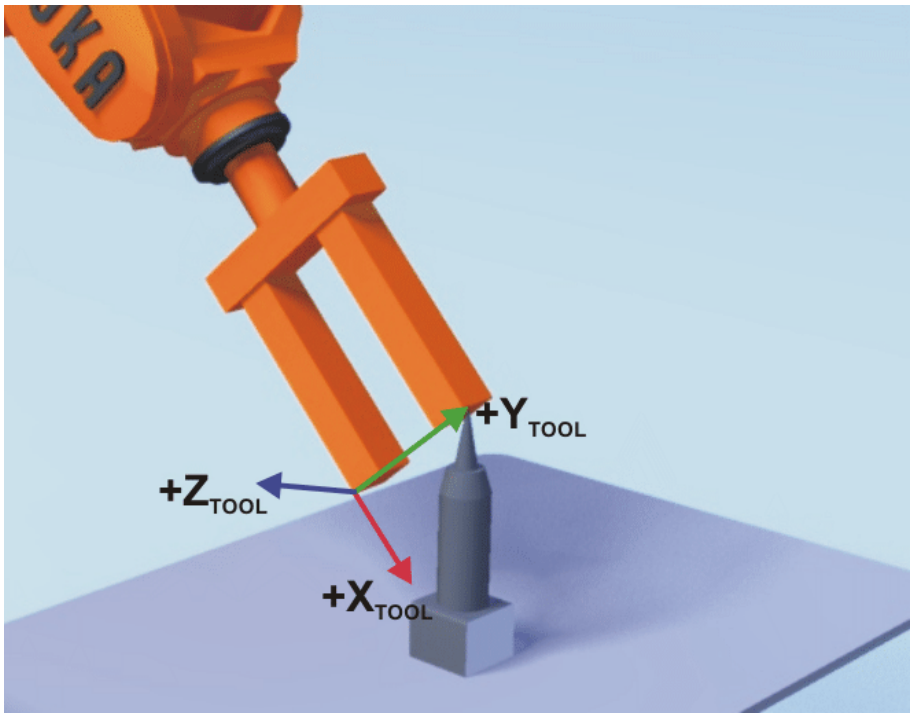


Fig. 5-16: Reference point on XY plane

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.
   (>>> 5.8.3 "Entering payload data" Page 112)

5.7.1.4 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 used: 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

Procedure

1. Select the menu **Setup** > **Measure** > **Tool** > **ABC World**.
2. Enter the number of the tool. Confirm with **Continue**.
3. Select a variant in the box **5D/6D**. Confirm with **Continue**.
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 **Measure**. Confirm with **Continue**.
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.
   (>>> 5.8.3 "Entering payload data" Page 112)

5.7.1.5 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 **Continue**.
3. Enter data. Confirm with **Continue**.
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.
   (>>> 5.8.3 "Entering payload data" Page 112)

5.7.2 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. (>>> 5.7.2.1 "Calibrating an external TCP" Page 98) If the calibration data are already known, they can be entered directly. (>>> 5.7.2.2 "Entering the external TCP numerically" Page 100)
2	**Calibration of the workpiece** The following methods are available: ■ Direct method (>>> 5.7.2.3 "Workpiece calibration: direct method" Page 100) ■ Indirect method (>>> 5.7.2.4 "Workpiece calibration: indirect method" Page 101)

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.

5.7.2.1 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 used: 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

Procedure

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

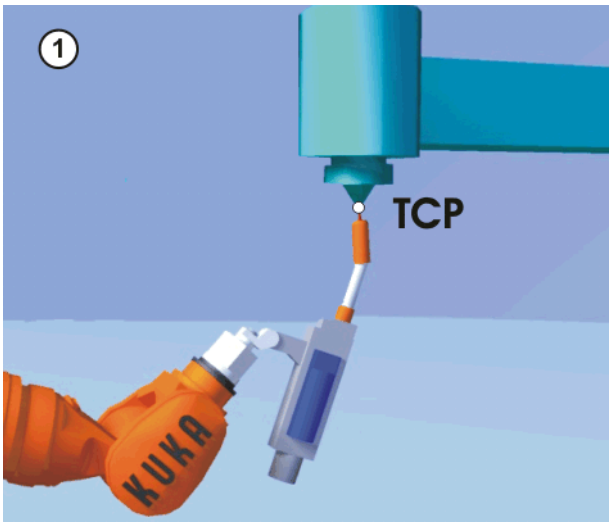


Fig. 5-17: Moving to the external TCP

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 $+X_{FLANGE}$

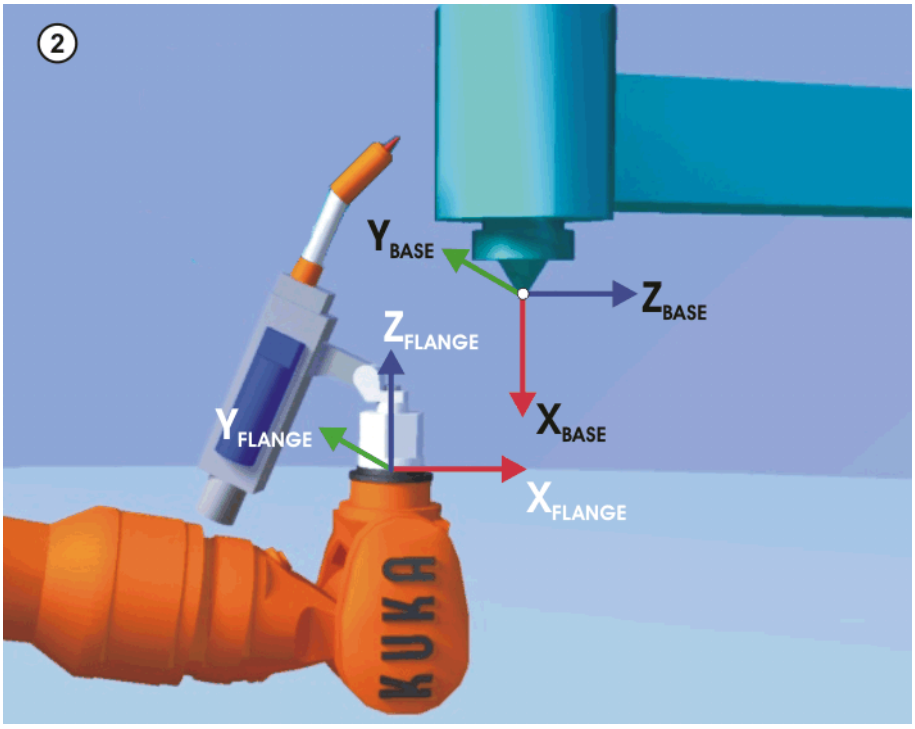


Fig. 5-18: Aligning the coordinate systems parallel to one another

7. Press **Measure**. Confirm with **Continue**.
8. Press **Save**.

5.7.2.2 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**.

5.7.2.3 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**.

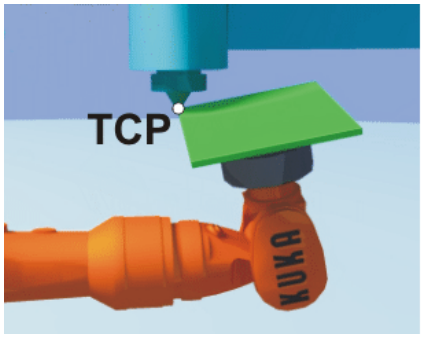


Fig. 5-19: Origin of the workpiece coordinate system

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**.

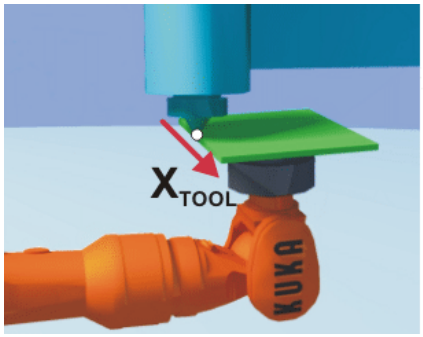


Fig. 5-20: Point on the positive X axis

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**.

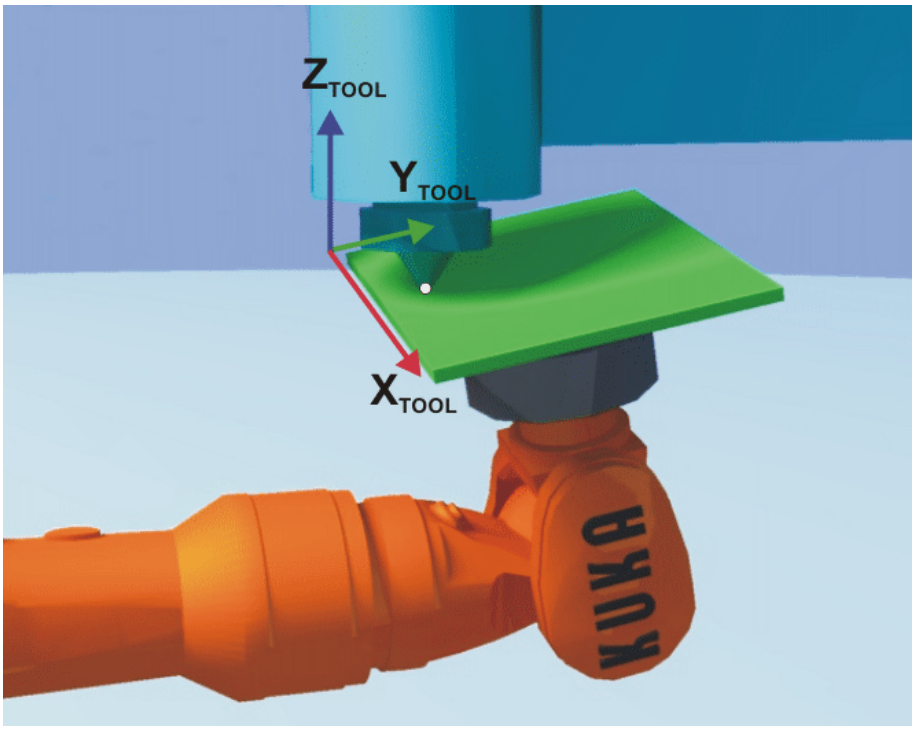


Fig. 5-21: Point on XY plane with a positive Y value

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.
   (>>> 5.8.3 "Entering payload data" Page 112)

5.7.2.4 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**.

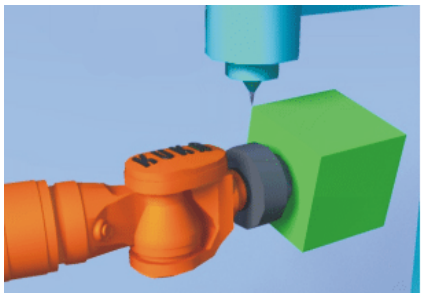
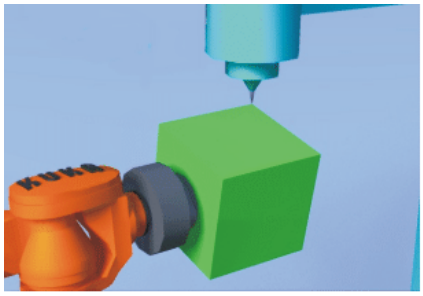
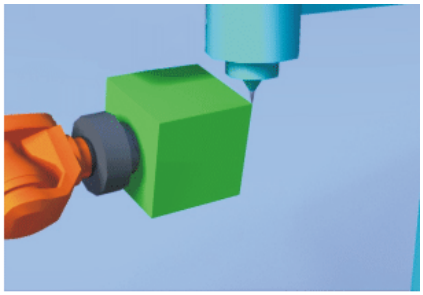


Fig. 5-22: Moving to a known point on the TCP

5. Repeat step 4 three times.





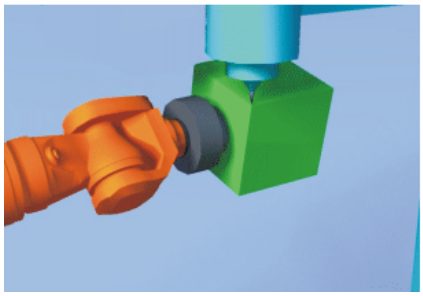


Fig. 5-23: Repeat three times

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.
   (>>> 5.8.3 "Entering payload data" Page 112)

5.7.3 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.

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. (>>> 5.7.2 "Fixed tool calibration" Page 98)

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
- Indirect method

If the calibration data are already known, they can be entered directly. (>>> 5.7.3.3 "Entering the base numerically" Page 105)

5.7.3.1 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**.

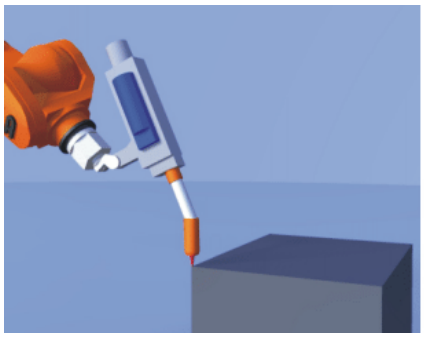


Fig. 5-24: Origin of the base coordinate system

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

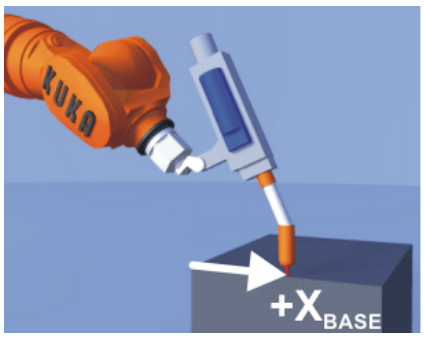


Fig. 5-25: Point on the positive X axis

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

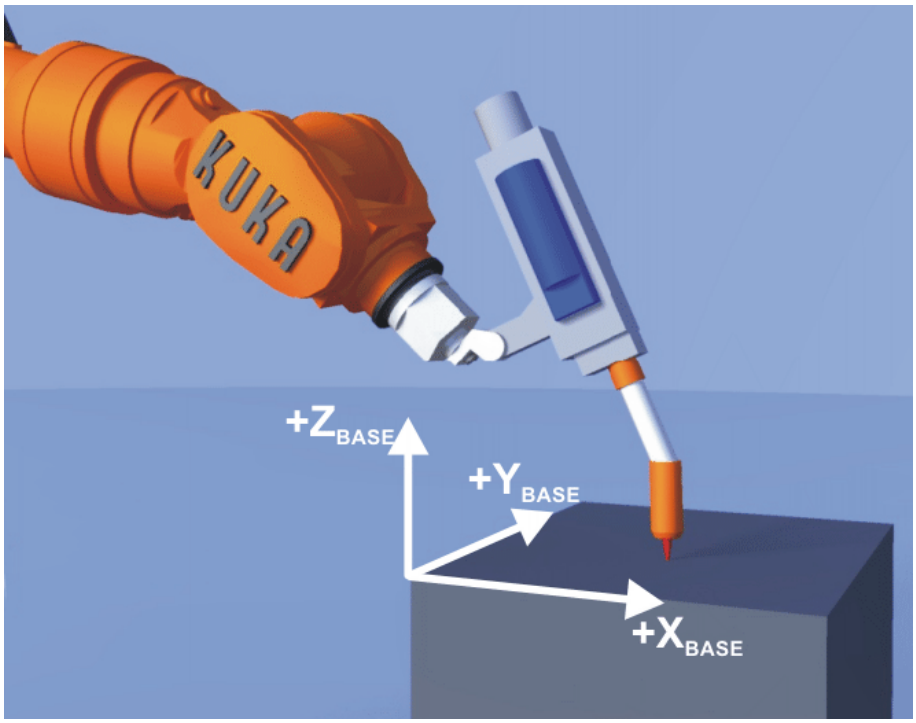


Fig. 5-26: Point on XY plane with a positive Y value

7. Press **Save**.

5.7.3.2 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**.

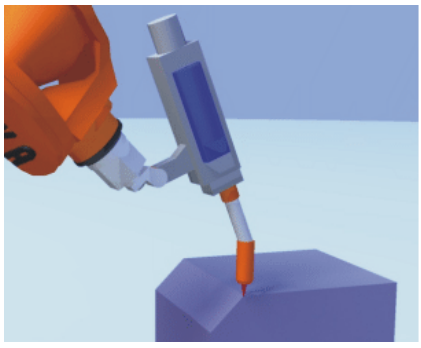


Fig. 5-27: Moving to a known point

5. Repeat step 4 three times.

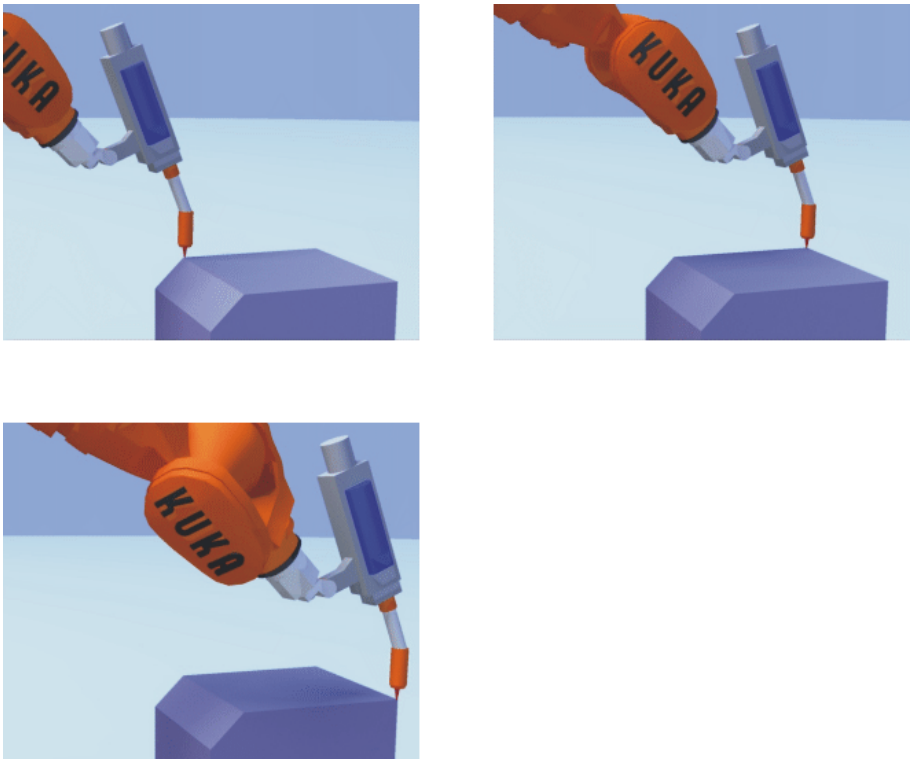


Fig. 5-28: Repeat three times

6. Press **Save**.

5.7.3.3 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**.

5.7.4 Calibrating an external kinematic system

Description

Calibration of the external kinematic system is necessary to enable the motion of the axes of the kinematic system to be synchronized and mathematically coupled with the robot axes. An external kinematic system can be a turn-tilt table or positioner, for example.

For linear units, the type of calibration described here cannot be used.

Basic information about external kinematic systems and the coordinate systems used can be found in the documentation **External axes**.

Overview

Calibration of an external kinematic system consists of 2 steps:

Step	Description
1	Calibrate the root point of the external kinematic system. (>>> 5.7.4.1 "Calibrating the root point" Page 106) If the calibration data are already known, they can be entered directly. (>>> 5.7.4.2 "Entering the root point numerically" Page 107)
2	If there is a workpiece on the external kinematic system: Calibrate the base. (>>> 5.7.4.3 "Base calibration (external kinematic system)" Page 108) If the calibration data are already known, they can be entered directly. (>>> 5.7.4.4 "Entering the offset base numerically" Page 110)
	If there is a tool mounted on the external kinematic system: calibrate the external tool. (>>> 5.7.4.5 "Calibrating an external tool" Page 110) If the calibration data are already known, they can be entered directly. (>>> 5.7.4.6 "Entering the external tool numerically" Page 111)

5.7.4.1 Calibrating the root point

Description

In order to be able to move the robot with a mathematical coupling to a kinematic system, the robot must know the precise location of the kinematic system. This location is determined by means of root point calibration.

The TCP of a tool that has already been calibrated is moved to a reference point on the kinematic system 4 times. The position of the reference point must be different each time. This is achieved by moving the axes of the kinematic system. The robot controller uses the different positions of the reference point to calculate the root point of the kinematic system.

Before calibration, the reference point must be defined in the system variable $ET*x*_TPINFL in the machine data. $ET*x*_TPINFL contains the position of the reference point relative to the FLANGE coordinate system of the kinematic system. (*x* = number of the kinematic system.)

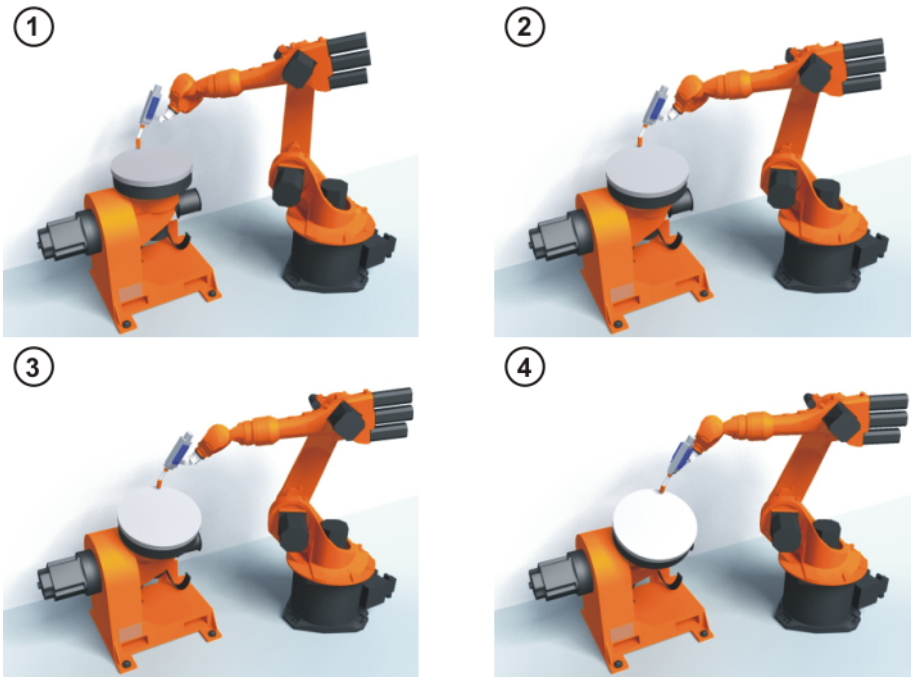


Fig. 5-29: Root point calibration principle

Precondition

- The machine data of the kinematic system have been configured and loaded into the robot controller.
  In particular: $ET*x*_TPINFL has been configured.
- The number of the external kinematic system is known.
- A previously calibrated tool is mounted on the mounting flange.
- Operating mode T1 or T2

Detailed information about creating machine data for external kinematic systems can be found in the documentation **External axes**.

Procedure

1. Select the menu **Setup** > **Measure** > **External kinematic** > **Root point**.
2. Enter the number of the external kinematic system.
3. Assign a name for the external kinematic system. Confirm with **Next**.
4. Enter the number of the reference tool. Confirm with **Next**.
5. The value of $ET*x*_TPINFL displayed for control purposes. Confirm with **Next**.
   (If the value is not correct, cancel the calibration and correct $ET*x*_TPINFL in the machine data.)
6. Move the TCP to the reference point.
7. Press **Measure**. Confirm with **Next**.
8. Repeat steps 6 and 7 three times. Each time, move the kinematic system first so that the reference point is approached from different positions.
9. Press **Save**.

5.7.4.2 Entering the root point numerically

Precondition

- The following numerical values are known, e.g. from CAD data:
  - Distance between the origin of the ROOT coordinate system and the origin of the WORLD coordinate system (X, Y, Z)
  - Orientation of the ROOT coordinate system relative to the WORLD coordinate system (A, B, C)

- The number of the external kinematic system is known.

Procedure

1. Select the menu **Setup** > **Measure** > **External kinematic** > **Root point (numeric)**.
2. Enter the number of the external kinematic system.
3. Assign a name for the external kinematic system. Confirm with **Next**.
4. Enter the data of the ROOT coordinate system. Confirm with **Next**.
5. Press **Save**.

5.7.4.3 Base calibration (external kinematic system)

Description

During this calibration, the user assigns a BASE coordinate system to a workpiece located on the kinematic system. This BASE coordinate system is relative to the FLANGE coordinate system of the kinematic system. The base is thus a moving base that moves in the same way as the kinematic system.

It is not strictly necessary to calibrate a base. If none is calibrated, the FLANGE coordinate system of the kinematic system is taken as the base.

During calibration, the TCP of a calibrated tool is moved to the origin and 2 other points of the desired base. These 3 points define the base. The coordinates are saved as BASE_DATA[17 .... 22]. Only one base can be calibrated per kinematic system.

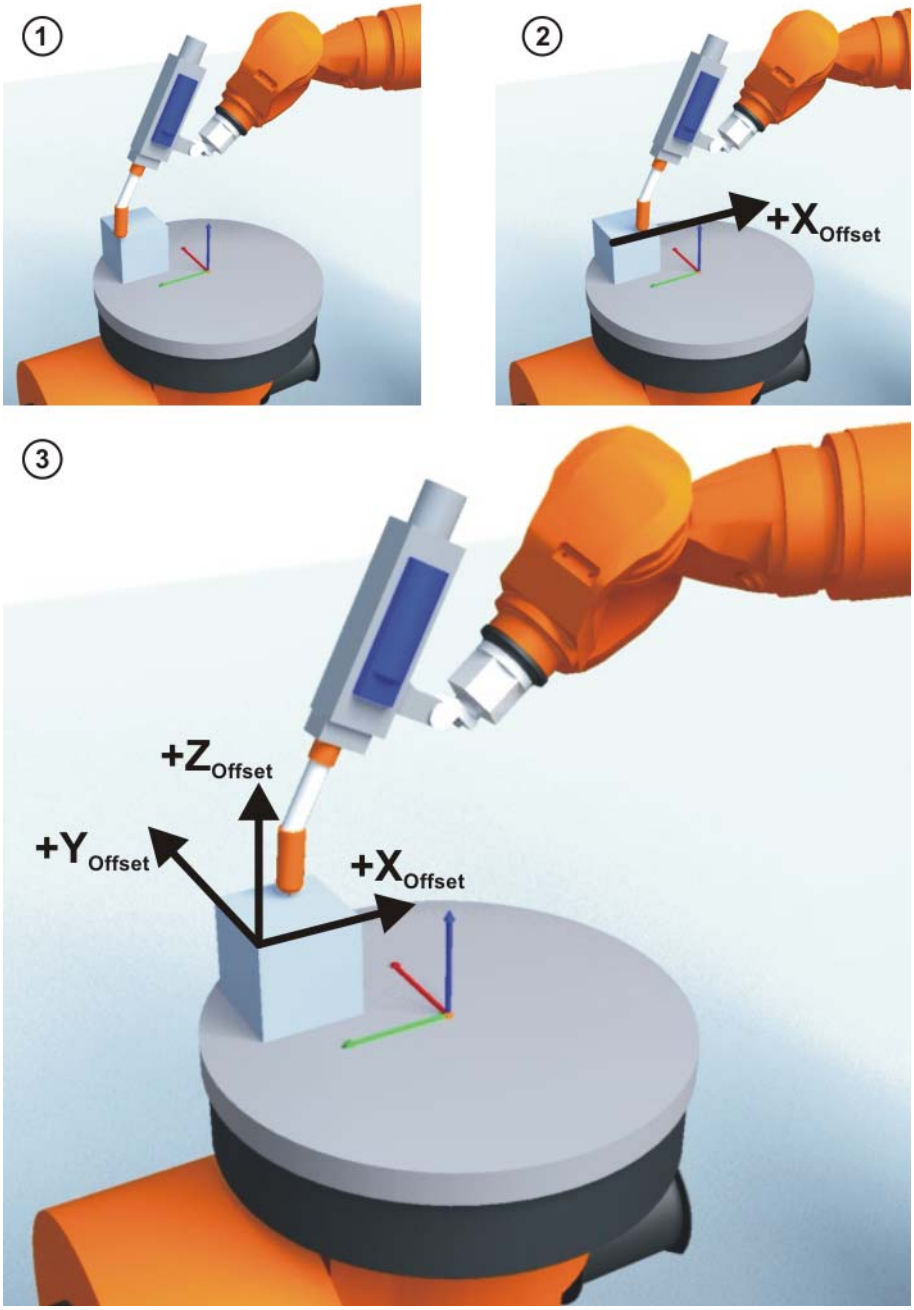


Fig. 5-30: Base calibration principle

Precondition

- The machine data of the kinematic system have been configured and loaded into the robot controller.
- The number of the external kinematic system is known.
- A previously calibrated tool is mounted on the mounting flange.
- Operating mode T1 or T2

Detailed information about creating machine data for external kinematic systems can be found in the documentation **External axes**.

Procedure

1. Select the menu **Setup** > **Measure** > **External kinematic** > **Offset**.
2. Enter the number of the external kinematic system.
   The name of the external kinematic system is displayed. Confirm with **Next**.
3. Enter the number of the reference tool. Confirm with **Next**.

4. Move the TCP to the origin of the offset base. Press **Measure** and confirm with **Next**.
5. Move the TCP to a point on the positive X axis of the offset base. Press **Measure** and confirm with **Next**.
6. Move the TCP to a point in the XY plane with a positive Y value. Press **Measure** and confirm with **Next**.
7. Press **Save**.

5.7.4.4 Entering the offset base numerically

Precondition

- The following numerical values are known, e.g. from CAD data:
  - Distance between the origin of the offset base and the origin of the FLANGE coordinate system of the kinematic system (X, Y, Z)
  - Rotation of the axes of the offset base relative to the FLANGE coordinate system of the kinematic system (A, B, C)
- The number of the external kinematic system is known.

Procedure

1. Select the menu **Setup** > **Measure** > **External kinematic** > **Offset (numeric)**.
2. Enter the number of the external kinematic system.

   The name of the external kinematic system is displayed. Confirm with **Next**.
3. Enter data. Confirm with **Next**.
4. Press **Save**.

5.7.4.5 Calibrating an external tool

Description

During calibration of the external tool, the user assigns a coordinate system to the tool mounted on the kinematic system. This coordinate system has its origin in the TCP of the external tool and is relative to the FLANGE coordinate system of the kinematic system.

First of all, the user communicates to the robot controller the TCP of the tool mounted on the kinematic system. This is done by moving a calibrated tool to the TCP.

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

- **5D**: The user communicates the tool direction 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 influenced by the user.

  The system always defines the orientation of the other axes in the same way. If the tool subsequently has to be calibrated again, e.g. after a crash, it is therefore sufficient to define the tool direction again. Rotation about the tool direction need not be taken into consideration.
- **6D**: The user communicates the direction of all 3 axes to the robot controller.

If **6D** is used: 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.

The coordinates of the external tool are saved as BASE_DATA[17...22].

Precondition

- The machine data of the kinematic system have been configured and loaded into the robot controller.
- The number of the external kinematic system is known.
- A previously calibrated tool is mounted on the mounting flange.
- Operating mode T1 or T2

Procedure

1. Select the menu **Setup** > **Measure** > **Fixed tool** > **Offset external kinematic**.
2. Enter the number of the external kinematic system.
   The name of the external kinematic system is displayed. Confirm with **Continue**.
3. Enter the number of the reference tool. Confirm with **Continue**.
4. Select a variant in the box **5D/6D**. Confirm with **Continue**.
5. Move the TCP of the calibrated tool to the TCP of the external tool. Press **Measure** and confirm with **Next**.
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 external tool.)
   If **6D** is selected:
   Align the mounting flange so that its axes are parallel to the axes of the external tool:
   - $+X_{BASE}$ parallel to $-Z_{FLANGE}$
     (i.e. align the mounting flange perpendicular to the tool direction of the external tool.)
   - $+Y_{BASE}$ parallel to $+Y_{FLANGE}$
   - $+Z_{BASE}$ parallel to $+X_{FLANGE}$
7. Press **Measure** and confirm with **Next**.
8. Press **Save**.

5.7.4.6 Entering the external tool numerically

Precondition

- The following numerical values are known, e.g. from CAD data:
  - Distance between the TCP of the external tool and the origin of the FLANGE coordinate system of the kinematic system (X, Y, Z)
  - Rotation of the axes of the external tool relative to the FLANGE coordinate system of the kinematic system (A, B, C)

Procedure

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

5.8 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!
If a robot is operated with incorrect load data or an unsuitable load, this can result in danger to life and limb and/or substantial material damage.

Sources

Load data can be obtained from the following sources:

- Software option KUKA.LoadDetect (only for payloads)
- Manufacturer information
- Manual calculation
- CAD programs

5.8.1 Checking loads with KUKA.Load

All load data (payload and supplementary loads) must be checked with the KUKA.Load software. Exception: If the payload is checked with KUKA.Load Detect, it is not necessary to check it with KUKA.Load.

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.

5.8.2 Determining payloads with KUKA.Load Detect

KUKA.Load Detect can be used to calculate payloads exactly and transfer them to the robot controller. KUKA.Load Detect can only be used for payloads over 20% of the rated payload.

Functional principle: the payload is mounted on the robot. The mass, center of gravity and the mass inertia at the center of gravity are determined exactly by means of pendulum motions.

A payload can also be checked with KUKA.Load Detect, in a similar way to with KUKA.Load. If a sign-off sheet is to be created for the load, however, the load must be checked with KUKA.Load.

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

5.8.3 Entering payload data

Description

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

Exception: If the payload data have already been transferred to the robot controller by KUKA.Load Detect, no manual entry is required.

Precondition

- The payload data have been checked with KUKA.Load or KUKA.Load Detect 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 **Next**.
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**: Orientation of the principal inertia axes relative to the flange
   - 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.)

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

5.8.4 Entering supplementary load data

Description

The supplementary load data must be entered in the robot controller.

Reference systems of the X, Y and Z values for each supplementary load:

Load	Reference system
Supplementary load A1	ROBROOT coordinate system A1 = 0°
Supplementary load A2	ROBROOT coordinate system A2 = -90°
Supplementary load A3	FLANGE coordinate system A4 = 0°, A5 = 0°, A6 = 0°

Precondition

- The supplementary loads have been verified with KUKA.Load and are suitable for this robot type.

Procedure

1. Select the menu sequence **Setup** > **Measure** > **Supplementary load data**.
2. Enter the number of the axis on which the supplementary load is to be mounted. Confirm with **Next**.
3. Enter the load data. Confirm with **Next**.
4. Press **Save**.

5.9 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.

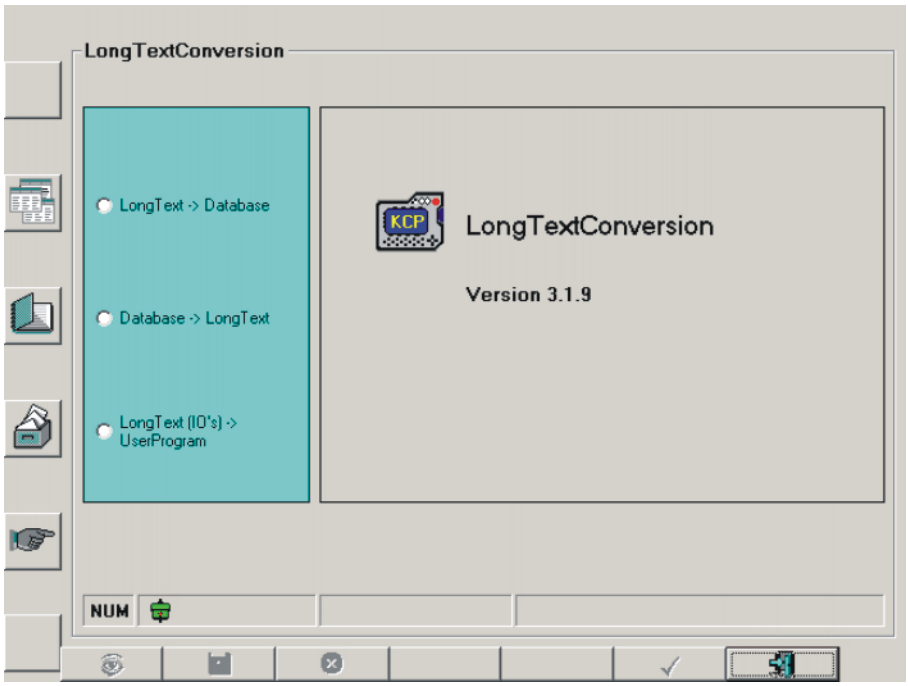


Fig. 5-31: Long text conversion window

5.9.1 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.

5.9.2 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.

5.9.3 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.

5.9.4 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

Keyword	Number	Meaning
CounterText	1 … 20	Counter
ANIN_	1 … 32	Analog input
ANOUT_	1 … 32	Analog output

6 Configuration

6.1 Reconfiguring the I/O driver

Precondition

- User group "Expert".
- Operating mode T1 or T2.

Procedure

Warning!
All outputs are reset!

1. Select the menu sequence **Configure** > **I/O Driver** > **Reconfigure I/O Driver**.
2. Acknowledge messages.

6.2 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.

6.3 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**.

6.4 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** > **Configure**.
   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

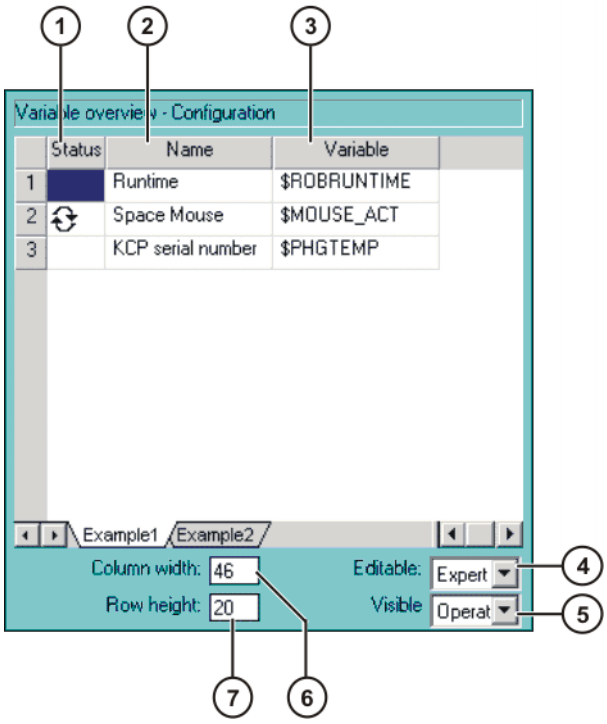


Fig. 6-1: Variable overview - Configuration window

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. (>>> 4.16.7 "Displaying the variable overview and modifying variables" Page 72)
Tab +	Switches to the next group.

Softkey	Description
Jump	Sets the focus on the next element in the user interface.
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.

6.5 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.

6.6 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.

6.7 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**.

(>>> 4.16.2 "Displaying digital inputs/outputs" Page 66)

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.
$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.

6.8 Configuring workspaces

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.

A maximum of 8 Cartesian (=cubic) and 8 axis-specific workspaces can be configured at any one time. The workspaces can overlap.

(>>> 6.8.1 "Configuring Cartesian workspaces" Page 122)

(>>> 6.8.2 "Configuring axis-specific workspaces" Page 125)

Exactly what reactions occur when the robot violates a workspace depends on the configuration.

6.8.1 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

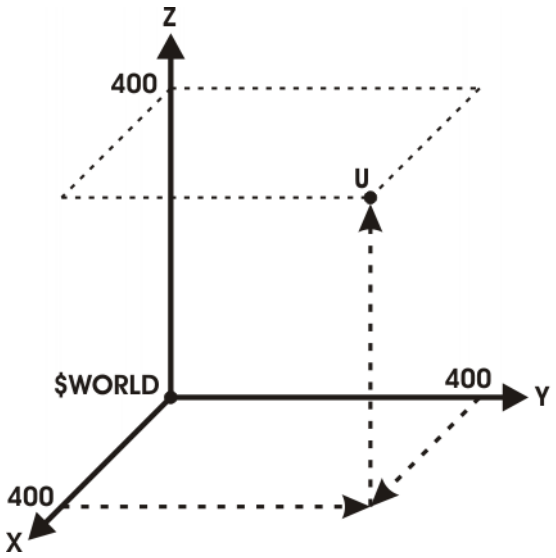


Fig. 6-2: Cartesian workspace, origin U

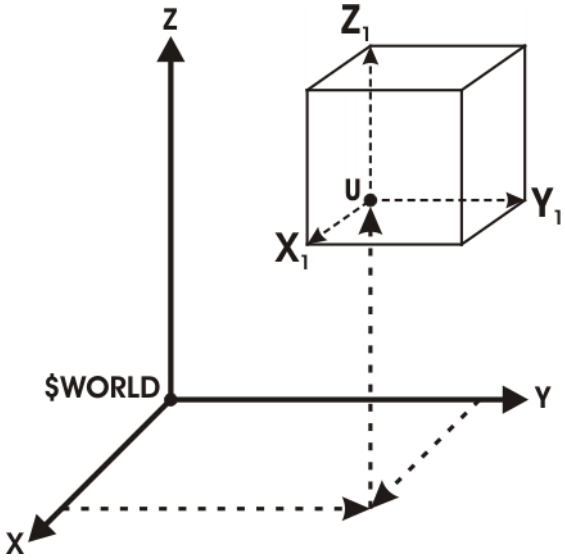


Fig. 6-3: Cartesian workspace, dimensions

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.

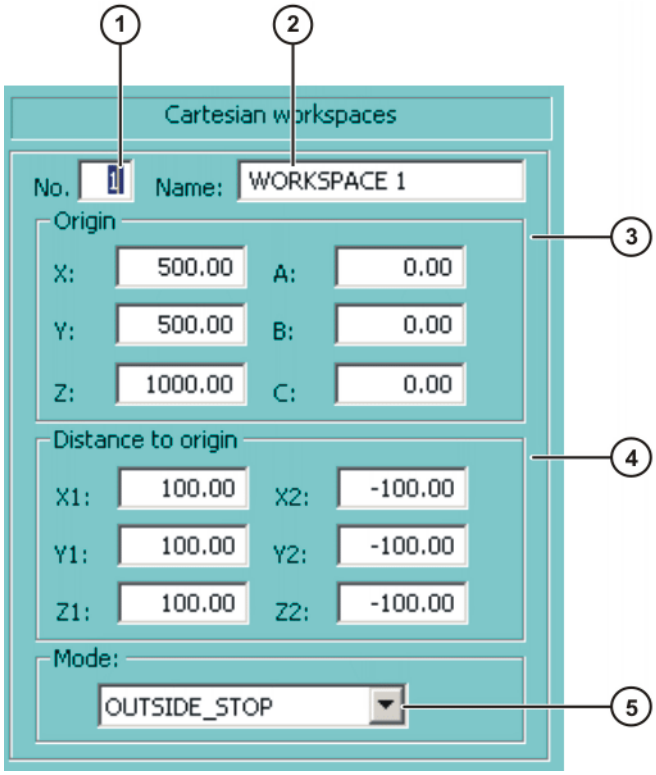


Fig. 6-4: Cartesian workspaces window

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 (>>> 6.8.3 "Mode for workspaces" Page 127)

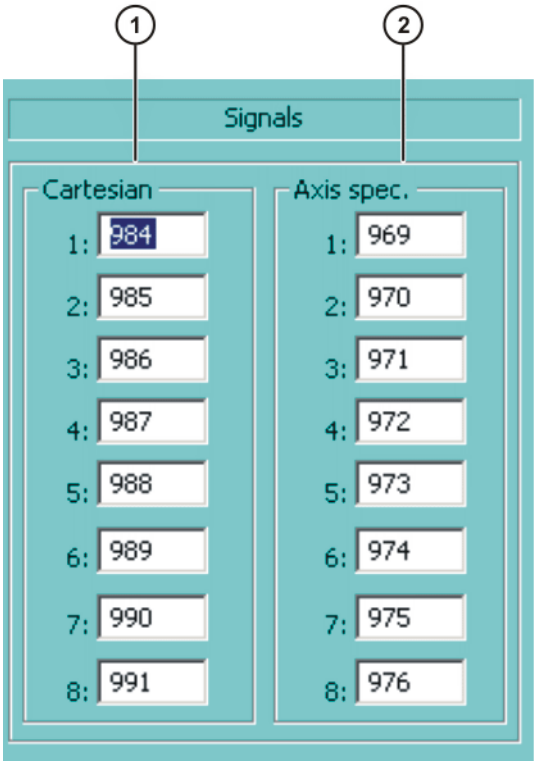


Fig. 6-5: Signals window

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.

6.8.2 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.

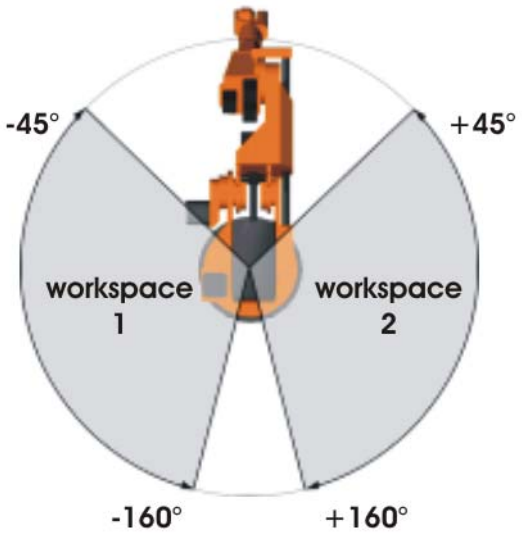


Fig. 6-6: Example of axis-specific workspaces for A1

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.

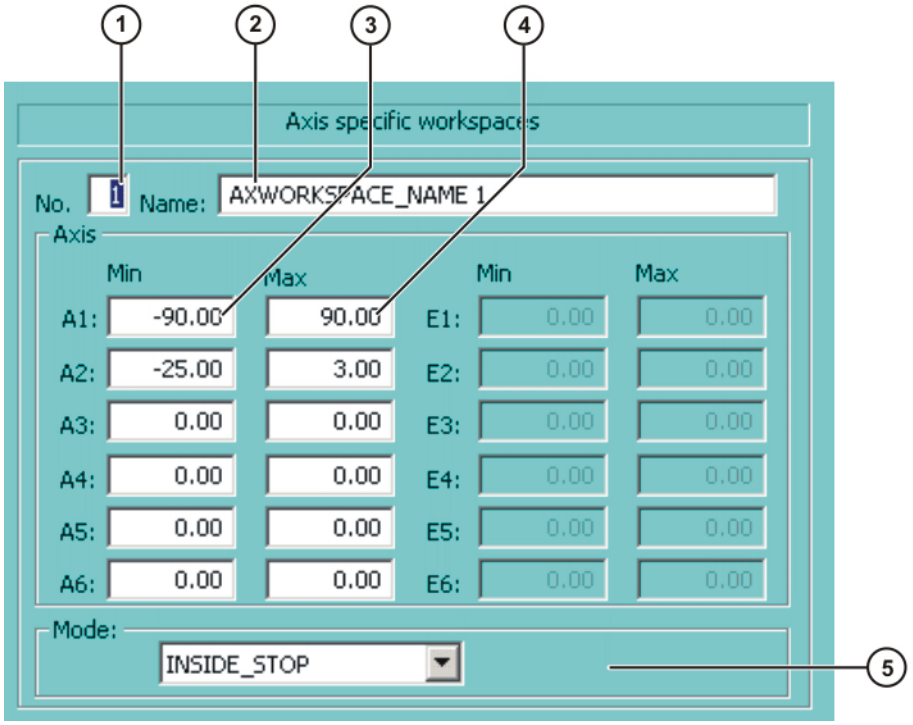


Fig. 6-7: Axis-specific workspaces window

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 (>>> 6.8.3 "Mode for workspaces" Page 127)

If the value 0 is entered for an axis under Item 3 and Item 4, the axis is not monitored, irrespective of the mode.

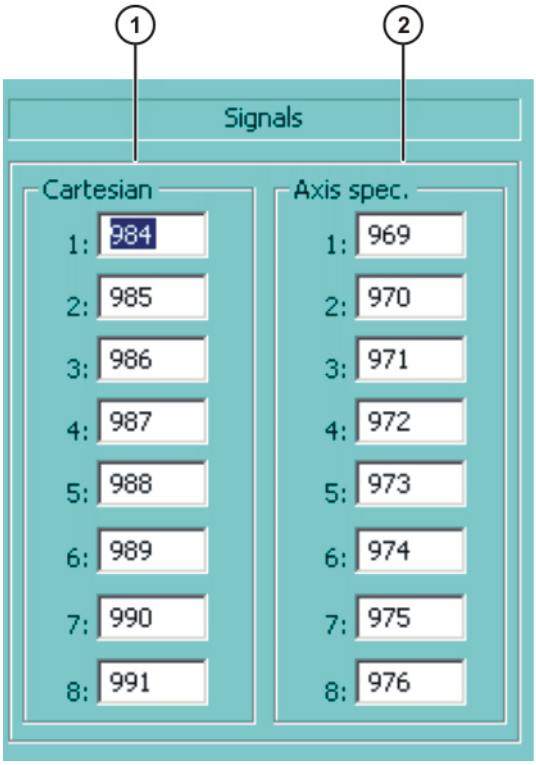


Fig. 6-8: Signals window

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.

6.8.3 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. (>>> 4.15 "Bypassing workspace monitoring" Page 65)
#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. (>>> 4.15 "Bypassing workspace monitoring" Page 65)

6.9 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**.

6.10 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

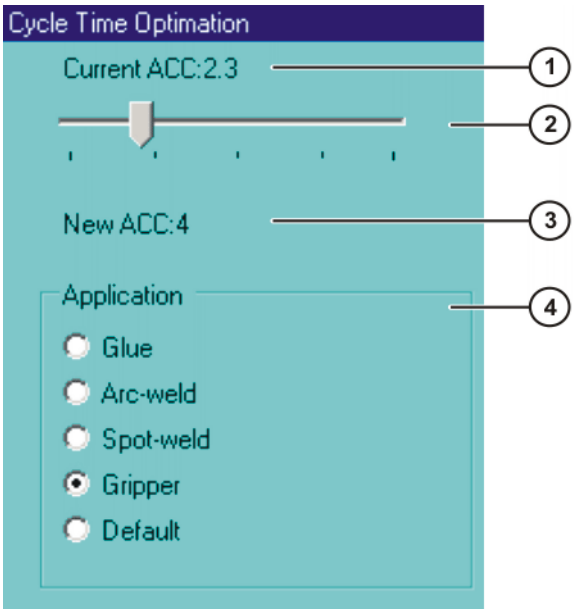


Fig. 6-9: Cycle Time Optimation window

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.

6.11 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

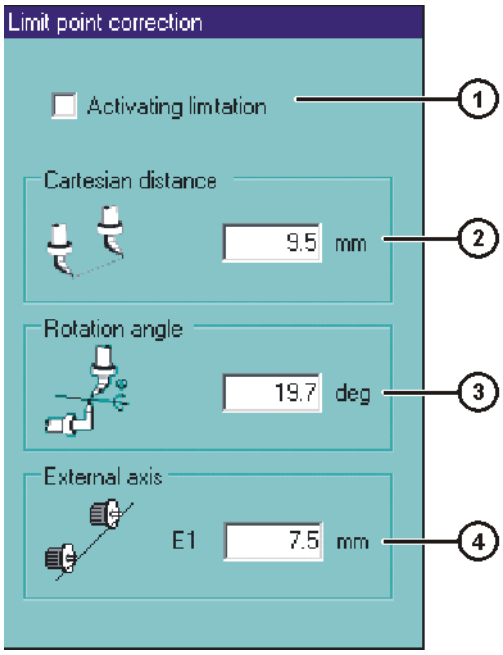


Fig. 6-10: Option window “Limit point correction”

Item	Description
1	Check box active: 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.
4	This box is only displayed if external axes are present. If several external axes are present, the next external axis can be displayed by pressing the **Ext Axes** softkey. Maximum permissible correction for the displayed external axis.

The following softkeys are available:

Softkey	Description
Correction	Activates or deactivates the check box **Activating limitation**.
Ext Axes	This softkey is only available if more than one external axis is present. Displays the next external axis.

6.12 Warm-up

Description

If a robot is started in low ambient temperatures, this results in increased friction in the gear unit. This can cause the motor current of an axis (or of more than one axis) to reach its maximum value. This stops the robot and the robot controller generates the error message *Regulator limit exceeded <axis number>*.

To avoid this, the motor current can be monitored during the warm-up phase. If a defined value is reached, the robot controller reduces the motion velocity. This, in turn, reduces the motor current.

The monitoring refers to PTP motions and PTP-CP approximate positioning blocks. CP motions (including Spline) are not monitored and their velocity is not reduced.

6.12.1 Configuring warm-up

Precondition

- Expert user group

Procedure

1. Open the file R1\Mada\$machine.dat.
2. Set the corresponding system variables to the desired values.
   (>>> 6.12.3 "System variables for warm-up" Page 132)
3. Close the file. Respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.

6.12.2 Warm-up sequence

Precondition

- $WARMUP_RED_VEL =TRUE
- Operating mode AUT or AUT EXT
- The robot is considered to be cold. This applies in the following cases:
  - Cold start
  - Or $COOLDOWN_TIME has expired.
  - Or $WARMUP_RED_VEL has been set from FALSE to TRUE.

Example

Sequence on the basis of the following example values in $machine.dat:

```
BOOL $WARMUP_RED_VEL = TRUE
REAL $WARMUP_TIME = 30.0
REAL $COOLDOWN_TIME = 120.0
INT $WARMUP_CURR_LIMIT = 95
INT $WARMUP_MIN_FAC = 60
REAL $WARMUP_SLEW_RATE = 5.0
```

1. The cold robot starts. The motor currents are monitored for 30 minutes ($WARMUP_TIME).
2. If the motor current of an axis exceeds 95% ($WARMUP_CURR_LIMIT) of the maximum permissible motor current, the monitoring is triggered. The robot controller then generates the message *Warm-up active* and reduces the internal override. The robot slows down and the motor current drops. The program override on the KCP remains unchanged!

   The internal override is reduced to a maximum of 60% ($WARMUP_MIN_FAC) of the programmed override. There is no way of influencing how quickly the internal override is reduced.
3. Once the monitoring is no longer triggered, the robot controller increases the internal override again. This is generally the case before the minimum $WARMUP_MIN_FAC has been reached! The robot accelerates again.

   Once a second, the robot controller edges back up towards the programmed override. $WARMUP_SLEW_RATE determines the rate of increase. In the example, the internal override is increased by 5% per second.
4. It is possible that the robot is still not warm enough and that the motor current thus exceeds the maximum $WARMUP_CURR_LIMIT once again. The robot controller reacts (within $WARMUP_TIME) the same way as the first time.
5. If the robot is warm enough for the robot controller to increase the internal override all the way back up to the programmed override, the robot controller deactivates the message *Warm-up active*.

6. After 30 minutes ($WARMUP_TIME), the robot is deemed to be warmed up and the motor currents are no longer monitored.

LOG file

The following events are logged in the file “Warmup.LOG” (path “KRC:\Robot-er\Log\”):

Entry	Meaning
Monitoring active	The motor currents are monitored.
Monitoring inactive	The motor currents are not monitored.
Controlling active	The velocity is reduced.
Controlling inactive	The velocity corresponds to the programmed override once again.

Example:

```
...
Date: 21.08.08 Time: 14:46:57 State: Monitoring active
Date: 21.08.08 Time: 14:54:06 State: Controlling active
Date: 21.08.08 Time: 14:54:07 State: Controlling inactive
Date: 21.08.08 Time: 18:23:43 State: Monitoring inactive
...
```

6.12.3 System variables for warm-up

The system variables for the warm-up can only be modified in the file $Machine.dat (path KRC:\R1\MADA).

If one of the values is outside the permissible range, the warm-up function is not active and the velocity is not reduced.

System variable	Description
$WARMUP_RED_VEL	■ **TRUE**: Warm-up functionality is activated. ■ **FALSE**: Warm-up functionality is deactivated (default). If $WARMUP_RED_VEL is set from FALSE to TRUE, this sets the runtime of the robot to zero. The robot is deemed to be cold, irrespective of how long it was previously under servo control.
$WARMUP_TIME	Time during which the motor currents are monitored by the warm-up function. If the cold robot is started, a runtime value is incremented. If the robot is not under servo control, the value is decremented. If the runtime is greater than $WARMUP_TIME, the robot is deemed to be warmed up and the motor currents are no longer monitored. ■ > 0.0 min
$COOLDOWN_TIME	If the warm robot is not under servo control, a standstill value is incremented. If the robot is under servo control, the value is decremented. If the standstill time is greater than $COOLDOWN_TIME, the robot is deemed to be cold and the motor currents are monitored. (Precondition: $WARMUP_RED_VEL = TRUE.) If the controller of a warm robot is shut down and restarted with a warm restart, the time the controller was switched off is counted as standstill time. ■ > 0.0 min

System variable	Description
$WARMUP_CURR_LIMIT	Maximum permissible motor current during warm-up (relative to the regular maximum permissible motor current) Regular maximum permissible motor current = ($CURR_LIM * $CURR_MAX) / 100 ■ 0 … 100%
$WARMUP_MIN_FAC	Minimum for the override reduction due to the warm-up function The internal override is reduced at most to the factor of the programmed override defined here. ■ 0 … 100%
$WARMUP_SLEW_RATE	Rate of increase for the increase in velocity Once the monitoring is no longer triggered, the robot controller increases the internal override again. The rate of increase is defined here. ■ > 0.0%/s

6.13 Collision detection

Collision detection is an extension and improvement of the torque monitoring function.
The torque monitoring function remains available. Programs in which torque monitoring is already defined can still be executed as before.
Alternatively, the lines with the torque monitoring in such programs can be deleted and collision detection can be used instead. Collision detection must not be used together with torque monitoring in a program.

Description

If the robot collides with an object, the robot controller increases the axis torques in order to overcome the resistance. This can result in damage to the robot, tool or other objects.

Collision detection reduces the risk of such damage. It monitors the axis torques. If these deviate from a specified tolerance range, the following reactions are triggered:

- The robot stops with a STOP 1.
- The robot controller calls the program tm_useraction. This is located in the Program folder and contains the HALT statement. Alternatively, the user can program other reactions in the program tm_useraction.

The robot controller automatically calculates the tolerance range. (Exception: no values are calculated in T1 mode.) A program must generally be executed 2 or 3 times before the robot controller has calculated a practicable tolerance range. The user can define an offset via the user interface for the tolerance range calculated by the robot controller.

If the robot is not operated for a longer period (e.g. over the weekend), the motors, gear units, etc., cool down. Different axis torques are required in the first few runs after such a break than in the case of a robot that is already at operating temperature. The robot controller automatically adapts the collision detection to the changed temperature.

Precondition

- In order to be able to use the collision detection function, acceleration adaptation must be activated.

  Acceleration adaptation is activated when system variable $ADAP_ACC **is not equal to** #NONE. (This is the default setting.) The system variable can be found in the file C:\KRC\Roboter\KRC\R1\MaDa\$ROBCOR.DAT.

- The tolerance range is only calculated for motion blocks that have been executed completely.
- To activate collision detection for a motion, the parameter **Collision detection** must be set to TRUE during programming. This can be seen from the addition `CD` in the program code:

```
PTP P2  Vel= 100 % PDAT1 Tool[1] Base[1] CD
```

The parameter **Collision detection** is only available if the motion is programmed via an inline form.
Information about collision detection for motions programmed without inline forms can be found in the documentation “KUKA.ExpertTech”.

Restrictions

- Collision detection is not possible for HOME positions and other global positions.
- Collision detection is not possible for external axes.
- Collision detection is not possible during backward motion.
- Collision detection is not possible in T1 mode.
- High axis torques arise when the stationary robot starts to move. For this reason, the axis torques are not monitored in the starting phase (approx. 700 ms).
- The collision detection function reacts much less sensitively for the first 2 or 3 program executions after the program override value has been modified. Thereafter, the robot controller has adapted the tolerance range to the new program override.

System variables

System variable	Description
$TORQ_DIFF $TORQ_DIFF2	The values of $TORQ_DIFF (torque) and $TORQ_DIFF2 (impact) are automatically calculated during program execution. These values are compared with the values from the previous program execution or with the default values. The highest value is saved. The values are always calculated, even when collision detection is deactivated. If collision detection is active, the system compares the values of $TORQ_DIFF and $TORQ_DIFF2 with the saved values during the motion.
$TORQMON_DEF[1] … $TORQMON_DEF[6]	Values for the tolerance range in program mode (per axis)* File C:\KRC\Roboter\KRC\MaDa\$CUSTOM.DAT
$TORQMON_COM_DEF[1] … $TORQMON_COM_DEF[6]	Values for the tolerance range in jog mode (per axis)* File C:\KRC\Roboter\KRC\MaDa\$CUSTOM.DAT
$COLL_ENABLE	Signal declaration. This output is set if the value of one of the $TORQMON_DEF[…] variables is less than 200. File: KRC:\STEU\Mada\$machine.dat
$COLL_ALARM	Signal declaration. This output is set if message 117 “Collision Detection axis <axis number>” is generated. The output remains set as long as $STOPMESS is active. File: KRC:\STEU\Mada\$machine.dat
$TORQMON_TIME	Response time for the collision detection. Unit: milliseconds. Default value: 0.0 File: C:\KRC\Roboter\KRC\Steu\MaDa\$CUSTOM.DAT.

*The width of the tolerance range is equal to the maximum torque [Nm] multiplied by the value in $TORQMON_... . The default value is 200. Unit: percent.

6.13.1 Calculating the tolerance range and activating collision detection

Precondition

- The acceleration adaptation is switched on.
- The load data have been entered correctly.
- In the program, the parameter **Collision detection** is set to TRUE for all motions that are to be monitored.
- If required: the desired collision response has been programmed in the program tm_useraction.

Procedure

1. Select the menu sequence **Configure** > **Tools** > **Collision detection**.
   (>>> 6.13.3 "Option window “Collision detection”" Page 136)
2. The box **KCP** must contain the entry **MonOff**. If this is not the case, press the **Deactivate** softkey.
3. Start the program and execute it several times. After 2 or 3 program executions, the robot controller has calculated a practicable tolerance range.
4. Press the **Activate** softkey. The box **KCP** in the **Collision detection** window now contains the entry **MonOn**.
   Save the configuration by pressing the **Close** softkey.

If required, the user can define an offset for the tolerance range.
(>>> 6.13.2 "Defining an offset for the tolerance range" Page 135)

The tolerance range must be recalculated in the following cases:
- The velocity has been modified.
- Points have been changed, added or removed.

6.13.2 Defining an offset for the tolerance range

Description

An offset for the torque and for the impact can be defined for the tolerance range. The lower the offset, the more sensitive the reaction of the collision detection. The higher the offset, the less sensitive the reaction of the collision detection.

Torque: The torque is effective if the robot meets a continuous resistance. Examples:

- The robot collides with a wall and pushes against the wall.
- The robot collides with a container. The robot pushes against the container and moves it.

Impact: The impact is effective if the robot meets a brief resistance. Example:

- The robot collides with a panel which is sent flying by the impact.

If the collision detection reacts too sensitively, do not immediately increase the offset. Instead, recalculate the tolerance range first and test whether the collision detection now reacts as desired.
(>>> 6.13.1 "Calculating the tolerance range and activating collision detection" Page 135)

Procedure

1. Select program.
2. Select the menu sequence **Configure** > **Tools** > **Collision detection**.
   (>>> 6.13.3 "Option window “Collision detection”" Page 136)
3. The offset for a motion can be modified while a program is running: if the desired motion is displayed in the **Collision detection** window, press the

01 or **02** status key. The window remains focused on this motion. Modify the offset using the **01** and **02** status keys.

Alternatively, a block selection to the desired motion can be carried out.

4. Save the change by pressing the **Save** softkey.
5. Save the configuration by pressing the **Close** softkey.
6. Set the original operating mode and program run mode.

6.13.3 Option window “Collision detection”

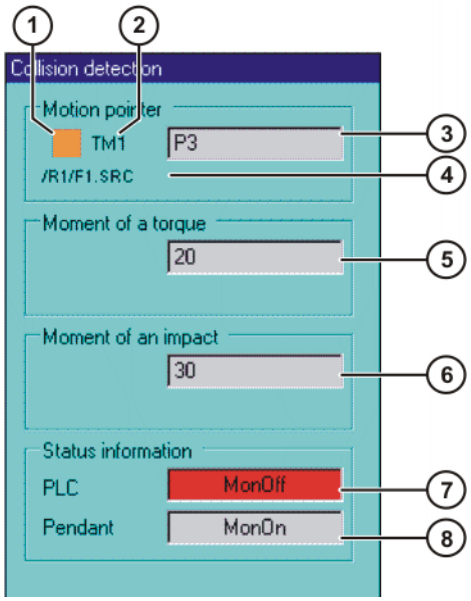


Fig. 6-11: Option window “Collision detection”

The values in the option window **Collision detection** do not always refer to the current motion. Deviations are particularly possible in the case of points which are close together and approximated motions.

Item	Description
1	The button indicates the status of a motion. ■ Red: the current motion is not monitored. ■ Green: the current motion is monitored. ■ Orange: the **01** or **02** status key has been pressed. The window remains focused on the motion and the offset can be modified. The change can be applied by pressing the **Save** softkey. ■ Pixelated: a program must generally be executed 2 or 3 times before the robot controller has calculated a practicable tolerance range. All the while the robot controller is in the learning phase, the button remains pixelated.
2	Number of the TMx variable The robot controller creates a TMx variable for each motion block in which the parameter **Collision detection** is set to TRUE. TMx contains all the values for the tolerance range of this motion block. If 2 motion blocks refer to the same point Px, the robot controller creates 2 TMx variables.
3	Point name
4	Path and name of the selected program

Item	Description
5	Offset for the torque. The lower the offset, the more sensitive the reaction of the collision detection. Default value: 20. **N.A.**: the option **Collision detection** in the inline form is set to FALSE for this motion.
6	Offset for the impact. The lower the offset, the more sensitive the reaction of the collision detection. Default value: 30. **N.A.**: the option **Collision detection** in the inline form is set to FALSE for this motion.
7	This box is only active in “Automatic External” mode. It appears gray in all other modes. **MonOn**: collision detection has been activated by the PLC. If collision detection is activated by the PLC, the PLC sends the input signal sTQM_SPSACTIVE to the robot controller. The robot controller responds with the output signal sTQM_SPSSTATUS. The signals are defined in the file $config.dat. **Note:** Collision detection is only active in Automatic External mode if both the **PLC** box and the **KCP** box show the entry **MonOn**.
8	**MonOn**: collision detection has been activated from the KCP. **Note:** Collision detection is only active in Automatic External mode if both the **PLC** box and the **KCP** box show the entry **MonOn**.

The following status keys are available:

Status key	Description
	Status key **01** modifies the offset for the torque.
	Status key **02** modifies the offset for the impact.

The following softkeys are available:

Softkey	Description
Activate	Activates collision detection. This softkey is not displayed if the torque or impact has been changed, but the changes have not yet been saved.
Deactivate	Deactivates collision detection. This softkey is not displayed if the torque or impact has been changed, but the changes have not yet been saved.
Save	Saves changes to the torque and/or impact.
Cancel	Rejects changes to the torque and/or impact.

6.13.4 Torque monitoring

Collision detection is an extension and improvement of the torque monitoring function.
The torque monitoring function remains available. Programs in which torque monitoring is already defined can still be executed as before.
Alternatively, the lines with the torque monitoring in such programs can be deleted and collision detection can be used instead. Collision detection must not be used together with torque monitoring in a program.

Description

Differences between torque monitoring and collision detection:

- The tolerance range is not automatically calculated by the robot controller, but must be defined by the user.
- The tolerance range only refers to the torque. No values can be defined for the impact.
- The robot controller cannot automatically adapt the tolerance range to changed temperatures.
- If a collision is detected, the robot stops with a STOP 1. It is not possible to call a user-defined program.

Overview

Step	Description
1	Determine suitable values for torque monitoring. (>>> 6.13.4.1 "Determining values for torque monitoring" Page 138)
2	Program torque monitoring. (>>> 6.13.4.2 "Programming torque monitoring" Page 138)

6.13.4.1 Determining values for torque monitoring

Description

The maximum torque deviation that has occurred can be determined as a percentage by means of the system variable $TORQ_DIFF[...].

Procedure

1. Select the menu sequence **Monitor** > **Variable** > **Single**.
2. Set the value of the variable $TORQ_DIFF[…] to 0.
3. Execute the motion block and read the variable again. The value corresponds to the maximum torque deviation.
4. Set the variable for the monitoring of the axis to this value plus a safety margin of 5 - 10%.

Only the value 0 can be assigned to the variables $TORQ_DIFF[...].

6.13.4.2 Programming torque monitoring

Precondition

- In order to be able to use the collision detection function, acceleration adaptation must be activated. Acceleration adaptation is activated when system variable $ADAP_ACC **is not equal to** #NONE. (This is the default setting.) The system variable can be found in the file C:\KRC\Roboter\KRC\R1\MaDa\$ROBCOR.DAT.
- A program is selected.

Procedure

1. Position the cursor in the line before the motion for which the torque monitoring is to be programmed.

2. Select the menu sequence **Commands** > **Moveparams** > **Torquemon.**. An inline form is opened.

Fig. 6-12

3. In the **TORQMON** box, select the entry **SetLimits**.

Fig. 6-13

4. For each axis, enter the amount by which the command torque may deviate from the actual torque.
5. Press the **Cmd OK** softkey.
6. If a response time for the torque monitoring is to be defined:
   Set the variable $TORQMON_TIME to the desired value. Unit: milliseconds. Default value: 0.

The values are automatically reset to the default value 200 in the following cases:

- Reset
- Block selection
- Program deselection

6.14 Configuring inline forms for motions

6.14.1 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.

6.14.2 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.

Caution!
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:
  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**.

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): %

6.15 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

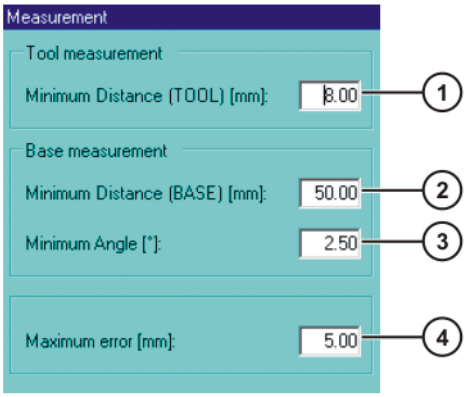


Fig. 6-14: Error tolerances

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**.

6.16 Backward motion

There are two methods for executing a program backwards:

- TRACE (>>> 6.16.1 "TRACE method" Page 142)
- SCAN (>>> 6.16.2 "SCAN method" Page 143)

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.

6.16.1 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** (>>> 6.16.4 "TRACE section" Page 144), 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.

6.16.2 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.

6.16.3 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

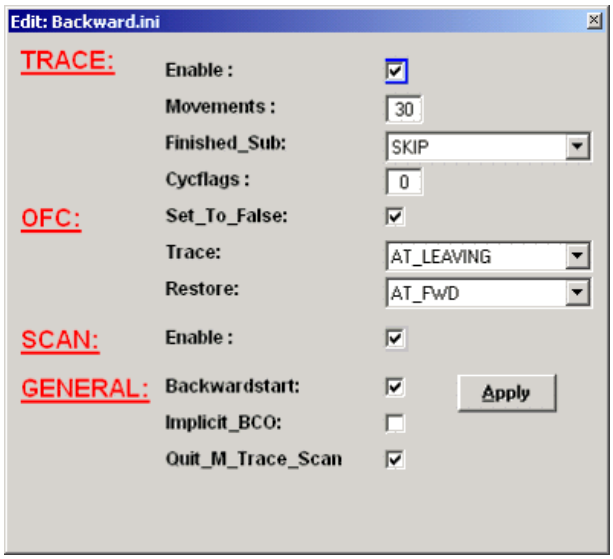


Fig. 6-15: Edit: Backward.ini window

- **TRACE** section (>>> 6.16.4 "TRACE section" Page 144)
- **OFC** section (>>> 6.16.5 "OFC section" Page 144)
- **SCAN** section (>>> 6.16.6 "SCAN section" Page 145)
- **GENERAL** section (>>> 6.16.7 "GENERAL section" Page 146)

6.16.4 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**

6.16.5 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.

6.16.6 SCAN section

Parameter	Description
Enable	■ **TRUE**: SCAN method activated ■ **FALSE**: SCAN method deactivated

6.16.7 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.

6.17 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.

```
 1 <configuration>
 2   <authenticationManagement>
 3      <AuthorizationConfiguration xmlns:xsd="http://www.w3.org/
2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
 4         <AutoLock>ModeOP2Aut ModeOP2Ext</AutoLock>
 5         <CanLockSystem>false</CanLockSystem>
 6         <MethodList>
 7            <AuthenticationMethod>
 8               <AuthenticationType>SelectionOnly</
AuthenticationType>
 9               <UserListName>secondaryUserList</UserListName>
10               <LeaseTime>300</LeaseTime>
11            </AuthenticationMethod>
12            <AuthenticationMethod>
13               <AuthenticationType>UsernameOnly</
AuthenticationType>
14               <LeaseTime>300</LeaseTime>
15             </AuthenticationMethod>
16         </MethodList>
17         <DefaultUser>
18            <Name>AutoKrcOperator</Name>
19            <UserLevel>5</UserLevel>
20            <DisplayName>KrcSecurity#Operator</DisplayName>
21            <TranslateDisplayName>true</TranslateDisplayName>
22            <Password>kukaAuto</Password>
23            <UsePasswordAutomaticaly>true</
UsePasswordAutomaticaly>
24         </DefaultUser>
25         <AnnouncementTime>300</AnnouncementTime>
26      </AuthorizationConfiguration>
27   </authenticationManagement>
28    <secondaryUserList>
29       <UserList xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
30          <UserInfo>
31             <Name>AutoKrcProgrammer</Name>
32             <UserLevel>10</UserLevel>
33             <DisplayName>KrcSecurity#Programmer</DisplayName>
34             <TranslateDisplayName>true</TranslateDisplayName>
35             <Password>kukaAuto</Password>
36             <UsePasswordAutomaticaly>true</
UsePasswordAutomaticaly>
37          </UserInfo>
38          <UserInfo>
39             <Name>KrcExpertProgrammer</Name>
40             <UserLevel>20</UserLevel>
41             <DisplayName>KrcSecurity#ExpertProgrammer</
DisplayName>
42             <TranslateDisplayName>true</TranslateDisplayName>
43          </UserInfo>
44          <UserInfo>
45             <Name>KrcAdministrator</Name>
46             <UserLevel>30</UserLevel>
47             <DisplayName>KrcSecurity#KrcAdministrator</
DisplayName>
48             <TranslateDisplayName>true</TranslateDisplayName>
49          </UserInfo>
50       </UserList>
51    </secondaryUserList>
52 </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: kukaAuto)	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: kukaAuto)	Key
Sicherheitsinbetrieb-nehmer / SafetyMainte-nance (29)	KrcSafetyMainte-nance	AutoKrcSafetyMaint	SafetyMainte-nance
Administrator / Adminis-trator (30)	KrcAdministrator	AutoKrcAdministrator	KrcAdministrator

6.18 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 150)

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. (>>> 6.18.1 "Example of setting up a new user group" Page 151)
2	Set up password for new user group. (>>> 6.18.2 "Setting up a password for a new user group" Page 153)
3	Define new user group as default user group (optional). (>>> 6.19 "Defining the default user group" Page 154)

6.18.1 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. (>>> 6.18.1.1 "Defining a user group" Page 152)
2	Define position of the softkey. (>>> 6.18.1.2 "Defining the position of the softkey" Page 152)
3	Enable function. (>>> 6.18.1.3 "Enabling the function" Page 153)

6.18.1.1 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 150))

6.18.1.2 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”.

6.18.1.3 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.

6.18.2 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: (>>> 6.6 "Changing the password" Page 119)

By default, the values entered under **HKEY_CURRENT_USER\...** in the registry database are not overwritten when the KSS is installed.

6.19 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.

6.20 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. (>>> 6.20.1 "Configuring CELL.SRC" Page 154)
2	Configuration of the inputs/outputs of the Automatic External interface. (>>> 6.20.2 "Configuring Automatic External inputs/outputs" Page 155)
3	Only if error numbers are to be transmitted to the higher-level controller: configuration of the P00.DAT file. (>>> 6.20.3 "Transmitting error numbers to the higher-level controller" Page 161)

6.20.1 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**.

6.20.2 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:

- (>>> 6.20.2.1 "Automatic External inputs" Page 157)
- (>>> 6.20.2.2 "Automatic External outputs" Page 159)

Description

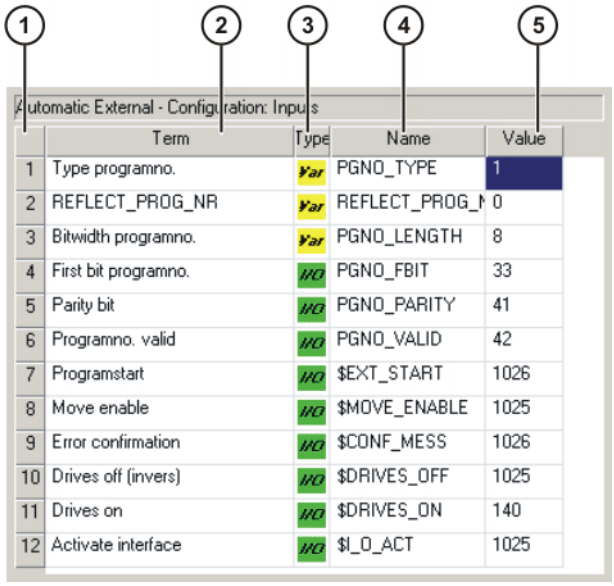


Fig. 6-16: Configuring Automatic External inputs

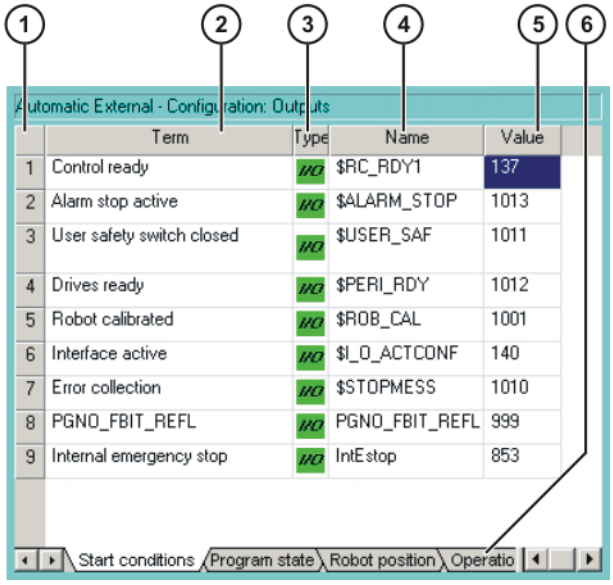


Fig. 6-17: Configuring Automatic External outputs

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 status ■ Robot position ■ Operating mode

The following softkeys are available:

Softkey	Description
Monitor	Switches to the Automatic External display. (>>> 4.16.4 "Displaying inputs/outputs for Automatic External" Page 68)
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.

6.20.2.1 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. (>>> 6.20.2.2 "Automatic External outputs" Page 159)

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].

6.20.2.2 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 **IntEstop**.

- Both outputs are FALSE: the EMERGENCY STOP was triggered on the KCP.
- **$ALARM_STOP** is FALSE, **IntEstop** 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, Operator safety or Command velocity)

PGNO_FBIT_REFL Output representing the first bit of the program number. Precondition: The variable REFLECT_PROG_NR has the value 1. (>>> 6.20.2.1 "Automatic External inputs" Page 157)

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.

IntEstop 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 **IntEstop**.

- Both outputs are FALSE: the EMERGENCY STOP was triggered on the KCP.
- **$ALARM_STOP** is FALSE, **IntEstop** 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.

6.20.3 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:

```
 1  DEFDAT  P00
 2
 3  BOOL PLC_ENABLE=TRUE ; Enable error-code transmission to plc
 4  INT I
 5  INT F_NO=1
 6  INT MAXERR_C=1 ; maximum messages for $STOPMESS
 7  INT MAXERR_A=1 ; maximum messages for APPLICATION
 8  DECL STOPMESS MLD
 9  SIGNAL ERR $OUT[25]  TO $OUT[32]
10  BOOL FOUND
11
12  STRUC PRESET INT OUT,CHAR PKG[3],INT ERR
13  DECL PRESET P[255]
    …
26  P[1]={OUT 2,PKG[] "P00",ERR 10}
    …
30  P[128]={OUT 128,PKG[] "CTL",ERR 1}
    …
35  STRUC ERR_MESS CHAR P[3],INT E
36  DECL ERR_MESS ERR_FILE[64]
37  ERR_FILE[1]={P[] "XXX",E 0}
    …
96  ERR_FILE[64]={P[] "XXX",E 0}
97  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.

6.20.4 Signal diagrams

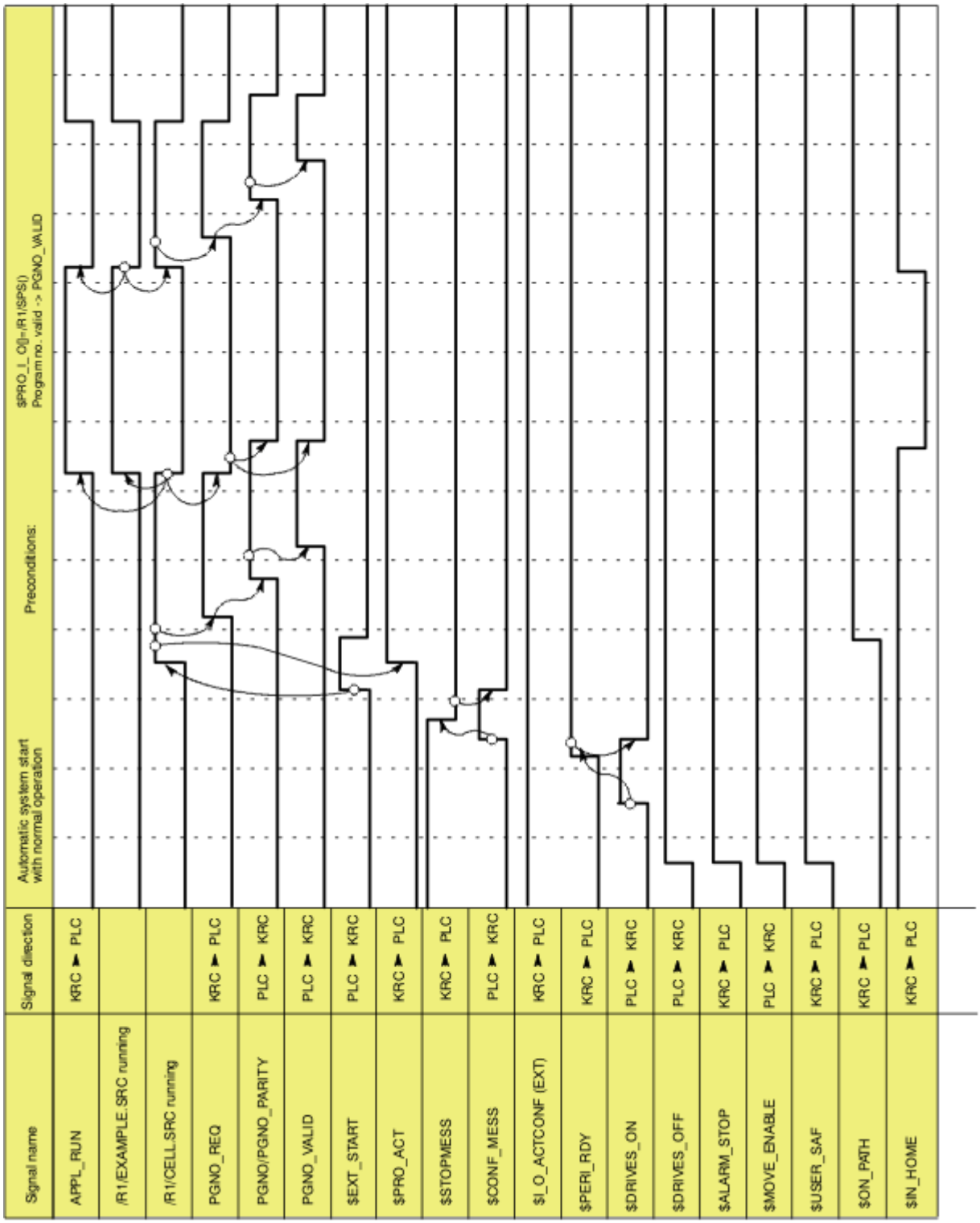


Fig. 6-18: Automatic system start and normal operation with program number acknowledgement by means of PGNO_VALID

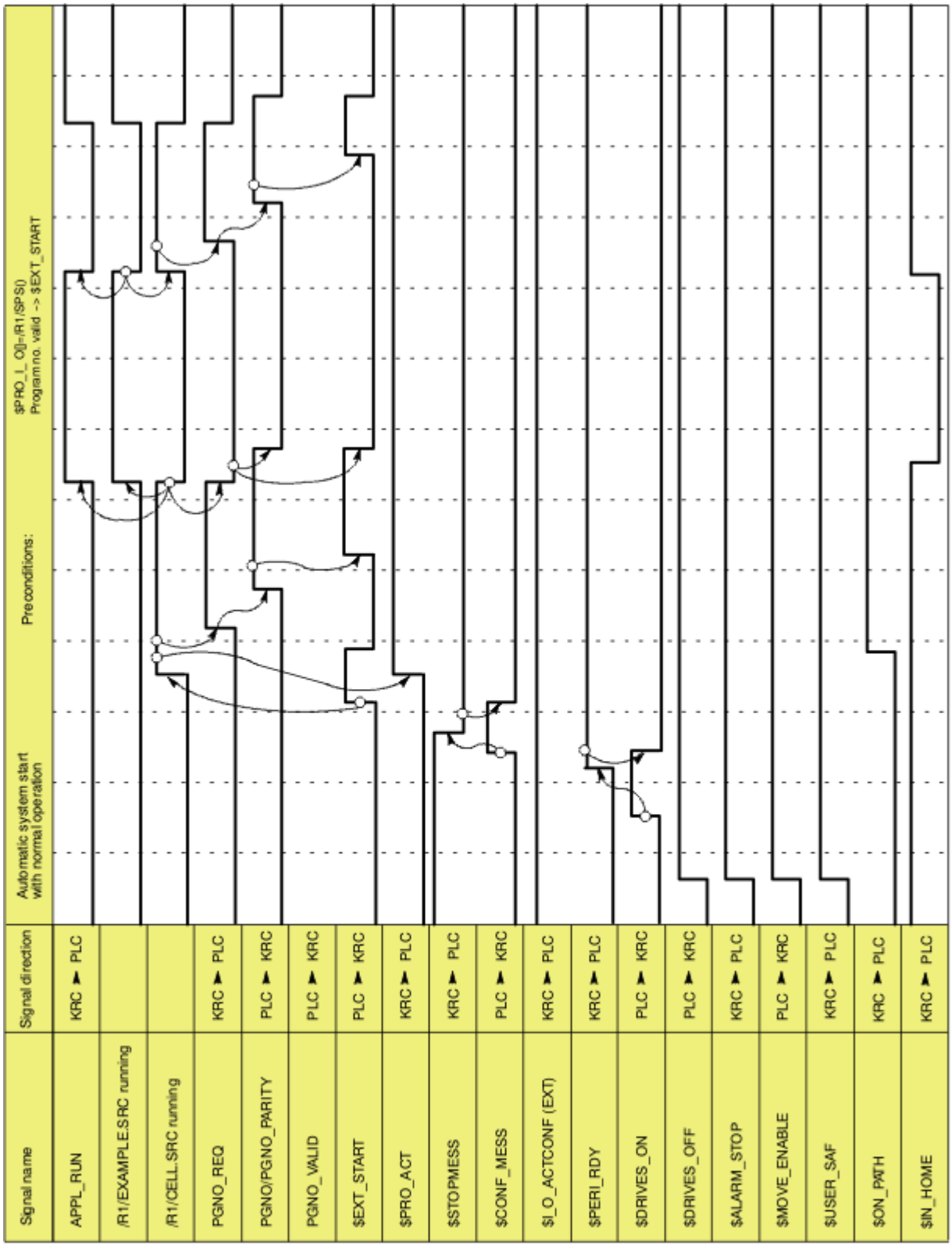


Fig. 6-19: Automatic system start and normal operation with program number acknowledgement by means of $EXT_START

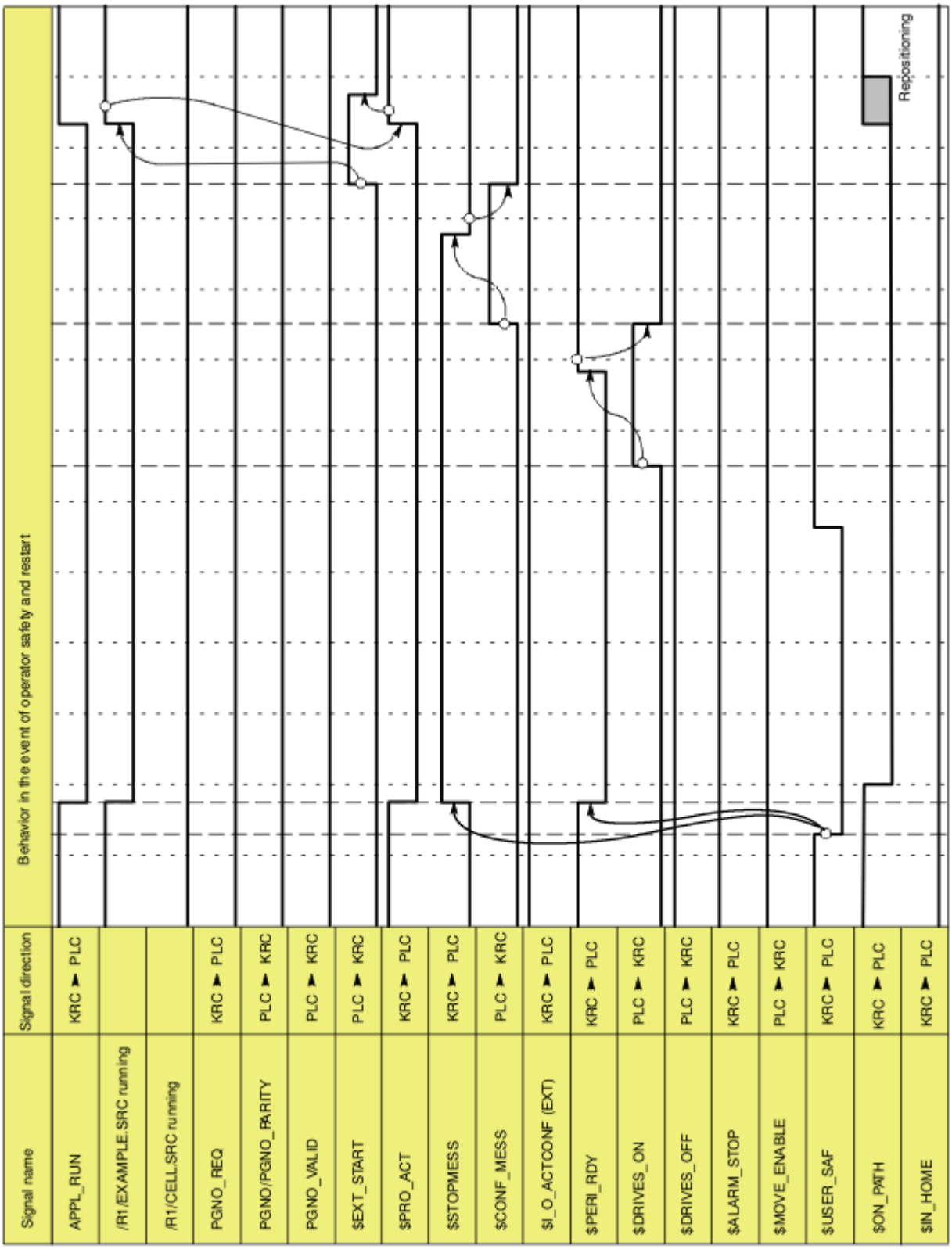


Fig. 6-20: Restart after dynamic braking (operator safety and restart)

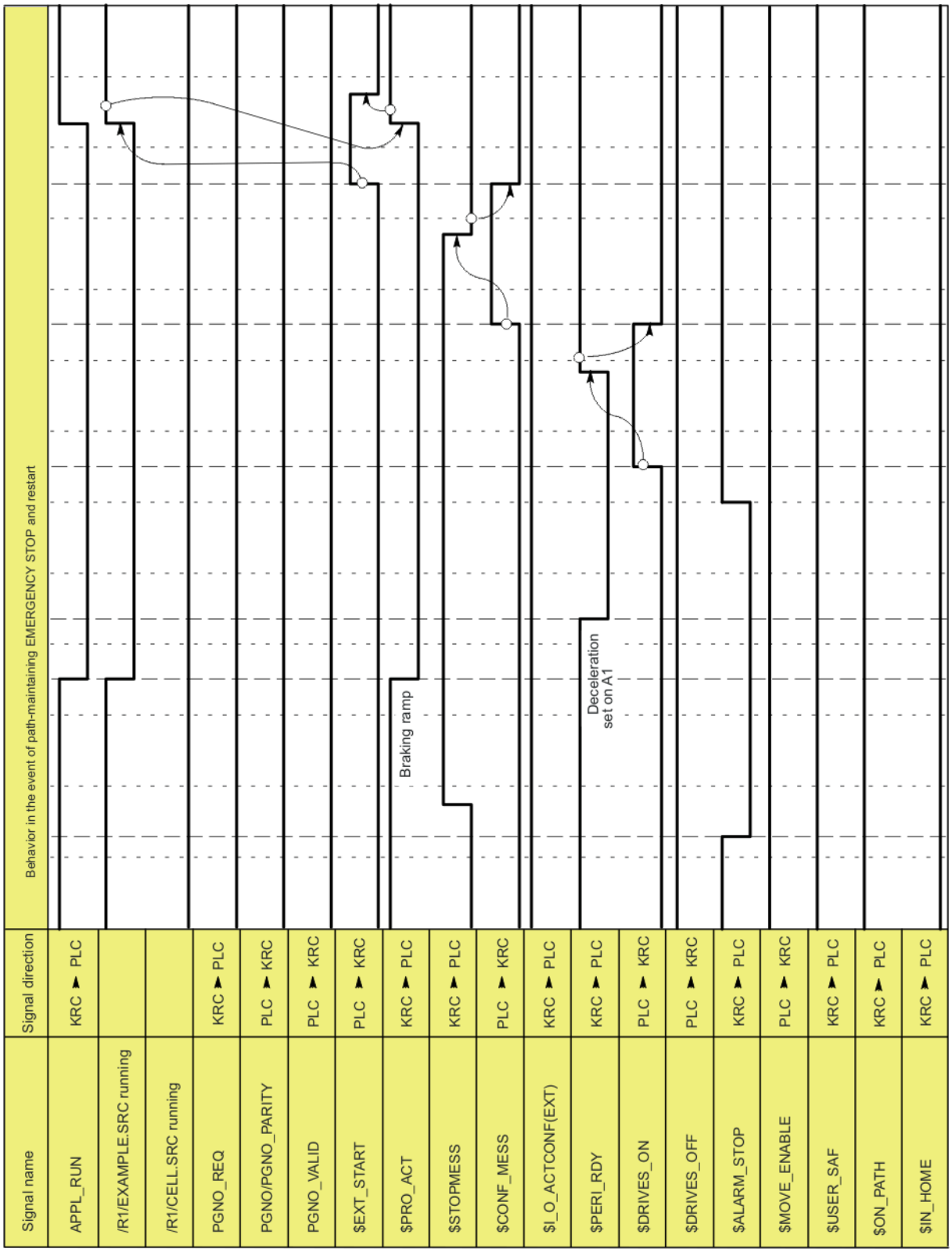


Fig. 6-21: Restart after path-maintaining EMERGENCY STOP

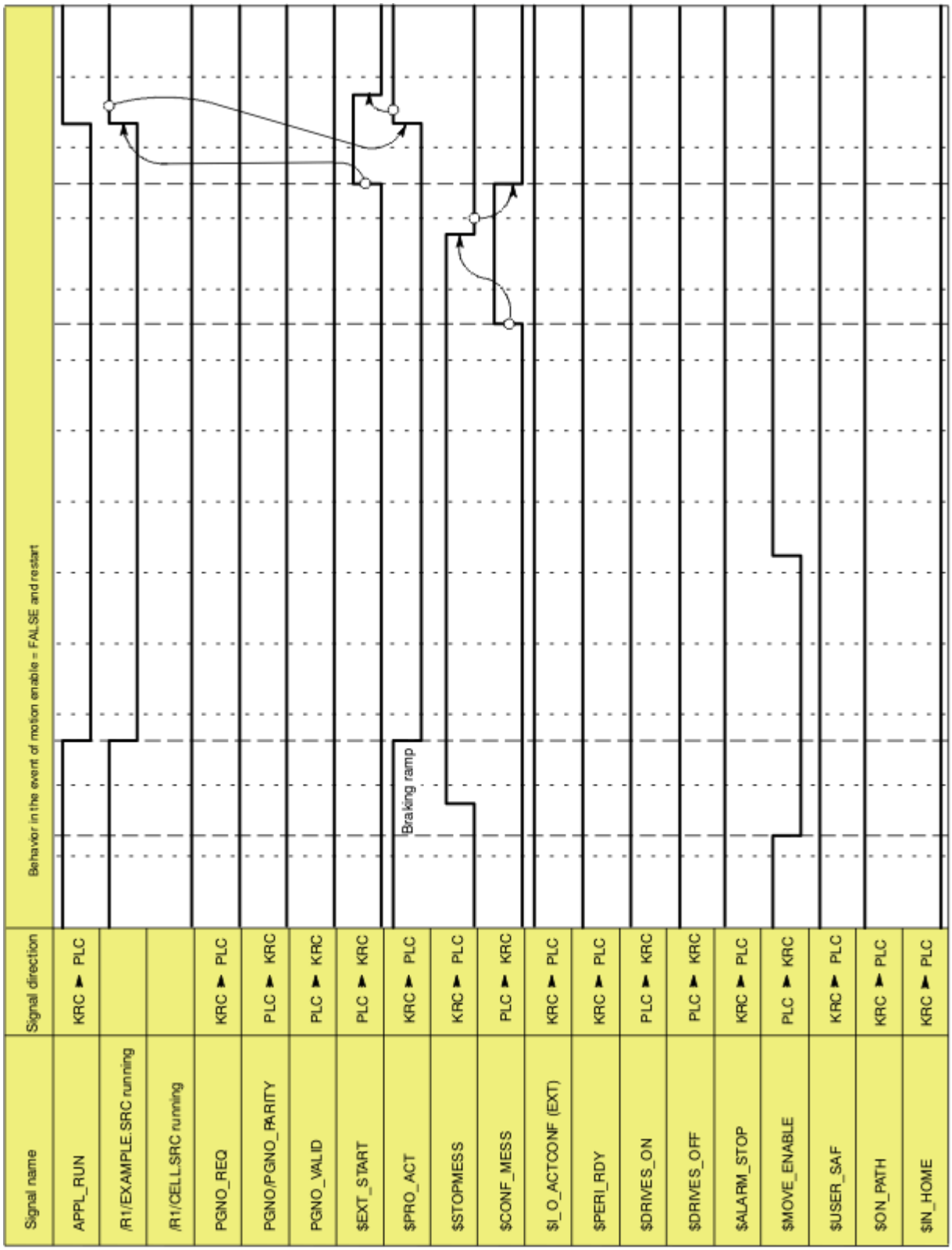


Fig. 6-22: Restart after motion enable

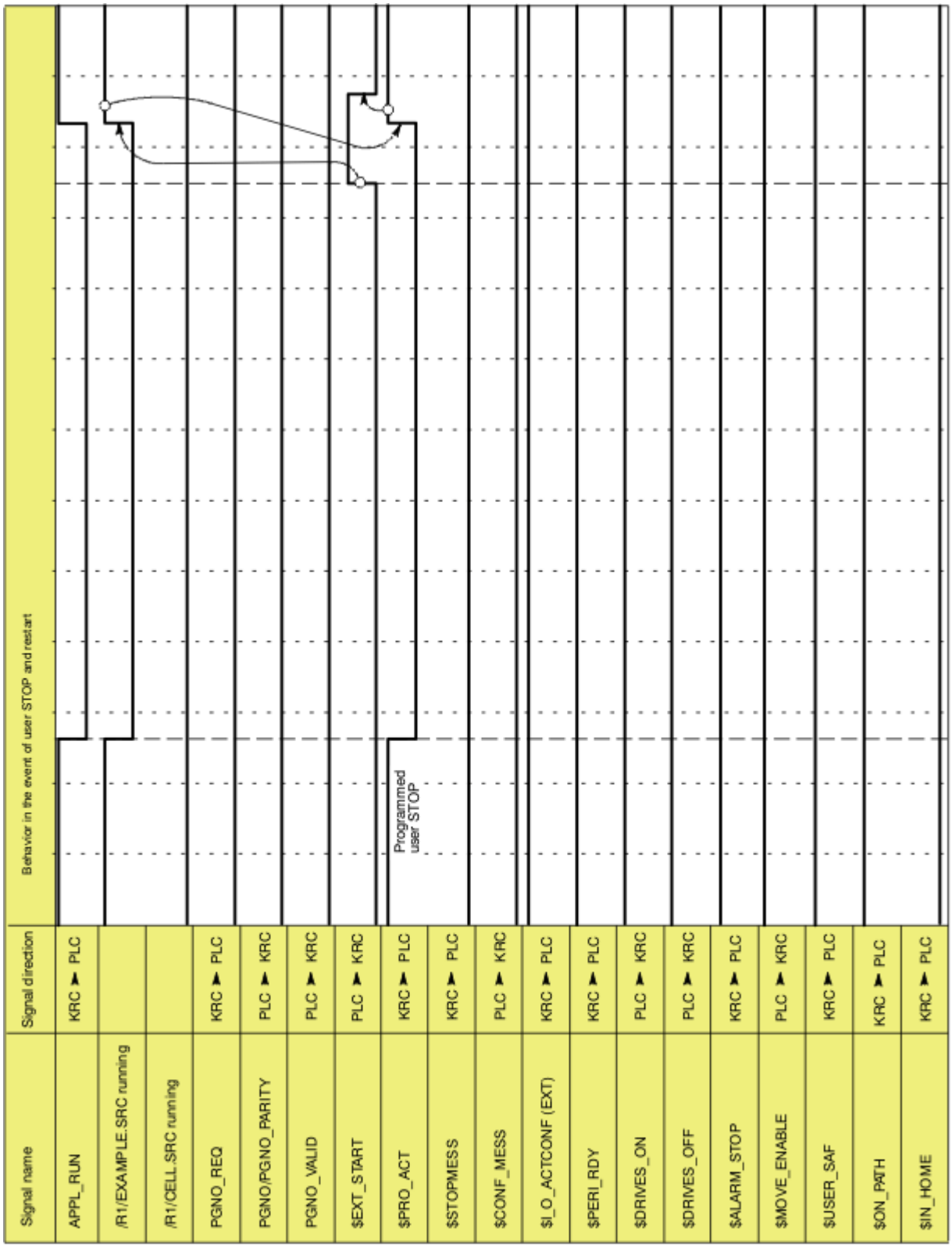


Fig. 6-23: Restart after user STOP

6.21 Torque mode

6.21.1 Overview of torque mode

Description

It is possible to switch either individual axes or several axes to torque mode. Torque mode means:

- The axis can push or pull with a defined torque against a resistance.
  Example: application of a defined pressure on the workpiece by an electric motor-driven spot welding gun.
- The axis can be set to “soft”. It can then be moved by application of an external force. It can be pushed away, for example.
  Example: the robot must grip a workpiece in a press that is then ejected by the press. In order for the robot to be able to yield and absorb the ejector stroke, the affected axis is set to “soft”.

Restrictions

Axis	Torque mode allowed?
A1	■ Wall-mounted robots: torque mode must not be used. ■ Floor- and ceiling-mounted robots: torque mode may be used.
A2	Torque mode must not be used.
A3 … A6	KUKA Roboter GmbH must be consulted if torque mode is to be used for these axes.
External axes	KUKA Roboter GmbH must be consulted if torque mode is to be used for external axes.

Diagonal ejector motion:

A diagonal ejector motion cannot be absorbed. If, however, the robot is positioned in such a way that the mounting plane is parallel to the ejector motion plane (e.g. inclined installation of the robot), torque mode is possible for A1.

Inclined installation of the robot is only permissible up to a certain angle of inclination. Further information is contained in the robot operating or assembly instructions.

6.21.1.1 Using torque mode

Description

Torque mode must be activated for the desired axis. The current for this axis is then limited, thereby reducing the torque.

Warning!
Current limitation can result in the axis no longer being able to achieve the necessary torque for holding, braking or moving the axis. Death to persons, severe physical injuries or considerable damage to property may result.

- The current for an axis may only be limited if torque mode has been activated for this axis.
- Limit the current gradually to determine the required limit. Do not limit the current further than necessary.

Warning!
If an axis is in torque mode, no brakes are applied at the end of the motion. This also applies to the brakes of axes that are not in torque mode, including external axes. After the end of the motion, the robot remains under servo control in order to maintain its position. In this way, the surface can reach temperatures that can cause burns to the skin. Contact must be avoided. Appropriate safety precautions must be taken, e.g. protective gloves must be worn.

Procedure

1. Activate torque mode for the axis ($TORQUE_AXIS).
2. Reduce the current limit ($CURR_RED).
3. If the axis is to be set to “soft”: move any axis so that the brakes are released. At the end of the motion, the brakes of all axes remain open.

   Alternatively, a “motion” to the current position can be executed. The robot does not move, but the brakes are released.
4. Optionally: generate a signal indicating that the axis is stationary (e.g. signal to an injection molding machine).
5. Perform the desired action, e.g. move to workpiece and build up pressure or push the axis away.
6. Optionally: wait for a signal to end torque mode.
7. Deactivate torque mode again ($TORQUE_AXIS). The current limits are automatically set to 100%.

   Only in the case of deactivation in the robot program or Submit program: the command position is automatically adjusted to the actual position.
8. If this is not carried out automatically: adjust the command position to the actual position.

6.21.1.2 Robot program example: setting A1 to “soft” in both directions

Description

This simple example illustrates the basic principle of torque mode.

In this example, A1 is to be set to “soft” in both directions. For this purpose, both the positive and negative current limits are set to 0%. This allows the axis to be moved by application of an external force.

Program

```
   ...
1  PTP {A1 10}
2  $TORQUE_AXIS = 'B000001'
3  $TORQ_VEL[1] = 130
4  $CURR_RED[1,1] = 0
5  $CURR_RED[1,2] = 0
6  PTP {A1 11}
   ...
7  $TORQUE_AXIS = 0
8  PTP {A1 -20}
    ...
```

Line	Description
2	Activate torque mode for A1.
3	Set the desired limit for velocity monitoring.
4	Set the positive current limit of A1 to 0.
5	Set the negative current limit of A1 to 0.
6	Execute a “motion” to release the brakes. (Since both current limits are set to 0, the robot will not actually move.) A1 can now be moved by application of an external force.

Line	Description
7	Deactivate torque mode again. During deactivation, the following occurs automatically: ■ The command position is adjusted to the actual position. ■ The current limits are set to 100%.
8	Move to the next position. (The motion from line 6 is not belatedly executed now, as the command-actual adjustment has been carried out in line 7.)

6.21.2 System variables for torque mode

6.21.2.1 $TORQUE_AXIS

Syntax

$TORQUE_AXIS = *Bit array*

Explanation of the syntax

Element	Description
Bit array	Bit array with which torque mode can be activated and deactivated for one or more axes. Examples: ■ Activate for A1: `$TORQUE_AXIS = 'B00001'` ■ Activate for E1: `$TORQUE_AXIS = 'B1000000'` ■ Deactivate for all axes: `$TORQUE_AXIS = 0`

Bit n	12 …	5	4	3	2	1	0
Axis	E6 …	A6	A5	A4	A3	A2	A1

Description

Monitoring:

When torque mode is activated, the following monitoring functions are disabled for the affected axis:

- Command value
- Stopped
- Positioning time
- Motor blocked

In order to be able still to detect hardware defects or sagging of the axis, the actual velocity continues to be monitored.

- T1: The monitoring limit set in the machine data applies.
- T2 and Automatic: The monitoring limit can be set using $TORQ_VEL.

Validity:

Assignment in the robot program:

- If $TORQUE_AXIS is set, it applies to all subsequent motions in the robot program (synchronous and asynchronous).
- The value remains valid until a different value is set in a robot program, Submit program or subprogram.
- When torque mode is activated, the monitoring functions of the affected axes are disabled, even if these axes are currently asynchronous and moved by a Submit program.

Assignment in the Submit program:

- If $TORQUE_AXIS is set, it applies to all subsequent motions in the Submit program (only asynchronous).

  If, at the same time, a robot program is active or interrupted, the assignment also applies there.
- When torque mode is deactivated, the following system variables are adapted to the current position: $AXIS_RET, $AXIS_BACK, $AXIS_FOR.

  If, at the same time, a robot program is active or interrupted, the adaptation also applies there.

Setting $TORQUE_AXIS in a Submit program is only recommended if no robot program is selected. This is the case of $PRO_STATE1==#P_FREE.

Assignment in other cases:

- During deactivation, the monitoring functions are immediately re-enabled without waiting for the end of the motion.
- During deactivation, the command position is not adapted to the actual position. The user must perform the adaptation manually.

  (>>> 6.21.3.4 "Robot interrupt program example" Page 179)
- In the case of assignment via a trigger subprogram or interrupt:

  The assignment is valid immediately, but only until the next exact positioning. Furthermore, the assignment is applied from the advance run onward, i.e. for all motions that are planned after the assignment. If there are already motions planned after the next exact positioning, the old value still applies to them!

  (>>> 6.21.3.6 "Negative example: robot program with trigger subprogram" Page 181)

Advance run stop:

An assignment to $TORQUE_AXIS in a robot program triggers an advance run stop.

Automatic modifications:

When torque mode is deactivated:

- The current limits are set to 100%.
- Only in the case of deactivation in the robot program or Submit program: the command position is adjusted to the actual position.

Torque mode is automatically deactivated in the following cases:

- Program is selected.
- Program is reset.
- Block selection
- The KRL statement RESUME is executed.
- If a running program is stopped and the robot is then jogged manually.

  If the program is then restarted, torque mode is automatically activated again. The robot moves back to the position at which it left the programmed path.

  Note: If the axis was set to “soft”, it does not move. The robot controller nonetheless internally executes a BCO run, i.e. the Start key must be held down until the programmed path is reached (BCO).

6.21.2.2 $CURR_RED

Syntax $CURR_RED[*Axis*, *y*] = *Limit*

Explanation of the syntax

Element	Description
Axis	Data type: INT ■ **1 … 6**: axis A1 ... A6 ■ **7 … 12**: external axis E1 … E6
y	Data type: INT ■ **1**: positive limit ■ **2**: negative limit
Limits	Data type: REAL Current limit of axes 1 - 12 in % of maximum current ■ **1 … 100%**

Description

This system variable can be used to modify the limit of the speed controller output and thus of the current.

Warning!
Current limitation can result in the axis no longer being able to achieve the necessary torque for holding, braking or moving the axis. Death to persons, severe physical injuries or considerable damage to property may result.

- The current for an axis may only be limited if torque mode has been activated for this axis.
- Limit the current gradually to determine the required limit. Do not limit the current further than necessary.

The direction of motion of the axis is not indicative of whether the current needs to be limited in the positive or negative direction, i.e. if an axis is to be set to “soft” in its positive direction of motion, this does not automatically mean that the current has to be limited in the positive direction.
The direction in which the current has to be limited depends on the specific application.

Validity:

Assignment in the robot program:

- The value becomes valid when the next motion is executed.
- Values set in the robot interpreter only apply there.

Assignment in the Submit program:

- The value becomes valid when the next asynchronous motion in the Submit interpreter is executed.
- Values set in the Submit interpreter only apply there.
- $CURR_RED is only valid for asynchronous axes. $CURR_RED can also be set for synchronous axes, but has no effect, as no synchronous axes can be moved in the Submit program. If a synchronous axis is subsequently switched to asynchronous mode, the previously set $CURR_RED value becomes valid.

Assignment in other cases:

- If $CURR_RED is set in the trigger, the value immediately becomes valid.

Automatic modifications:

$CURR_RED is automatically reset to 100% in the following cases:

- Program is selected.
- Program is reset.
- Block selection

- The KRL statement RESUME is executed.
- If torque mode is deactivated with $TORQUE_AXIS, the current limits of the affected axes are reset to 100%. There is thus no need to reset $CURR_RED separately.

Advance run stop:

An assignment to $CURR_RED does not trigger an advance run stop.

Example 1

Robot program:

```
...
$CURR_RED[7,1] = 20
PTP {...}
PTP {...}
$CURR_RED[7,1] = 60
PTP {...}
PTP {...}
...
```

Submit program:

```
...
$CURR_RED[7,1] = 40
ASYPTP {...}
...
ASYPTP {...}
...
```

In the robot program, the positive current limit of external axis E1 is set to 20%. This value now applies from the next motion onwards until it is set to 60%.

If, at any point, E1 is switched to asynchronous mode and then programmed in the Submit program as shown above, the current limit of 40% only applies to the subsequent ASYPTP motion(s) in the Submit program. If the axis is subsequently moved again in the robot interpreter, the last value set in the robot program applies.

Example 2

Robot program:

```
...
$CURR_RED[7,1] = 20
PTP {...}
TRIGGER WHEN DISTANCE=0 DELAY=0 DO UP1() PRIO=4
PTP {...}
$CURR_RED[7,1] = 60
PTP {...}
...
```

Trigger subprogram UP1:

```
...
$CURR_RED[7,1] = 40
...
```

In the robot program, the positive current limit of external axis E1 is set to 20%. This value is valid from the next motion onwards.

If the current limit is set to 40% in the trigger, the value is valid immediately. It also remains valid for subsequent motions until a different value is programmed. This is the case in the example when $CURR_RED is set to 60.

6.21.2.3 $TORQ_VEL

Syntax

$TORQ_VEL[*Axis*] = *Velocity limit*

Explanation of the syntax

Element	Description
Axis	Data type: INT ■ **1 … 6**: axis A1 ... A6 ■ **7 … 12**: external axis E1 … E6
Velocity limit	Data type: REAL ■ **0 … 150 %** Limit for the actual velocity as a % of the maximum velocity.

Description

When torque mode is activated, the following monitoring functions are disabled for the affected axis:

- Command value
- Stopped
- Positioning time
- Motor blocked

In order to be able still to detect hardware defects or sagging of the axis, the actual velocity continues to be monitored. The maximum permissible velocity for the operating modes T2 and Automatic can be set in the program by means of $TORQ_VEL. If this velocity is exceeded, the drives are switched off and a message is generated. (For T1, the velocity set in the machine data applies. It cannot be influenced by means of $TORQ_VEL.)

Advance run stop:

An assignment to $TORQ_VEL does not trigger an advance run stop.

6.21.2.4 $CURR_ACT

Syntax

$CURR_ACT[*Axis*] = *Current*

Explanation of the syntax

Element	Description
Axis number	Data type: INT ■ **1 … 6**: axis A1 ... A6 ■ **7 … 12**: external axis E1 … E6
Current	Data type: REAL Current value of current of axes A1 to A12 as % of maximum amplifier current. ■ **-100 … +100 %**

Description

This variable is write-protected. It can be used to limit $CURR_RED to the current value set for the current.

6.21.2.5 Overview: writability of the system variables

Can be written to by...	$CURR_ACT	$CURR_RED	$TORQUE_AXIS	$TORQ_VEL
Robot program	No	Yes	Yes	Yes
Direct assignment in trigger	No	Yes	Yes	Yes
Subprogram called from in a robot program by a trigger or interrupt	No	Yes	Yes	Yes
Submit program	No	Yes	Yes	Yes
Subprogram called from in a Submit program by an interrupt (Triggers are not possible in Submit programs.)	No	Yes	Yes	Yes
Command (e.g. variable correction)	No	No	No	Yes *

* But not while a program is running or interrupted.

6.21.3 Other examples

6.21.3.1 Robot program example: setting A1 to “soft” in one direction

Description

The robot must grip a workpiece in a press that is then ejected by the press. In order for the robot to be able to yield and absorb the ejector stroke, A1 is set to “soft” in the ejection direction.

Program

```
   ...
1  $TORQUE_AXIS = 'B000001'
2  $TORQ_VEL[1] = 130
3  $CURR_RED[1,1] = 0
4  PTP $AXIS_ACT
5  $OUT[17] = TRUE
6  WAIT FOR $IN[17]
7  $TORQUE_AXIS = 0
8  PTP {A1 -20}
    ...
```

Line	Description
1	Activate torque mode for A1.
2	Set the desired limit for velocity monitoring.
3	Set the positive current limit to 0.
4	Execute a “motion” to the current position to release the brakes. A1 can now be moved in the positive direction by the ejector of the press.
5	Signal to the press controller that the axis is ready for the ejection.
6	Wait for signal from press controller that the workpiece has been ejected.
7	Deactivate torque mode again.
8	Move to the next position.

6.21.3.2 Robot program example: E1 builds up pressure

Description

The electric motor-driven spot welding gun (external axis E1) is to exert a defined pressure on the workpiece.

A position behind the workpiece has been defined as the end point of the motion; in reality, this position cannot be reached. This causes the gun electrode to touch the workpiece and the desired pressure to be built up. As soon as the pressure has been built up, the command position changes, but the actual position remains virtually constant.

Only the positive current limit is reduced. The negative current limit remains unchanged. This makes it possible to move away from the workpiece again at full velocity.

Program

```
   ...
1  PTP {E1 -4}
2  $TORQUE_AXIS = 'B1000000'
3  $TORQ_VEL[1] = 130
4  $CURR_RED[7,1] = 20
5  PTP {E1 5}
6  WAIT FOR END_OF_WELD
7  $TORQUE_AXIS = 0
8  PTP {E1 -20}
   ...
```

Line	Description
1	Move the gun to a position just short of contact with the workpiece.
2	Activate torque mode for E1.
3	Set the desired limit for velocity monitoring.
4	Set the required current limit for E1. (Positive torque -> push.)
5	Motion towards end point behind workpiece. The pressure on the workpiece builds up.
6	Wait for end of weld signal.
7	Deactivate torque mode again.
8	The weld gun moves away from the workpiece again.

6.21.3.3 Robot program example: E1 builds up pressure (with trigger subprogram)

Description

The electric motor-driven spot welding gun (external axis E1) is to exert a defined pressure on the workpiece.

A position behind the workpiece has been defined as the end point of the motion; in reality, this position cannot be reached (P5). This causes the gun electrode to touch the workpiece and the desired pressure to be built up. As soon as the pressure has been built up, the command position changes, but the actual position remains virtually constant.

Only the positive current limit is reduced (in 2 steps here). The negative current limit remains unchanged. This makes it possible to move away from the workpiece again at full velocity.

Program

Main program:

```
   ...
1  PTP P1 C_PTP
2  TRIGGER WHEN DISTANCE=0 DELAY=0 DO TORQUE_ON() PRIO=-1
3  PTP P2 C_PTP
4  $CURR_RED[7,1]=CURR_VALUE_1
5  PTP P3 C_PTP
6  TRIGGER WHEN DISTANCE=1 DELAY=0 DO $CURR_RED[7,1]=CURR_VALUE_2
7  PTP P4 C_PTP
8  PTP P5
9  $TORQUE_AXIS=0
   ...
```

Subprogram:

```
1  DEF TORQUE_ON()
2  $TORQUE_AXIS = 64
3  END
```

Line	Main program description
1	The motion to P1 is still without torque mode. The current limit for E1 is 100%.
2	Torque mode is activated by means of a trigger subprogram. (The current limit initially remains set to 100%.)
3	Torque mode remains activated for this and subsequent motions, as all motions are approximated. **Note:** $TORQUE_AXIS set by a trigger subprogram or an interrupt only remains valid until the next exact positioning point. (The information from the motion block before $TORQUE_AXIS then becomes valid again.) $TORQUE_AXIS of this kind is also effective from the advance run onwards. (>>> 6.21.3.6 "Negative example: robot program with trigger subprogram" Page 181)
4	The positive current limit of E1 is set to the value of CURR_VALUE_1. This value remains valid until the trigger in line 6 is triggered. The assignment to $CURR_RED does not trigger an advance run stop.
6	The positive current limit of E1 is set to the value of CURR_VALUE_2. This value remains valid until torque mode is deactivated in line 9.
7	Move the gun to a position just short of contact with the workpiece.
8	Motion towards end point P5 behind the workpiece. The pressure on the workpiece builds up.
9	Deactivate torque mode when P5 has been reached. (Only the command position of the gun reaches P5.) On deactivation, the command position is automatically adjusted to the actual position, the current limit is set back to 100% and the monitoring functions are reactivated. The next motion in the program is planned from the actual position. The assignment to $TORQUE_AXIS triggers an advance run stop.

6.21.3.4 Robot interrupt program example

Description

Torque mode can also be used in interrupt programs. In this example, E1 is to be switched to torque mode when a certain error occurs. For this purpose, an interrupt program is used that is executed when the error occurs.

Program

Main program:

When the error ERROR_ON occurs, the interrupt program ERROR_PROG() is called.

```
...
INTERRUPT DECL 30 WHEN ERROR_ON==TRUE DO ERROR_PROG()
...
```

Interrupt program:

```
1  DEF ERROR_PROG()
2  BRAKE
3  $TORQUE_AXIS=64
4  $CURR_RED[7,1]=10
5  $CURR_RED[7,2]=30
6  PTP $AXIS_INT
7  HALT
8  ACT_POS = GET_ACT_POS()
9  PTP ACT_POS
10 HALT
11 $TORQUE_AXIS=0
12 PTP $POS_RET
13 END
```

Line	Description
2	The current motion is stopped as soon as ERROR_PROG() is executed.
3	Activate torque mode for E1.
4	Set the positive current limit of E1 to the required value, here 10%.
5	Set the negative current limit of E1 to the required value, here 30%.
6	Under certain circumstances, the robot may have moved in the interval between activation of the interrupt and BRAKE. For this reason, it is moved back to the position at which the interrupt was triggered.
7	The program is stopped. The error can now be eliminated. (Instead of HALT, other statements can be programmed according to the specific situation, e.g. waiting for a signal.) Following HALT, the Start key must be held down to resume the program.
8, 9	Before torque mode is deactivated, the command position must be adjusted to the actual position. (The comparison is not carried out automatically in this case, because torque mode is not deactivated directly in the robot program, but in the interrupt program.) The positions are compared by performing a motion to the current actual position. This can be read from the system variable $AXIS_INC ($AXIS_ACT cannot be used because it contains the current command values.) The value in $AXIS_INC is specified in increments, however, and must be converted to millimeters and degrees. This is carried out with the function GET_ACT_POS(). (In the concrete application, the user must write his own function for this.)

Line	Description
10	Wait for the end of the motion PTP ACT_POS. Deactivation of torque mode and resetting of the current limit must not be carried out during a motion.
11	Deactivate torque mode. The current limits are automatically set to 100%.
12	In order to be able to resume execution of the main program, the robot must return to the point at which it left the programmed path. ($AXIS_RET could also be used for this.) The main program is resumed. The advance run that was planned before the interrupt was triggered is retained.

If it is not necessary for the advance run to be retained on returning to the main program, RESUME can be used in the interrupt program. This has the advantage that torque mode is deactivated automatically and the command position is adjusted to the actual position.

The interrupt program then looks like this:

```
1  DEF ERROR_PROG()
2  BRAKE
3  $TORQUE_AXIS=64
4  $CURR_RED[7,1]=10
5  $CURR_RED[7,2]=30
6  PTP $AXIS_INT
7  RESUME
8  END
```

6.21.3.5 Submit program example: E1 builds up pressure

Program

```
   ...
1  IF $PRO_STATE1==#P_FREE
2    $ASYNC_AXIS='B000001'
3    $TORQUE_AXIS=64
4    $CURR_RED[7,1]=10
5    ASYPTP {E1 -10}
6    WAIT FOR $ASYNC_STATE==#IDLE
7    $TORQUE_AXIS=0
8    $ASYNC_AXIS=0
9  ENDIF
   ...
```

Line	Description
1	Ensure that no robot program is selected. Otherwise, the assignment to $TORQUE_AXIS will also apply there.
2	Switch E1 to asynchronous motion.
3	Activate torque mode for E1. This assignment applies from the following asynchronous motion onwards, i.e. from ASYPTP {E1 -10}.
4	Set the positive current limit to 10%. This assignment applies from the following asynchronous motion onwards, i.e. from ASYPTP {E1 -10}.
5	Motion towards end point {E1 -10} behind the workpiece. The pressure on the workpiece builds up.

Line	Description
6	Wait for the end of the asynchronous motion. (Only the command position of the gun reaches the end point.) **Note:** There is no advance run in Submit programs. Without the WAIT statement, torque mode would thus already be deactivated again during the motion.
7	Deactivate torque mode. On deactivation, the command position is automatically adjusted to the actual position, the current limit is set back to 100% and the monitoring functions are reactivated.
8	Switch E1 back to synchronous motion.

6.21.3.6 Negative example: robot program with trigger subprogram

Description

$TORQUE_AXIS set by a trigger subprogram or an interrupt has 2 effects:

- It is valid immediately, but only until the next exact positioning.
- Furthermore, it is applied from the advance run onward, i.e. for all motions that are planned after the assignment.

If there are already motions planned after the next exact positioning, the old value still applies to them!

This example shows how not to program.

Program

Main program:

```
    ...
 1  $ADVANCE=3
 2  $TORQUE_AXIS=64
 3  PTP {E1 30} C_PTP
 4  PTP {E1 20} C_PTP
 5  TRIGGER WHEN DISTANCE=0 DELAY=0 DO TORQUE_OFF() PRIO=-1
 6  PTP {E1 10} C_PTP
 7  PTP {E1 0}
 8  PTP {E1 -10}
 9  PTP {E1 -20}
10  PTP {E1 -30}
    ...
```

Subprogram:

```
1  DEF TORQUE_OFF()
2  $TORQUE_AXIS=0
3  END
```

Line	Main program description
2	Activate torque mode for E1.
5	Deactivate torque mode by means of a trigger subprogram.
6	The trigger is activated in this line. $TORQUE_AXIS=0 is now valid until the next exact positioning point, i.e. until line 7. It also applies to all motions that are planned from now onwards. The advance run is now in line 9.
7, 8	Torque mode is activated in this motion block. (Not desired!) $TORQUE_AXIS=0 is no longer valid from line 7 onwards because of the exact positioning. Furthermore, this motion was already planned when the trigger was activated.

Line	Main program description
8, 9	Torque mode is activated in this motion block. (Not desired!) This motion was also already planned when the trigger was activated.
10	Torque mode is deactivated for E1. The advance run pointer was in line 9 when the trigger was activated. $TORQUE_AXIS=0 is thus valid again from line 10 onwards.

6.22 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.

6.22.1 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.

   (>>> 6.22.2 "Configuring T1 and T2 Consistency, AUT and EXT Consistency" Page 182)

   (>>> 6.22.3 "Configuring Logic Consistency" Page 183)

5. Save the configuration with the **Apply** softkey.

6.22.2 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

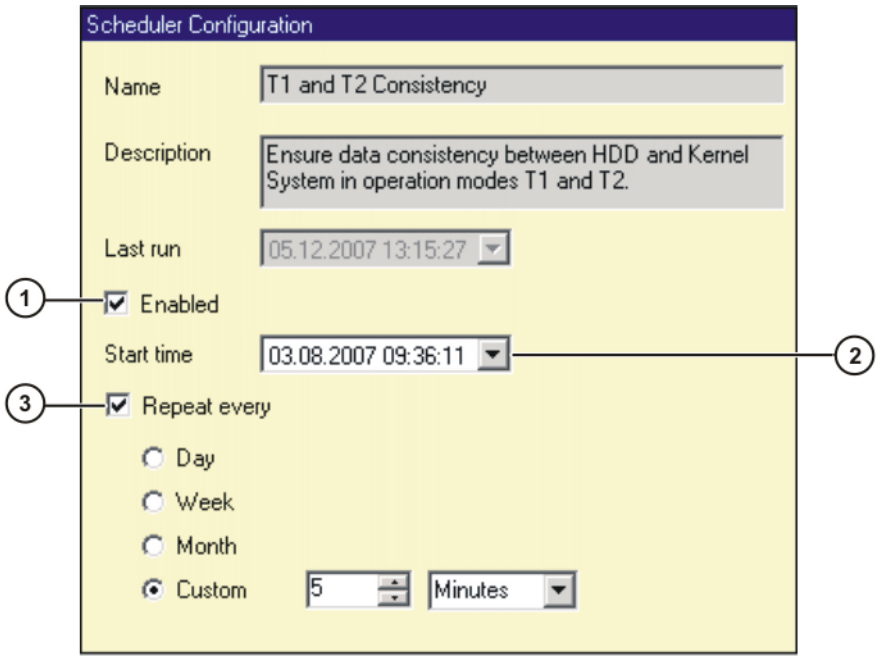


Fig. 6-24: Configuring T1 and T2 Consistency

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.

Caution!
If the intervals selected are too small, this can result in damage to the hard drive. An interval of several minutes is recommended.

6.22.3 Configuring Logic Consistency

Description The following settings are possible here:

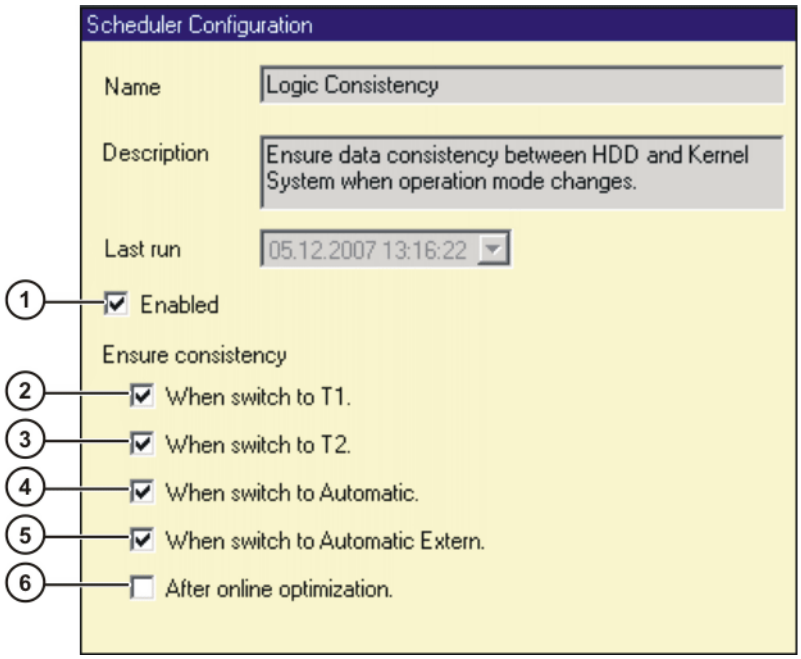


Fig. 6-25: Configuring Logic Consistency

Item	Description
1	■ Checkbox activated: data comparison is activated. ■ Check box not active: data comparison is deactivated.
2	■ Checkbox activated: data are compared when operating mode is switched to T1. ■ Check box not active: data comparison is deactivated.
3	■ Checkbox activated: data are compared when operating mode is switched to T2. ■ Check box not active: data comparison is deactivated.
4	■ Checkbox activated: data are compared when operating mode is switched to Automatic. ■ Check box not active: data comparison is deactivated.
5	■ Checkbox activated: data are compared when operating mode is switched to Automatic External. ■ Check box not active: data comparison is deactivated.
6	■ Checkbox activated: data are compared after online optimizing. ■ Check box not active: data comparison is deactivated.

6.23 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**.

6.23.1 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:

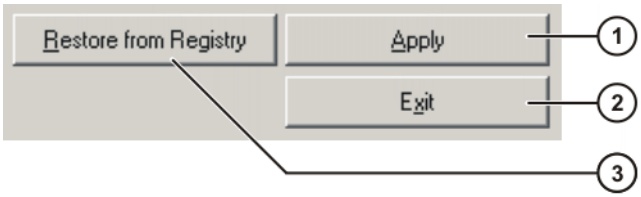


Fig. 6-26

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**.

6.23.2 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.

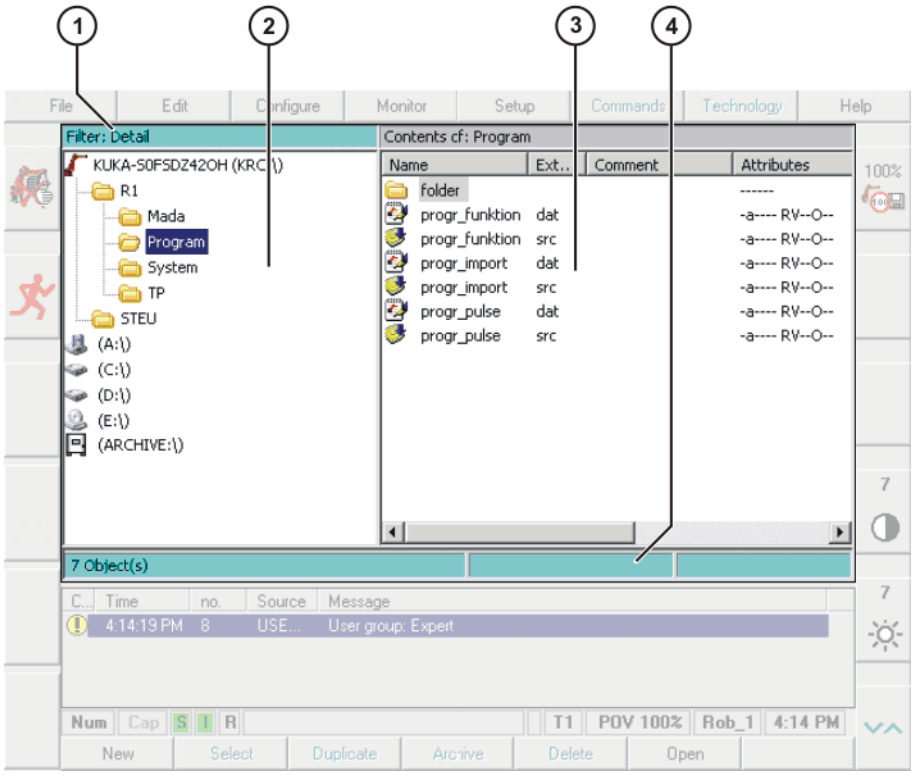


Fig. 6-27: Navigator

1	Header	3	File list
2	Directory structure	4	Status bar

Description: Display levels

The user group for which the Navigator is to be configured is selected here.

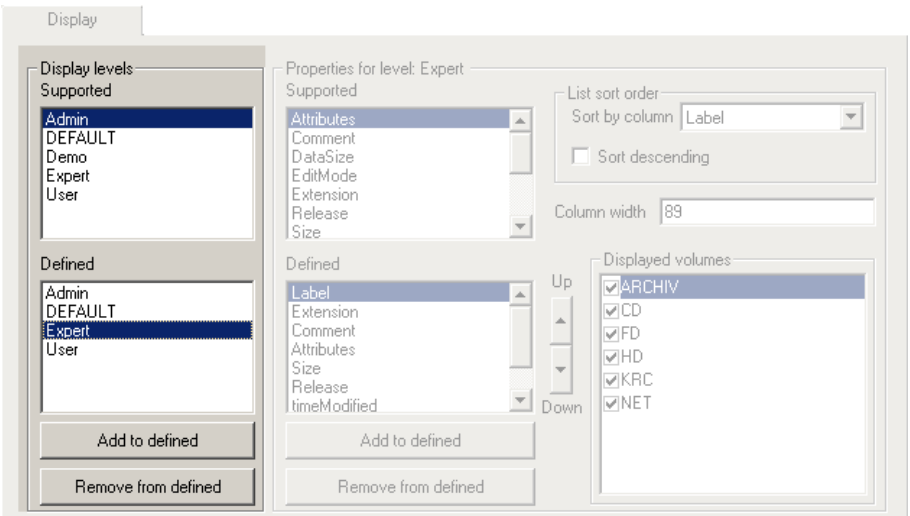


Fig. 6-28: Display tab, Display levels

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.

Element	Description
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.

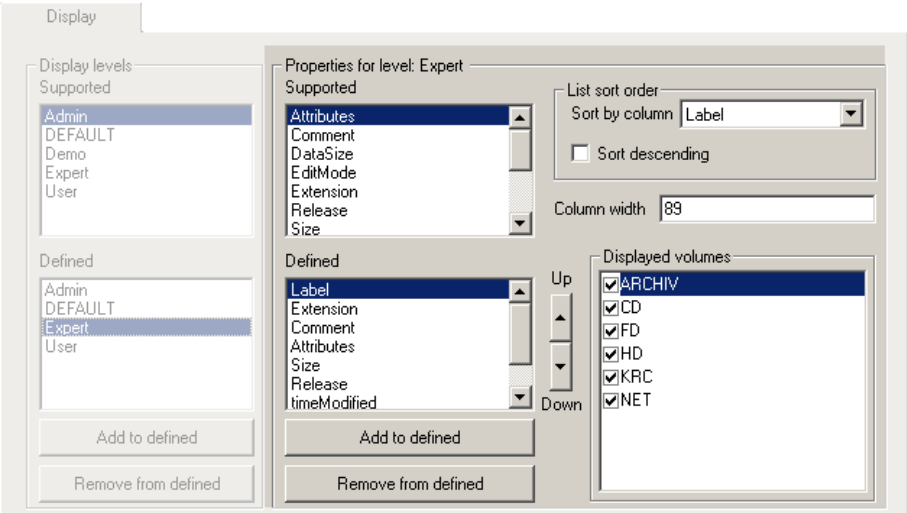


Fig. 6-29: Display tab, Properties for level

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.

6.23.3 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.

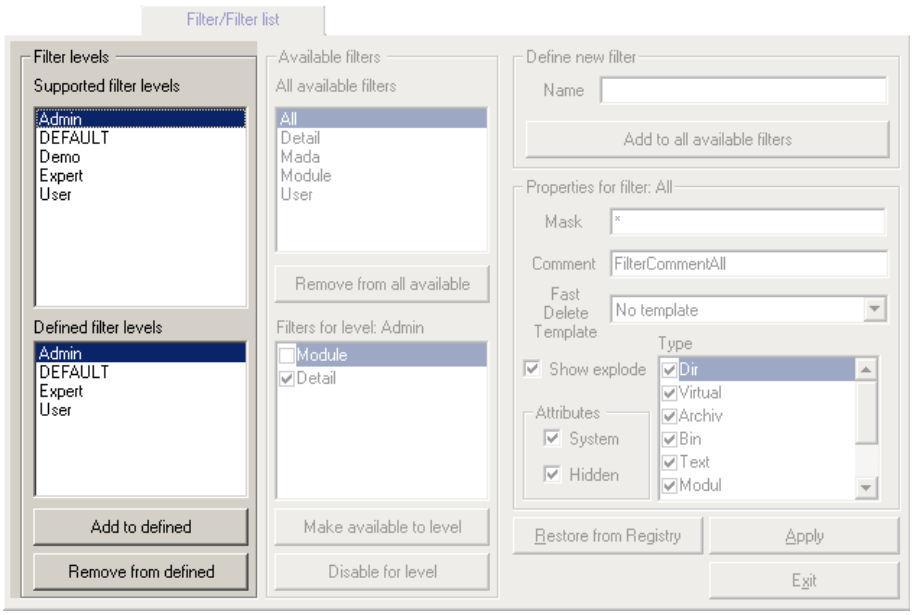


Fig. 6-30: Filter tab, Filter levels

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.

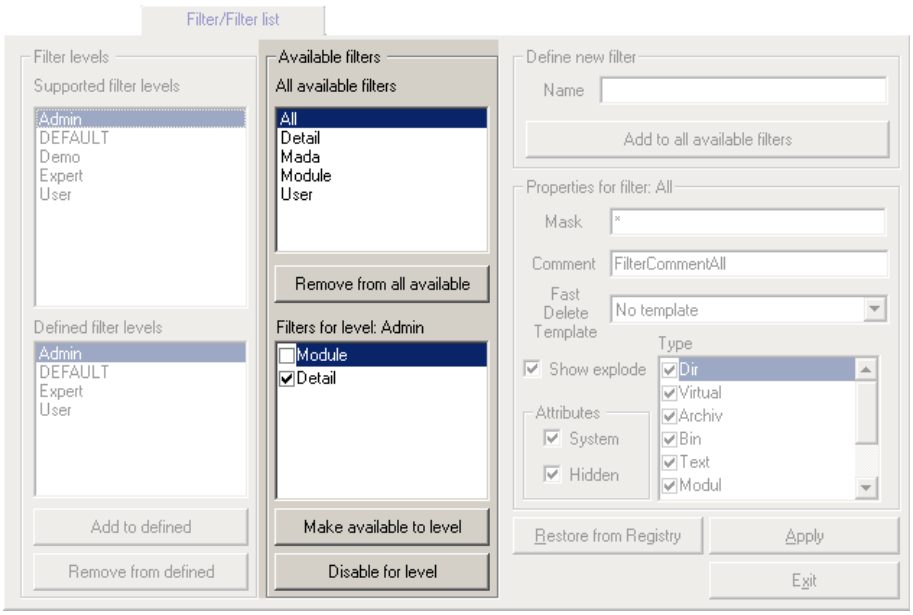


Fig. 6-31: Filter tab, Available filters

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.

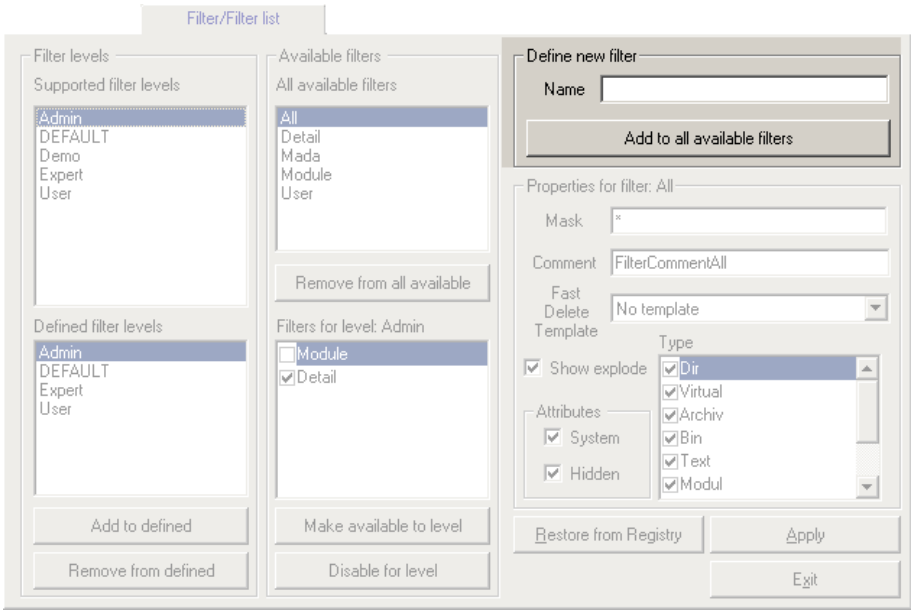


Fig. 6-32: Filter tab, Define new filter

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.

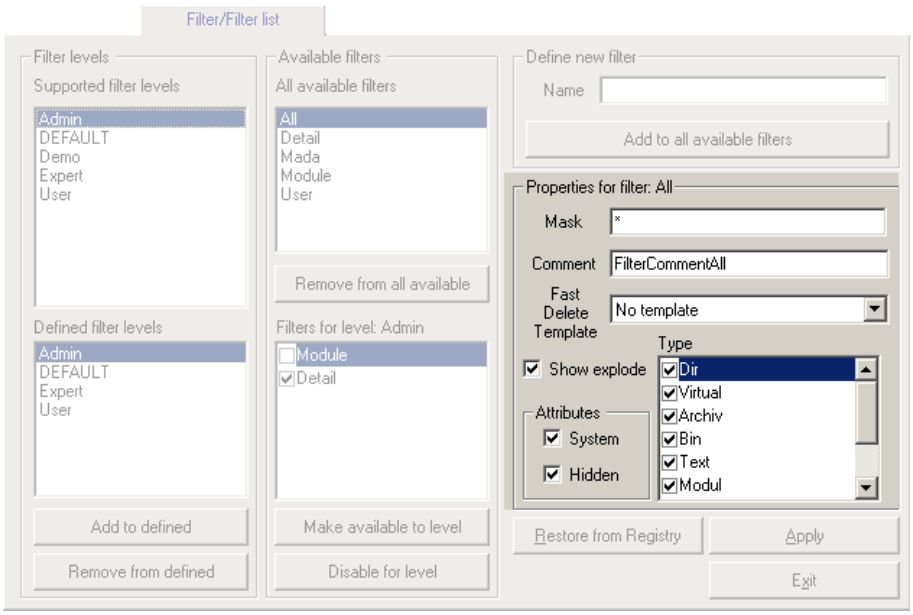


Fig. 6-33: Filter tab, Properties for filter

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

6.23.4 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.

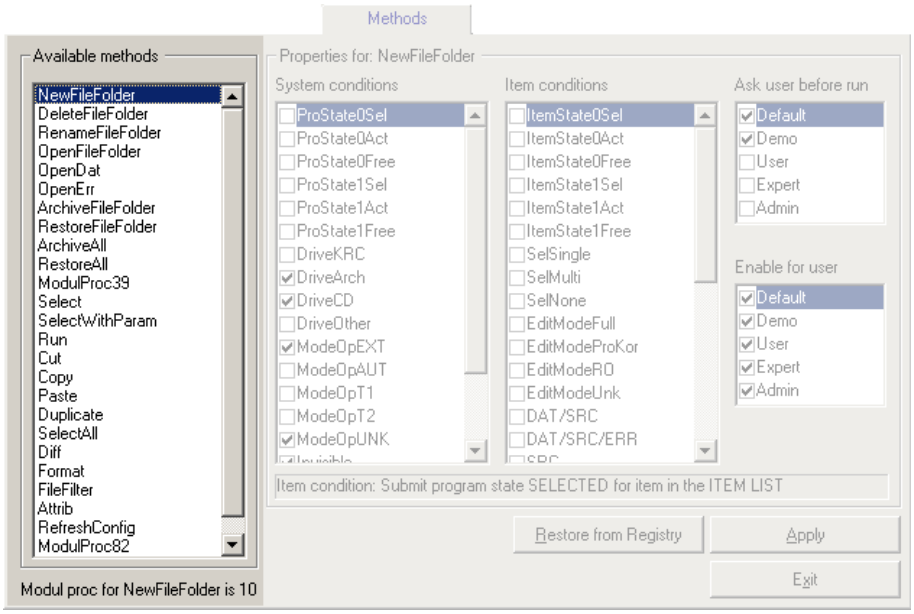


Fig. 6-34: Methods tab, Available methods

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 floppy
FileFilter	Filter
Attrib	Attribute
RefreshConfig	BOF Reinitialization
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.

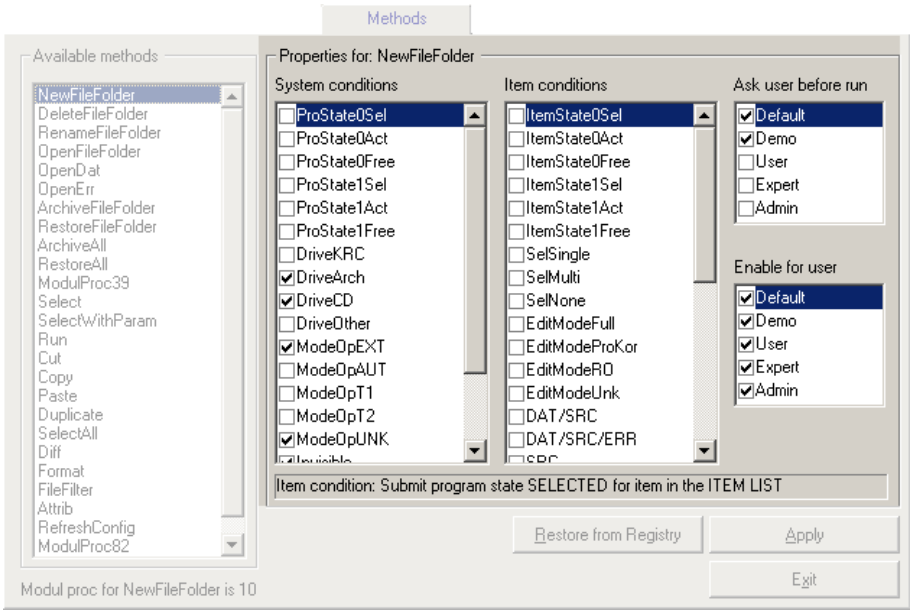


Fig. 6-35: Methods tab, Properties

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

Item conditions	Description
SRC	Object is an SRC file
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

6.23.5 User Methods tab

Overview

The layout and function of this tab are largely identical to those of the **Methods** tab (>>> 6.23.4 "Methods tab" Page 191).

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.

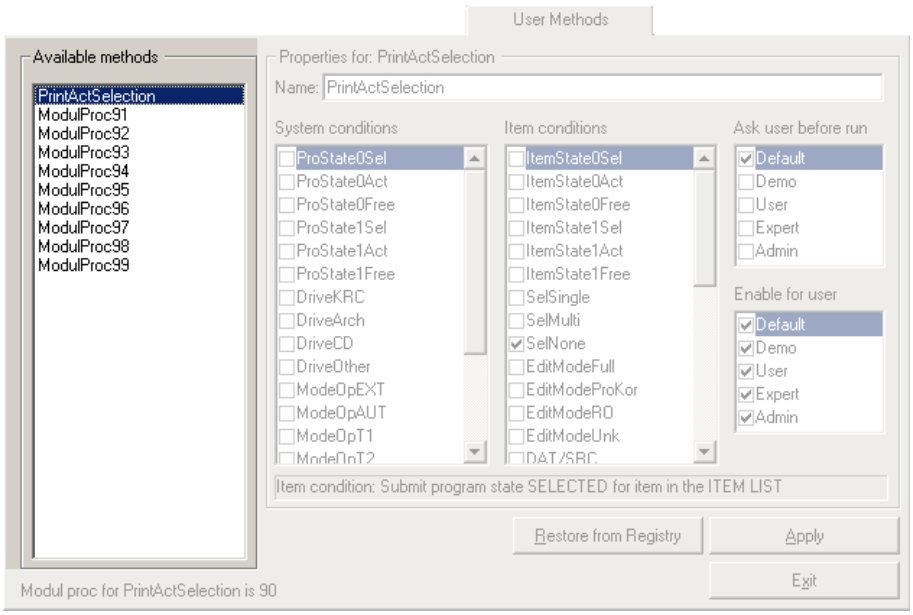


Fig. 6-36: User Methods tab, Available methods

Available methods	Description
PrintActSelection	Print the current selection.
ModulProc91	Internal identifier
ModulProc92	Internal identifier
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.

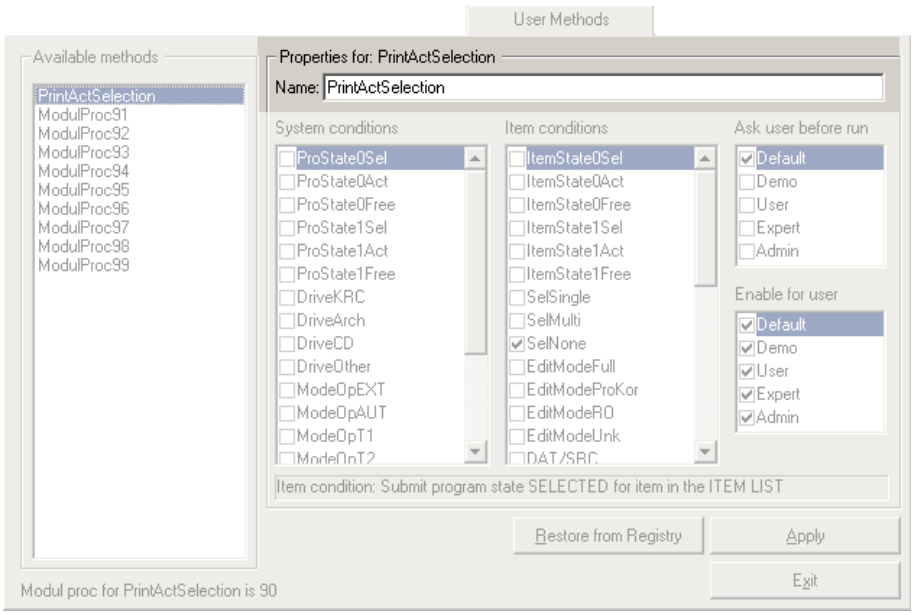


Fig. 6-37: User Methods tab, Properties

Element	Description
Properties for:	The method/Navigator command selected in **Available methods** is displayed.
Name	Designation for the method/Navigator command

6.23.6 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.

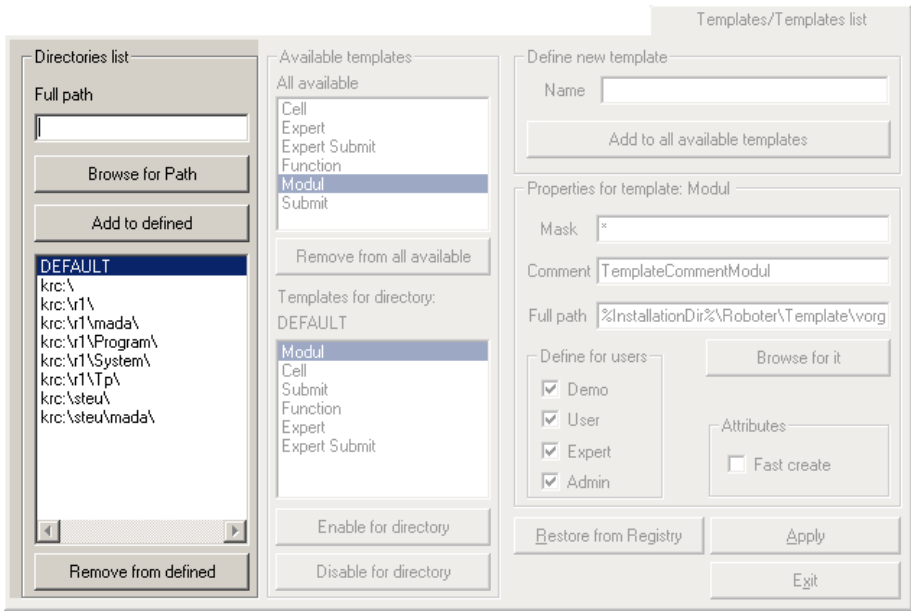


Fig. 6-38: Templates tab, Directories list

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.

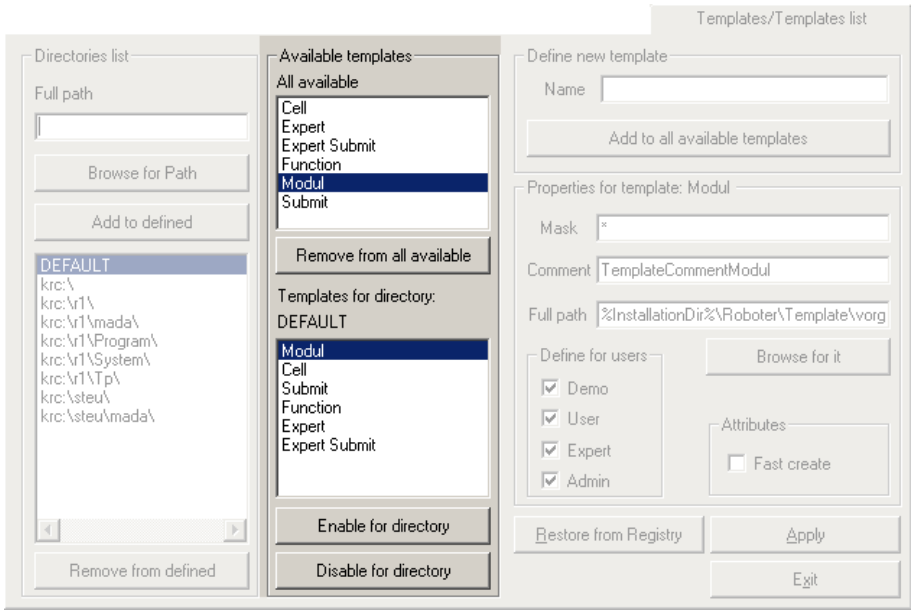


Fig. 6-39: Templates tab, Available templates

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.

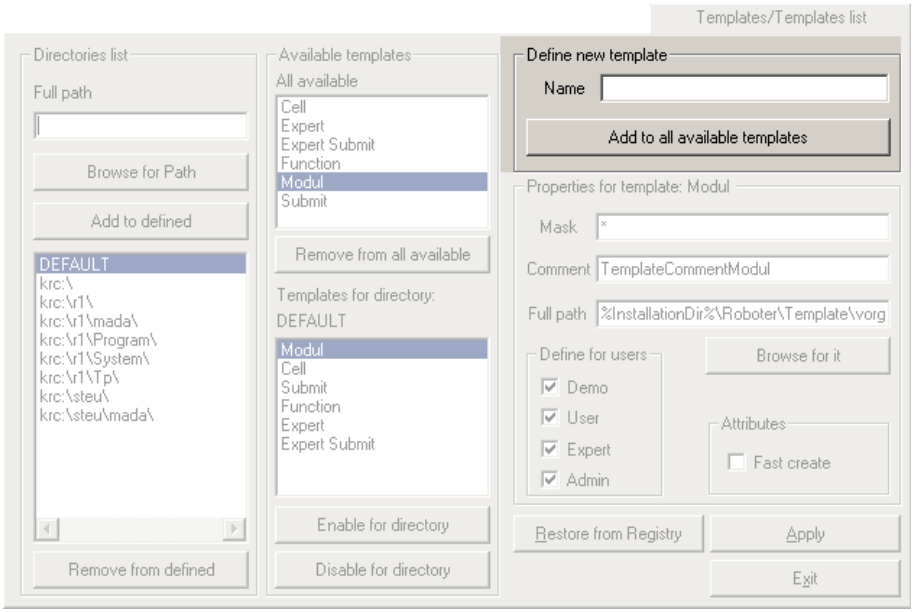


Fig. 6-40: Templates tab, Define new template

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.

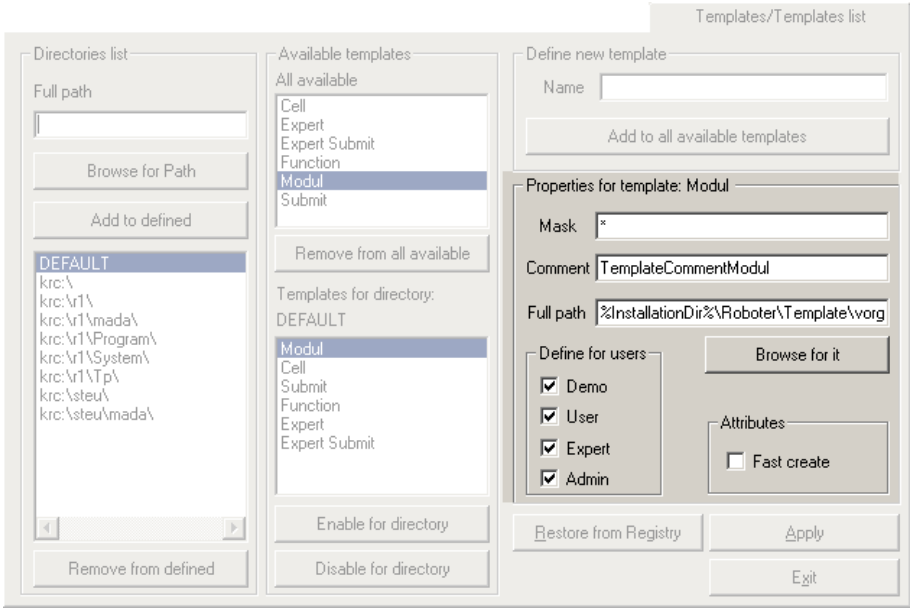


Fig. 6-41: Templates tab, Properties for template

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.

6.23.7 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

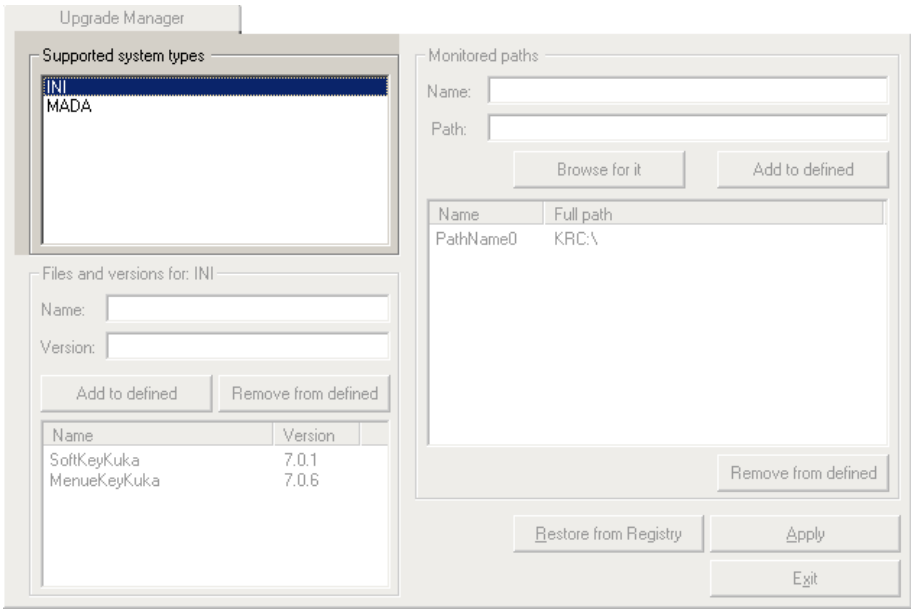


Fig. 6-42: Upgrade Manager tab, Supported system types

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.

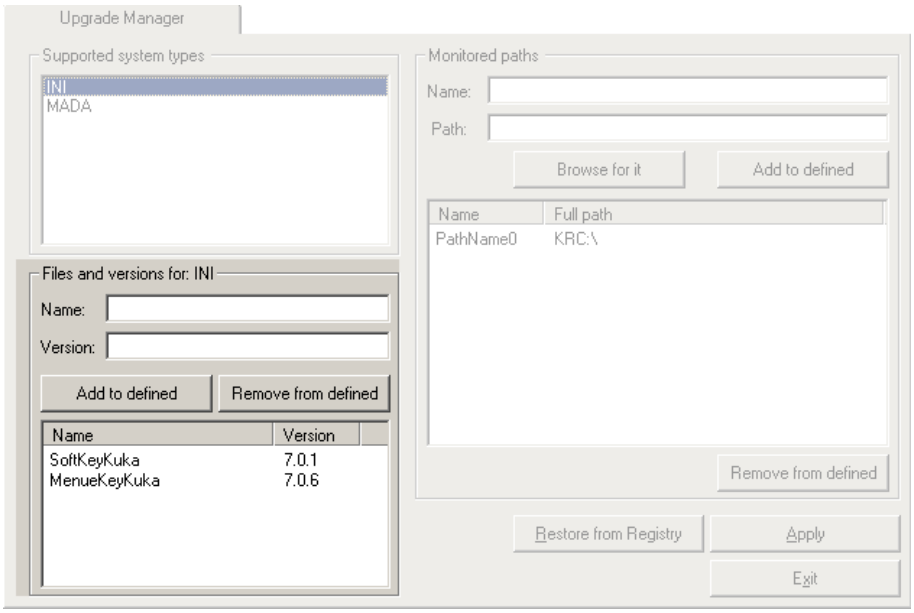


Fig. 6-43: Upgrade Manager tab, Files and versions

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.

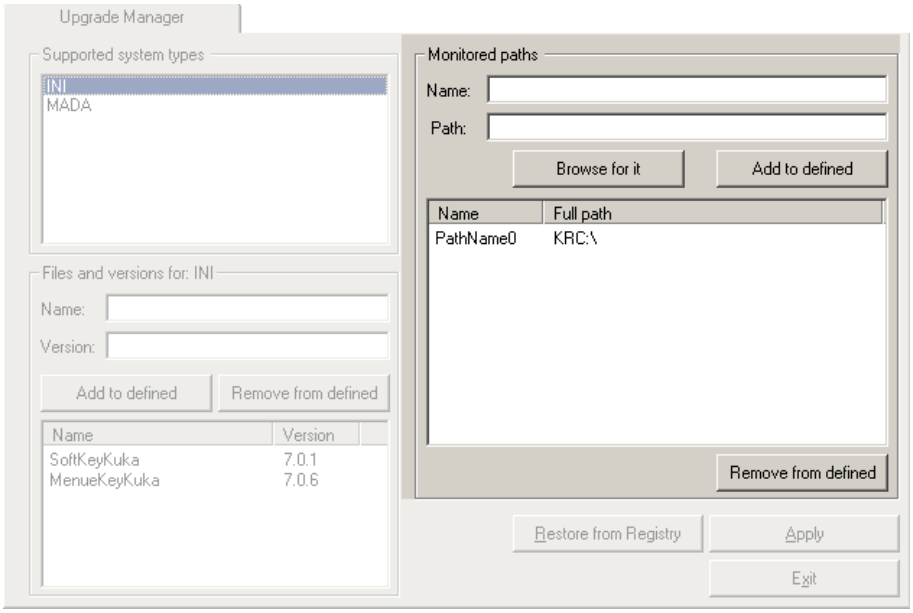


Fig. 6-44: Upgrade Manager tab, Monitored paths

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

6.23.8 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 (>>> 6.23.9 "History Info tab" Page 203).

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.

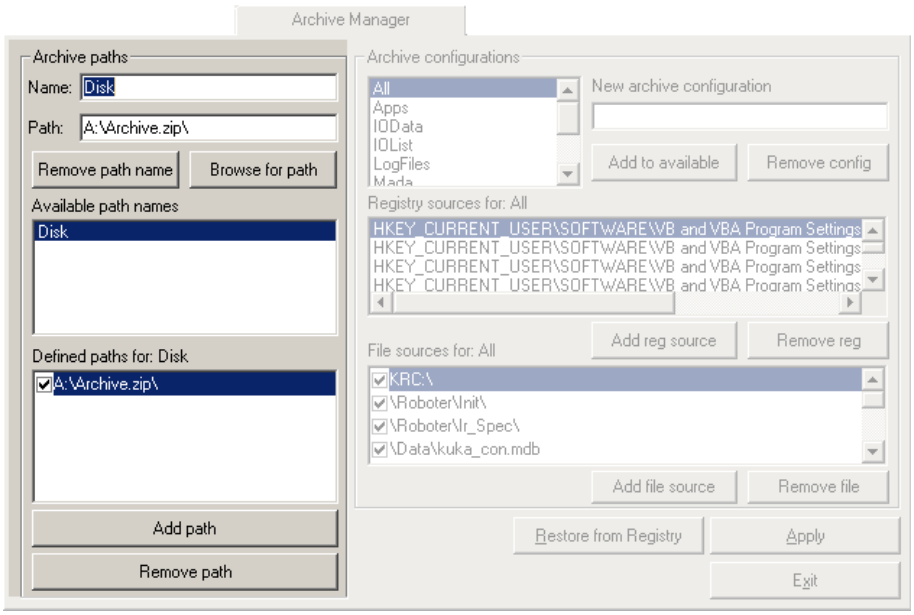


Fig. 6-45: Archive Manager tab, Archive paths

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**.

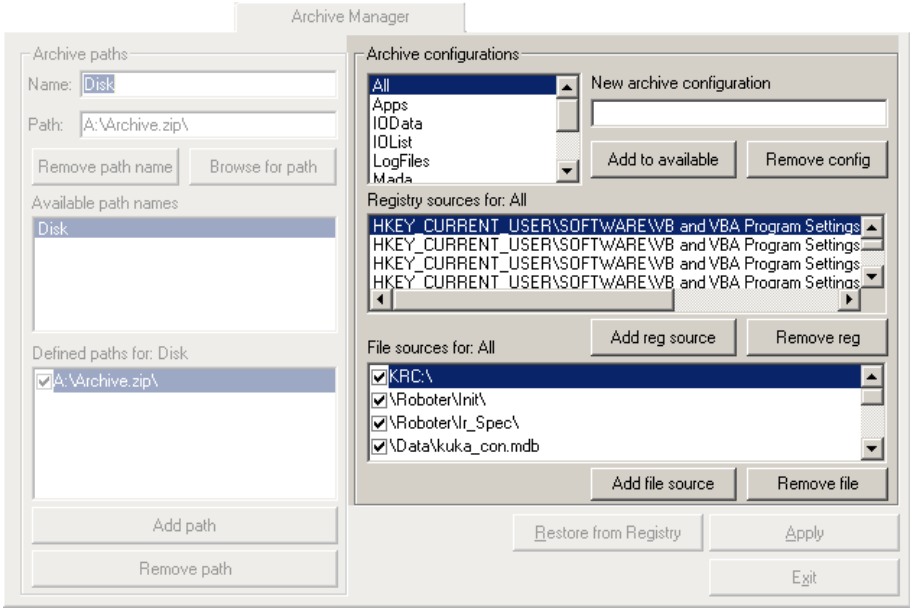


Fig. 6-46: Archive Manager tab, Archive configurations

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 in item 1.
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**.
File sources for:	Files and directories to be archived Check box active: The file or directory is restored

Element	Description
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**.

6.23.9 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

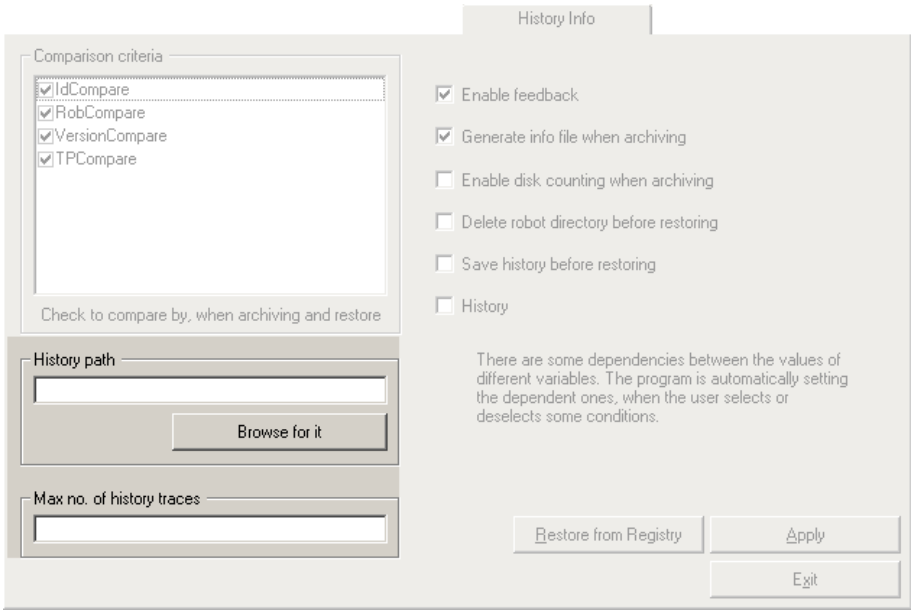


Fig. 6-47: History Info tab, History

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

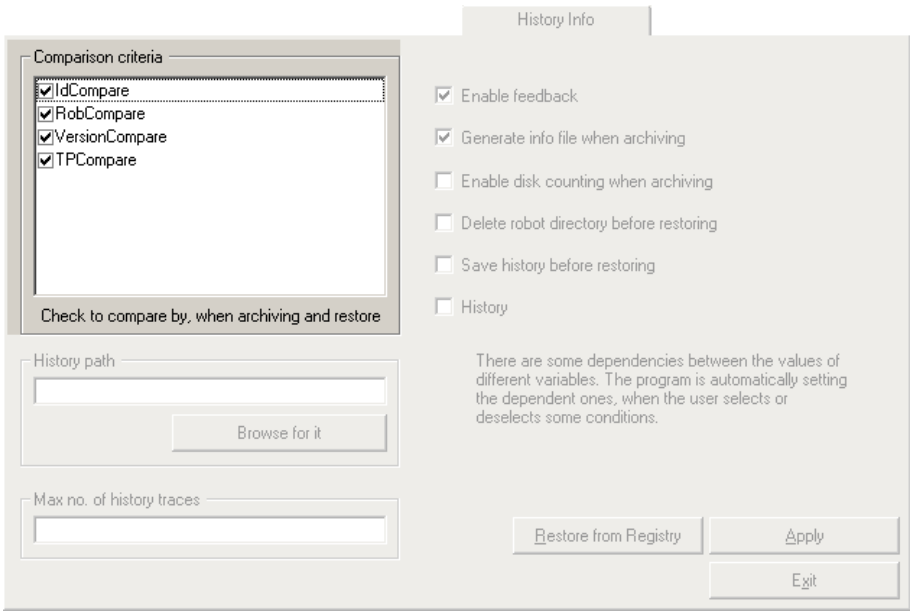


Fig. 6-48: History Info tab, Comparison criteria

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

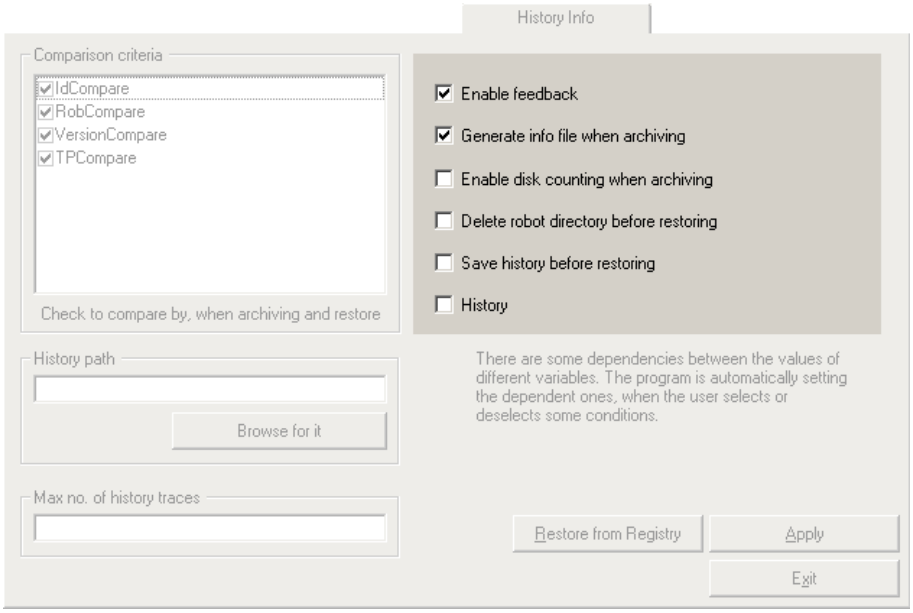


Fig. 6-49: History Info tab, further settings

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.

6.23.10 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

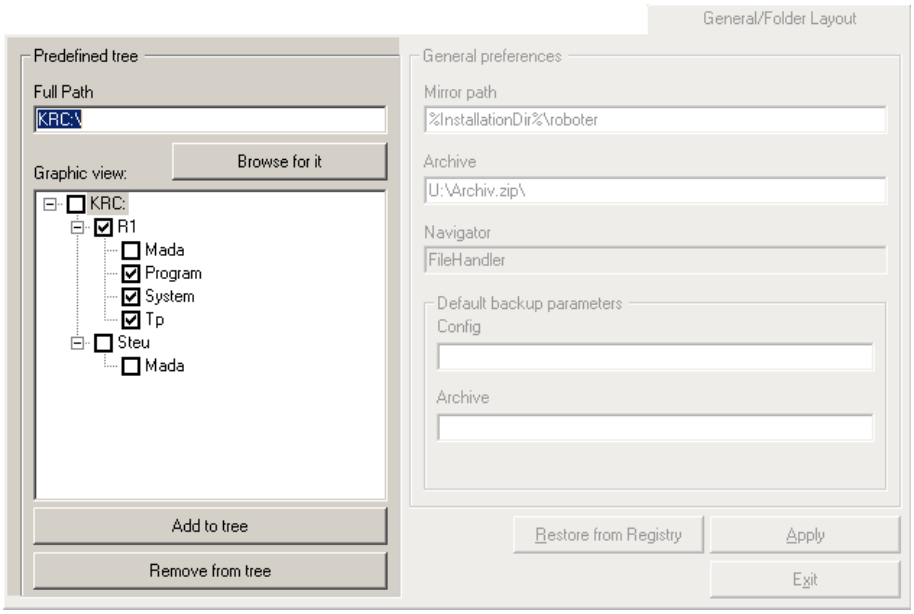


Fig. 6-50: General/Folder Layout tab, Predefined tree

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.

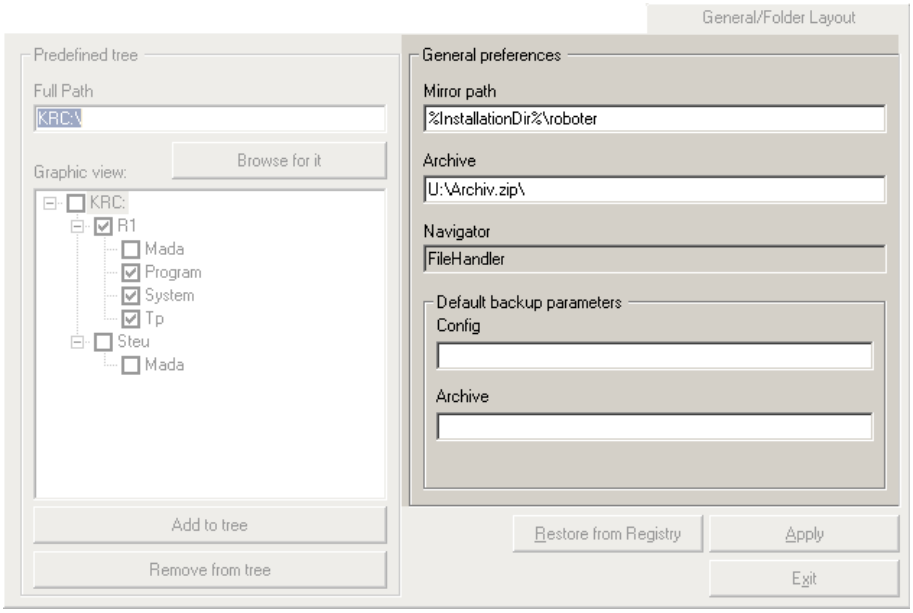


Fig. 6-51: General/Folder Layout tab, General preferences

7 Program management

7.1 Navigator file manager

Overview

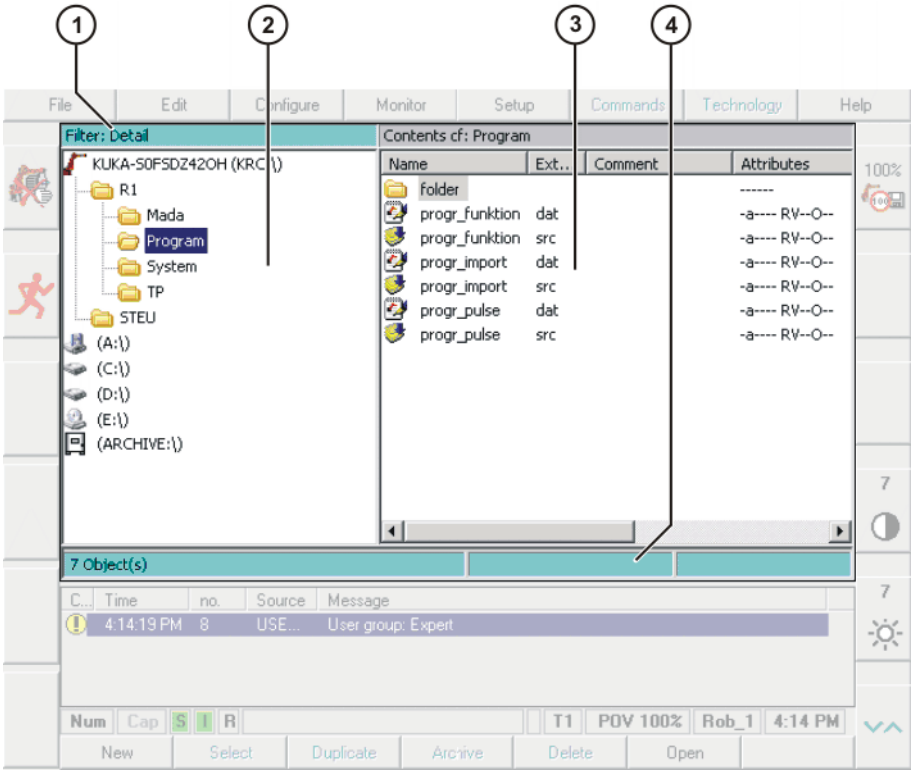


Fig. 7-1: Navigator

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.
  (>>> 7.1.1 "Selecting filters" Page 208)
- 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
Modified	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

7.1.1 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**.

7.1.2 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

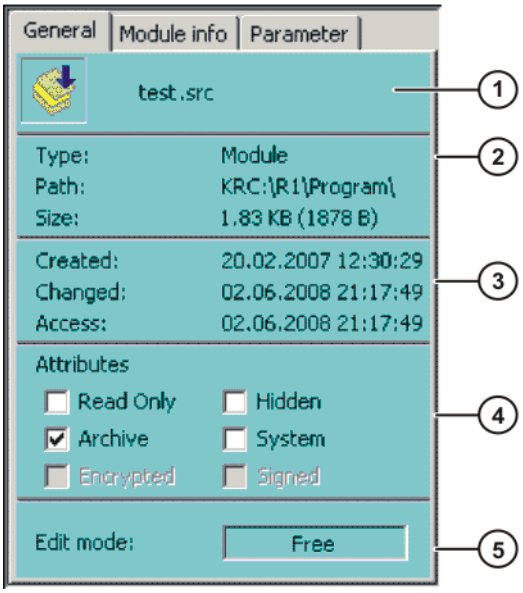


Fig. 7-2: File properties, “General” tab

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

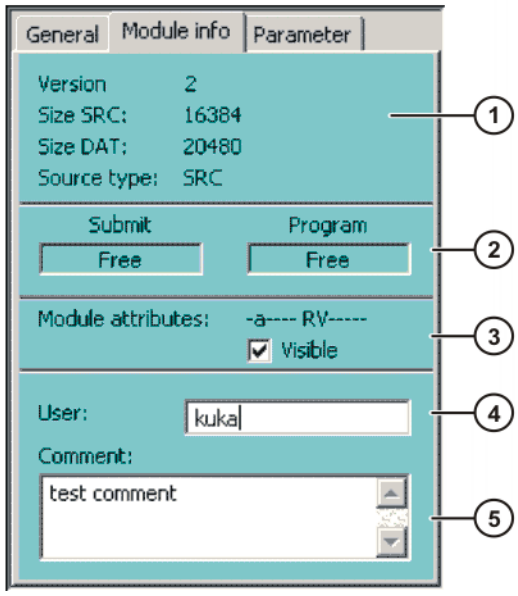


Fig. 7-3: File properties, “Module info” tab

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

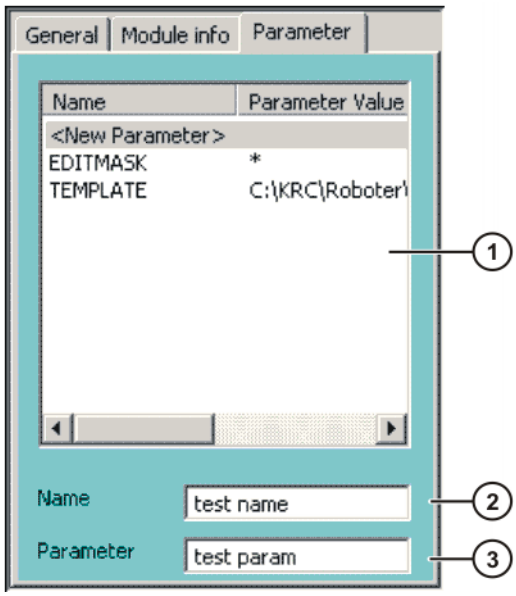


Fig. 7-4: File properties, “Parameter” tab

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

7.1.3 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
	Archive in ZIP format
	The contents of a directory are being read.
	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.

7.1.4 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**.

7.1.5 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.

7.1.6 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**.

7.1.7 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 the KUKA.Encryption software is contained in the Expert documentation “KUKA.Encryption 1.0”.

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.

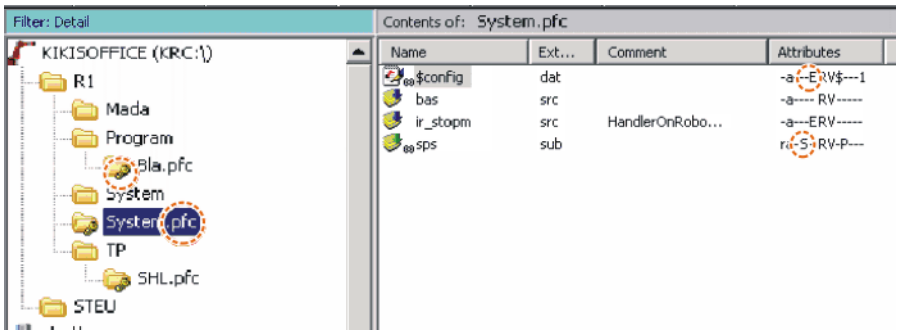


Fig. 7-5: Encrypted files in the Navigator

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.

7.2 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.

  (>>> 7.2.1 "Selecting a program" Page 215)

  (>>> 7.2.2 "Opening a program" Page 216)
- It is possible to toggle backwards and forwards between the program display and the Navigator.

  (>>> 7.2.3 "Toggling between the Navigator and the program" Page 216)
- 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.

  (>>> 7.2.4 "Selecting one program and opening another program" Page 217)

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.

7.2.1 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

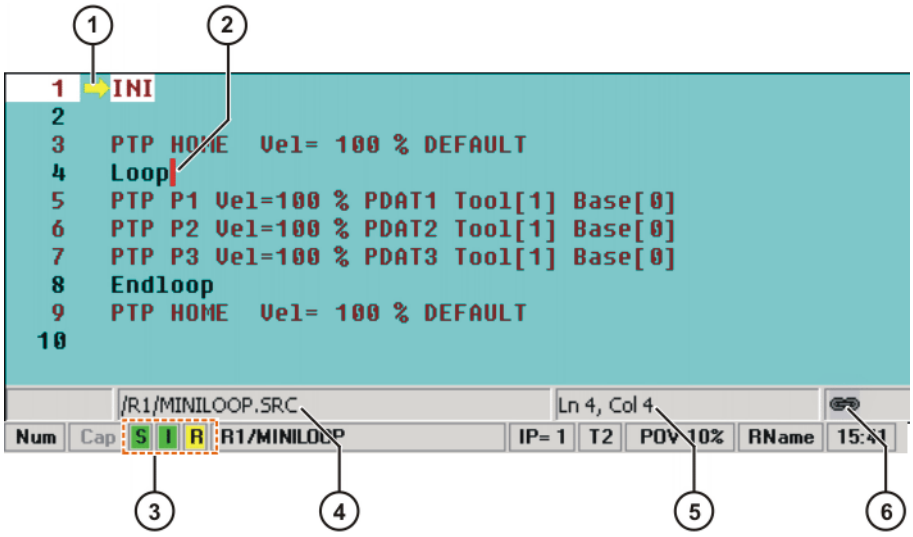


Fig. 7-6: Program is selected

1 Block pointer
2 Cursor
3 (>>> 4.2.4 "Status bar" Page 50)
4 Program path and file name

5 Position of the cursor
6 The icon indicates that the program is selected.

7.2.2 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

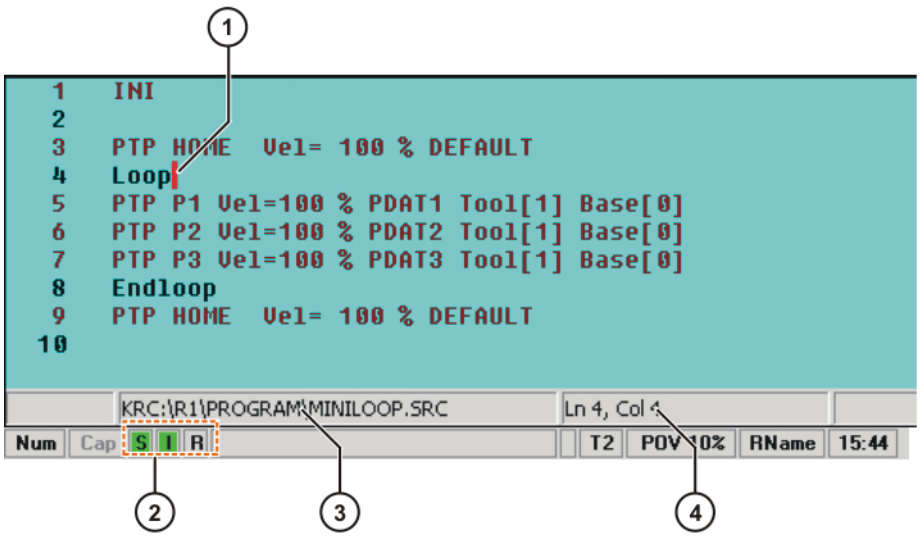


Fig. 7-7: Program is open

1 Cursor
2 (>>> 4.2.4 "Status bar" Page 50)
3 Program path and file name
4 Position of the cursor

7.2.3 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.

7.2.4 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.

7.3 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 (>>> 7.3.1 "HOME position" Page 218)
8	LIN motion (>>> 9.2.3 "Programming a LIN motion" Page 258)
14	PTP motion (>>> 9.2.1 "Programming a PTP motion" Page 257)
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.

7.3.1 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°	+ 90°	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.

7.4 Displaying/hiding program sections

7.4.1 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** > **Tools** > **Editor** > **Def-line**.
  Check mark activated in menu: DEF line is displayed.
  Check mark not activated in menu: DEF line is hidden.

7.4.2 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** > **Tools** > **Editor** > **ASCII Mode**.
  Check mark activated in menu: ASCII mode is activated.
  Check mark not activated in menu: ASCII mode is deactivated.

7.4.3 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. 7-8: 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** > **Tools** > **Editor** > **Linebreak**.
  Check mark activated in menu: line break function is activated.
  Check mark not activated in menu: line break function is deactivated.

7.4.4 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.
  (>>> 7.6.3 "Creating Folds" Page 228)

If a program is deselected, all Folds are automatically closed.

```
2
3   PTP HOME  Vel= 100 % DEFAULT
4
```

Fig. 7-9: Example of a closed Fold

```
2
3   PTP HOME  Vel= 100 % DEFAULT
4   $BWDSTART = FALSE
5   PDAT_ACT=PDEFAULT
6   FDAT_ACT=FHOME
7   BAS (#PTP_PARAMS,100 )
8   $H_POS=XHOME
9   PTP  XHOME
```

Fig. 7-10: 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.

7.5 Starting a program

7.5.1 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.

7.5.2 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.

7.5.3 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.
(>>> 7.4.3 "Activating/deactivating the line break function" Page 219)

```
25   INTERRUPT DECL 15 WHEN
   ↳ $MEAS_PULSE[TOUCH_I[TOUCH_ACTIVE].IN_NR] DO H70 (6,CD0 )
```

Fig. 7-11: 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
	The block pointer is located higher up in the program.
	The block pointer is located lower down in the program.

7.5.4 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.

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	Meaning
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.

7.5.5 Starting a program forwards (manual)

Precondition

- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Select the program run mode.
2. Hold the enabling switch down and wait until the status bar indicates  (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.

7.5.6 Starting a program forwards (automatic)

Precondition

- A program is selected.
- Operating mode Automatic (not Automatic External)

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. Program is executed.

To stop a program that has been started in Automatic mode, press the STOP key.

7.5.7 Carrying out a block selection

Description

A program can be started at any point by means of a block selection.

Precondition

- Program is selected.
- Operating mode T1 or T2.

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.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.

7.5.8 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.
(>>> 6.16 "Backward motion" Page 141)

Precondition

- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Hold the enabling switch down and wait until the status bar indicates  (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.

7.5.9 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**.

7.5.10 Starting Automatic External mode

Precondition
- Operating mode T1 or T2
- Inputs/outputs for Automatic External and the program CELL.SRC are configured.

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).

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.

To stop a program that has been started in Automatic mode, press the STOP key.

7.6 Editing a program

Overview
- A running program cannot be edited.
- Programs cannot be edited in AUT EXT mode.

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.

Action	Possible in user group …?
Insert comment or stamp	**User**: Yes **Expert**: Yes
Delete lines	**User**: Yes **Expert**: Yes
Create folds	**User**: No **Expert**: Yes
Copy	**User**: No **Expert**: Yes
Paste	**User**: No **Expert**: Yes
Insert blank lines (press the Enter key)	**User**: No **Expert**: Yes
Cut	**User**: No **Expert**: Yes
Find	**User**: Yes **Expert**: Yes Possible for all user groups in an open program, even in AUT EXT mode.
Replace	**User**: No **Expert**: Yes (program is open, not selected)
Programming with inline forms	**User**: Yes **Expert**: Yes
KRL programming	**User**: Possible to a certain extent. KRL instructions covering several lines (e.g. LOOP … ENDLOOP) are not permissible. **Expert**: Yes

7.6.1 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

Fig. 7-12: 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.

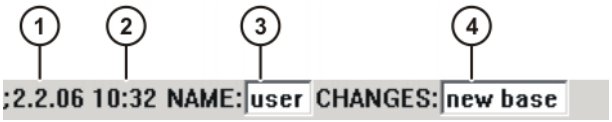


Fig. 7-13: Inline form "Stamp"

Item	Description
1	Current system date (cannot be edited)
2	Current system time
3	Name or ID of the user
4	Any text

7.6.2 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 **Program** > **Delete**.
3. Confirm the request for confirmation with **Yes**.

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.

Lines cannot be restored once they have been deleted!

7.6.3 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.

```
 4
 5   ;;FOLD outputs
 6   $OUT[1] = TRUE
 7   $OUT[2] = TRUE
 8   $OUT[3] = TRUE
 9   ;;ENDFOLD outputs
10
```

Fig. 7-14: Creating a sample Fold, step 1

2. Delete the second semicolon.

```
 4
 5   ;FOLD outputs
 6   $OUT[1] = TRUE
 7   $OUT[2] = TRUE
 8   $OUT[3] = TRUE
 9   ;ENDFOLD outputs
10
```

Fig. 7-15: Creating a sample Fold, step 2

3. Position the cursor in a line outside the Fold. The Fold closes.

```
 4
 5   outputs
 6
```

Fig. 7-16: Creating a sample Fold, step 3

7.6.4 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”

7.7 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**.

7.8 Archiving

7.8.1 Archiving data

Description

Depending on the robot controller used, different paths can be selected. Which path the robot controller archives data to depends on the configuration in the KRC Configurator.

(>>> 6.23.8 "Archive Manager tab" Page 200)

Path	Robot controller
KUKA USB stick	Only KR C2 edition2005
Network path	All robot controllers
Floppy disk	Only robot controllers that have a floppy disk drive

Caution!
The only USB stick that may be used is the KUKA USB stick. Data may be lost or modified if any other USB stick is used.

Information about the KRC Configurator is contained in the “Operating and Programming Instructions for System Integrators”.

Precondition

For all paths:

- The desired path is configured in the KRC Configurator.

For archiving to floppy:

- Disk has been formatted (if required).
  (>>> 7.8.3 "Formatting the floppy disk" Page 230)
- Disk is in disk drive.

More than one floppy disk will be required if large quantities of data are involved!

For archiving to a KUKA USB stick:

- The stick is connected.

Procedure

1. Select the menu sequence **File** > **Archive** and the desired menu item.
   (>>> 7.8.2 "Menu item “Archive”" Page 230)
2. Confirm the request for confirmation with **Yes**.
3. Only in the case of archiving to a floppy disk: a message indicates when another disk is required. Insert new disk.

4. A message indicates completion of the archiving process. The file ARCHIVE.ZIP is generated by default.

Caution!
When archiving to the KUKA USB stick: the stick must not be removed until the LED on the stick is no longer lit. Otherwise, the stick could be damaged.

7.8.2 Menu item “Archive”

Description

The following menu items are available for archiving. Exactly which files are archived depends on the configuration in the KRC Configurator.

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”.
Configuration > KUKA TechPack	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.

7.8.3 Formatting the floppy disk

Description

This procedure can be used to format floppy disks in the case of robot controllers that have a floppy disk drive.

Procedure

1. Insert floppy disk into the drive.
2. Select the menu sequence **File** > **Format floppy disk**.
3. Confirm the request for confirmation with **Yes**. A message indicates completion of the formatting process.

Caution!
Do not remove the floppy disk from the drive until the LED on the drive is no longer lit. Otherwise the disk drive and/or the floppy disk could suffer damage.

7.8.4 Restoring data

Overview

Caution!

Only KSS 5.5 archives may be loaded into KSS 5.5. If other archives are loaded, the following may occur:

- Error messages
- Robot controller is not operable.
- Personal injury and damage to property.

The robot controller cannot detect whether multiple floppy disks were used for archiving. For this reason, the user is not prompted, after the first floppy disk, to insert additional disks, but must do so without prompting. The order in which the disks are inserted is irrelevant.

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.

7.8.4.1 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 data have been archived to floppy disk, they are also restored from the floppy disk.

Precondition

- If data were archived to floppy: Disk is in disk drive.
- If data were archived to KUKA USB stick: The stick is connected.

Procedure

1. Select the menu sequence **File** > **Restore** and the desired menu item.
2. Confirm the request for confirmation with **Yes**.
   A message indicates completion of the restoration process.
3. Only if restoring data from floppy: if data have been archived on more than one disk, insert the next floppy disk and repeat steps 1 and 2.
4. Only if restoring data from floppy: remove floppy from drive.

Caution!
When restoring data from the KUKA USB stick: the stick must not be removed until the LED on the stick is no longer lit. Otherwise, the stick could be damaged.

5. Reboot the robot controller.

7.8.4.2 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

- The path from which data are to be restored must be configured in the KRC Configurator.
- If data are to be restored from floppy: Disk is in disk drive.
- If data are to be restored from the USB stick: The stick is connected.

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.
4. Confirm the request for confirmation with **Yes**. A message indicates completion of the restoration process.
5. Only if restoring data from floppy: if data have been archived on more than one disk, insert the next floppy disk and repeat steps 1 to 4.
6. Only if restoring data from floppy: remove floppy from floppy drive.

Caution!
When restoring data from the KUKA USB stick: the stick must not be removed until the LED on the stick is no longer lit. Otherwise, the stick could be damaged.

7. Reboot the robot controller.

The **Restr. All** softkey ignores which objects are selected in the Navigator and accesses the path that is configured for archiving.

8 Basic principles of motion programming

8.1 Overview of motion types

The following motion types can be programmed:

- Point-to-point motions (PTP)
  (>>> 8.2 "Motion type PTP" Page 233)
- Linear motions (LIN)
  (>>> 8.3 "Motion type LIN" Page 233)
- Circular motions (CIRC)
  (>>> 8.4 "Motion type CIRC" Page 234)
- Spline motions
  (>>> 8.7 "Motion type “Spline”" Page 239)

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.

8.2 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.

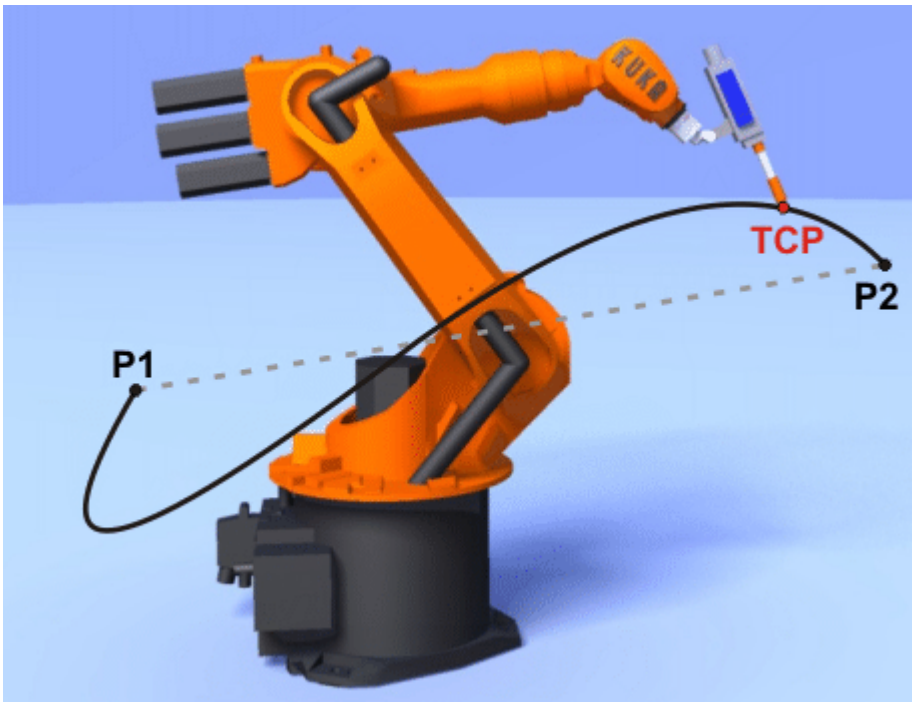


Fig. 8-1: PTP motion

8.3 Motion type LIN

The robot guides the TCP at a defined velocity along a straight path to the end point.

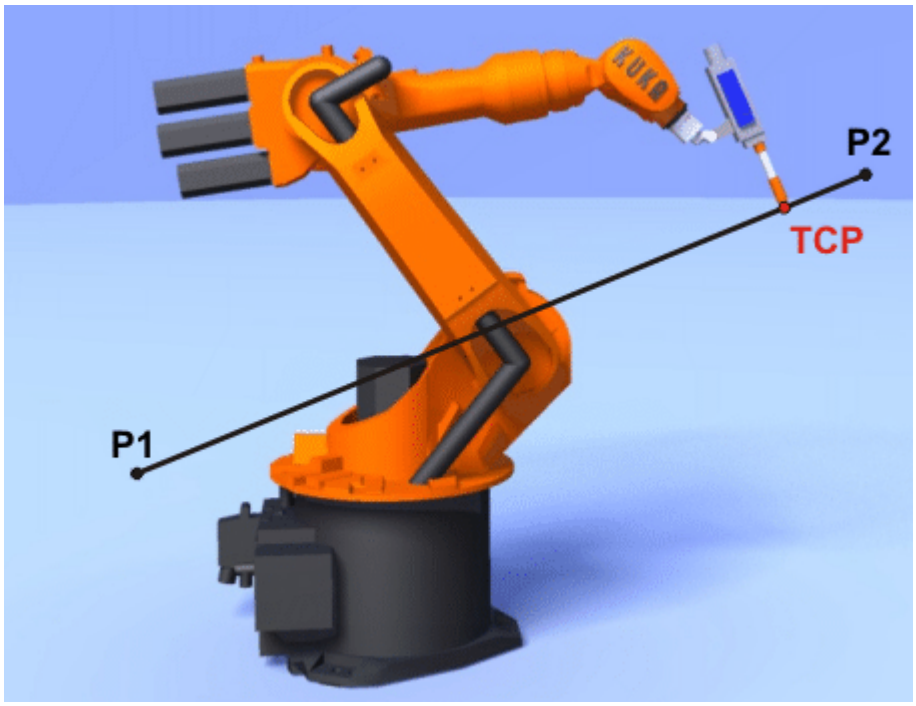


Fig. 8-2: LIN motion

8.4 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.

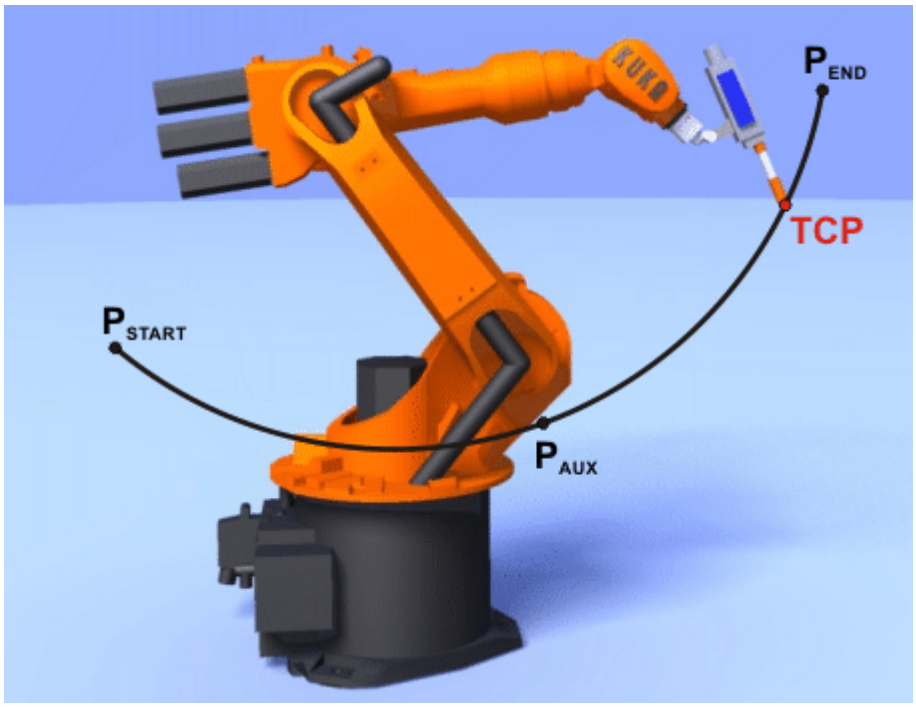


Fig. 8-3: CIRC motion

8.5 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.

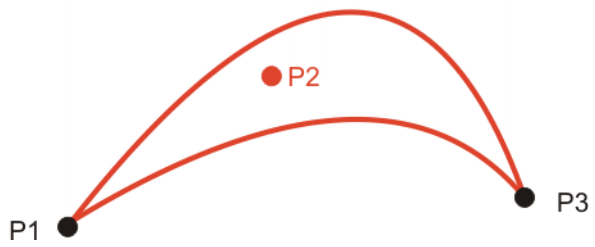


Fig. 8-4: PTP motion, P2 is approximated

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.

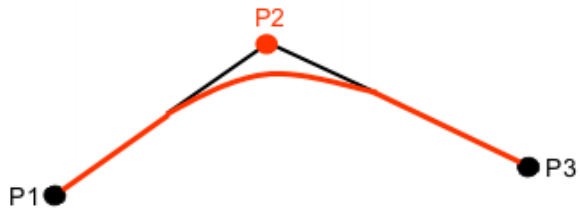


Fig. 8-5: LIN motion, P2 is approximated

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.

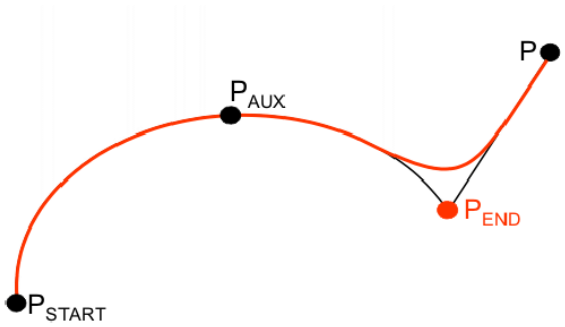


Fig. 8-6: CIRC motion, P_{END} is approximated

8.6 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**
  (>>> 9.2.9 "Option window “Motion parameter” (LIN, CIRC)" Page 263)
- 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. **Note**: If, with **Standard**, the robot passes through a wrist axis singularity, use **Wrist PTP** instead.
■ Option window: **Wrist PTP** ■ $ORI_TYPE = #JOINT	The orientation of the TCP changes continuously during the motion. This is done by linear transformation (axis-specific motion) of the wrist axis angles. **Note**: Use **Wrist PTP** if, with **Standard**, the robot passes through a wrist axis singularity. The orientation of the TCP changes continuously during the motion, but not uniformly. **Wrist PTP** is thus not suitable if a specific orientation must be maintained exactly, e.g. in the case of laser welding.

If a wrist axis singularity occurs with **Standard** and the desired orientation cannot be maintained exactly enough with **Wrist PTP**, 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**.

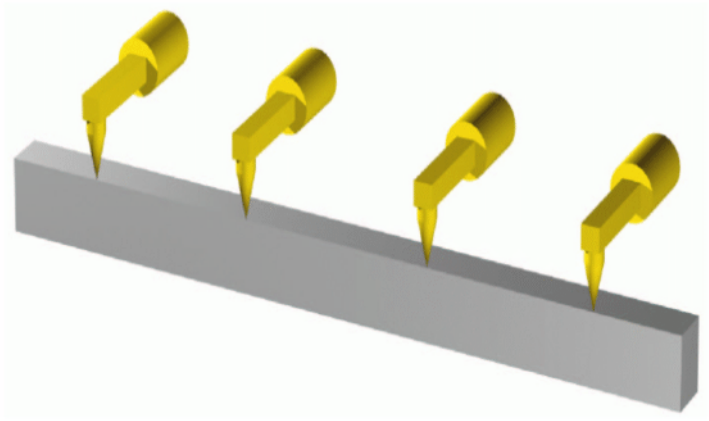


Fig. 8-7: Orientation control - Constant

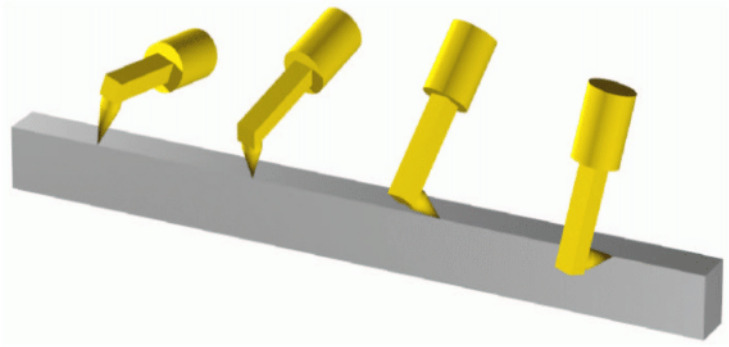


Fig. 8-8: Standard or Wrist PTP

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

$CIRC_TYPE is meaningless if $ORI_TYPE = #JOINT.

(>>> 8.6.1 "Combinations of $ORI_TYPE and $CIRC_TYPE" Page 237)

8.6.1 Combinations of $ORI_TYPE and $CIRC_TYPE

$ORI_TYPE = #CONSTANT, $CIRC_TYPE = #PATH:

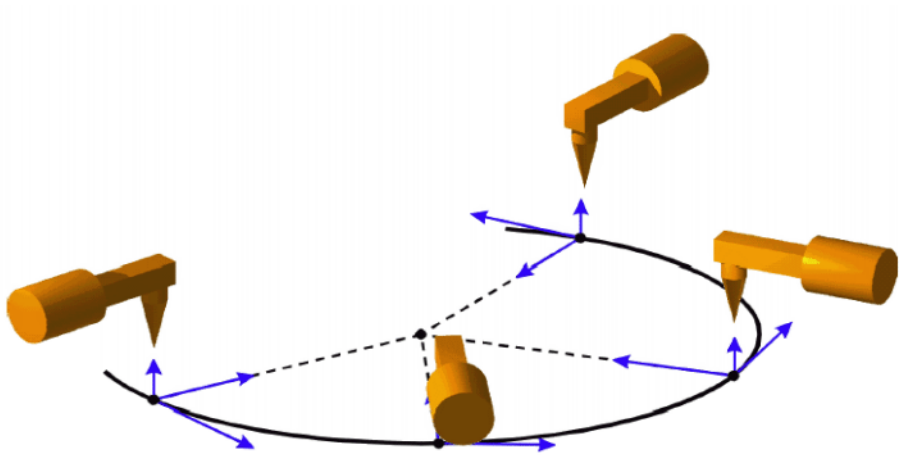


Fig. 8-9: Constant orientation, path-related

$ORI_TYPE = #VAR, $CIRC_TYPE = #PATH:

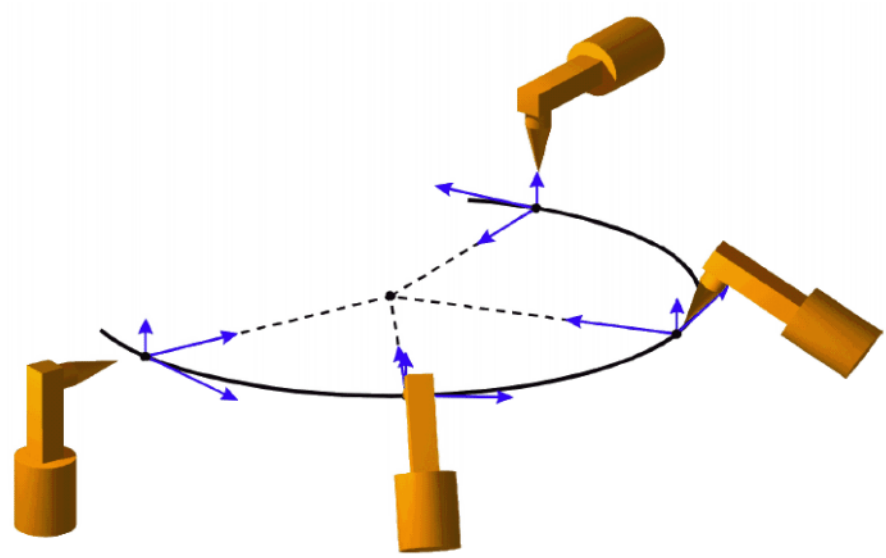


Fig. 8-10: Variable orientation, path-related

$ORI_TYPE = #CONSTANT, $CIRC_TYPE = #BASE:

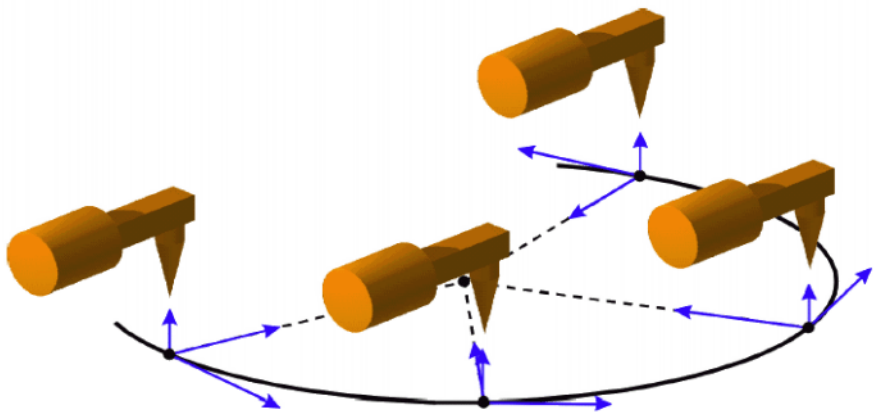


Fig. 8-11: Constant orientation, base-related

$ORI_TYPE = #VAR, $CIRC_TYPE = #BASE:

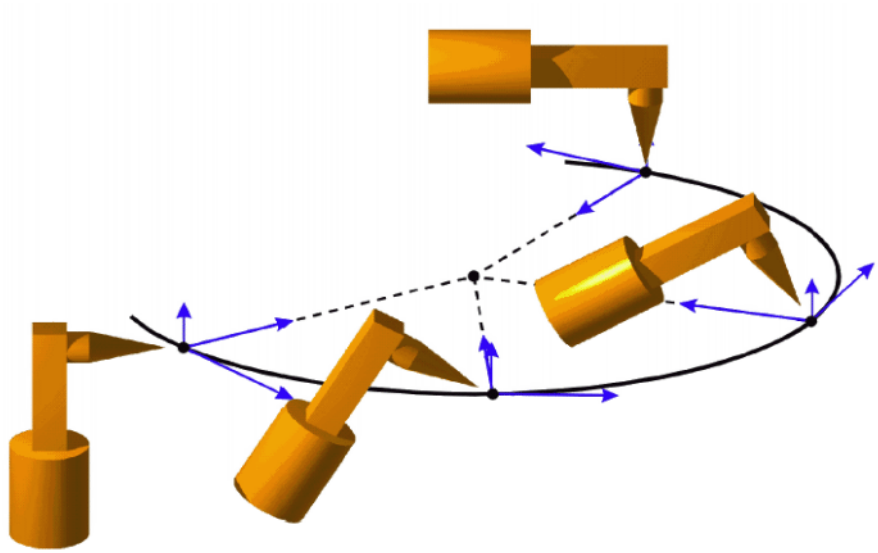


Fig. 8-12: Variable orientation, base-related

8.7 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.

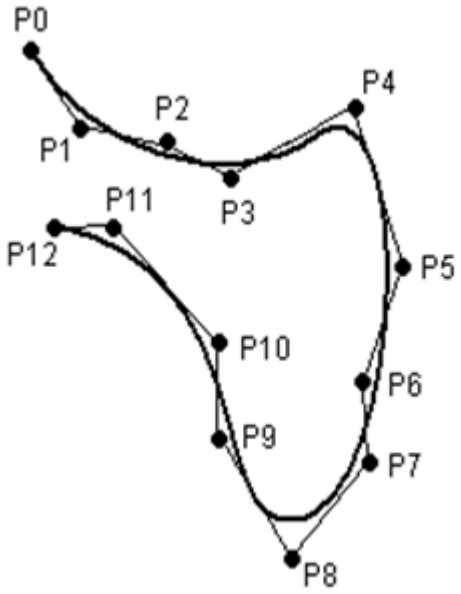


Fig. 8-13: Curved path with LIN

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.

  (>>> 8.7.1 "Velocity profile for spline motions" Page 241)
- The path always remains the same, irrespective of the override setting, velocity or acceleration.
- Circles and tight radii are executed with great precision.

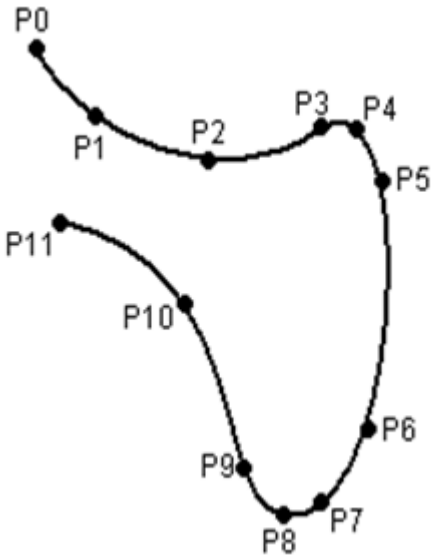


Fig. 8-14: Curved path with spline block

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.

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.

8.7.1 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.

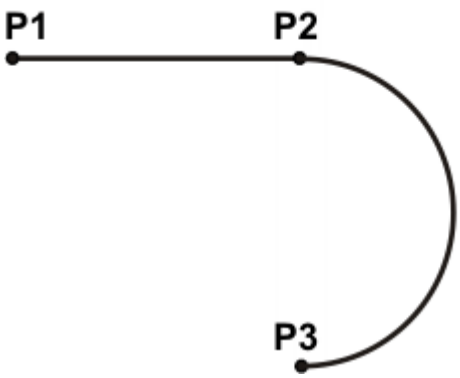


Fig. 8-15: Exact positioning at P2

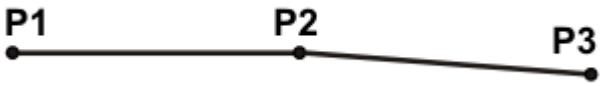


Fig. 8-16: Exact positioning at P2

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.

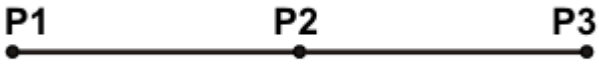


Fig. 8-17: P2 is executed without exact positioning.

- 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.)

8.7.2 Block selection with spline motions

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
3 … 9	Spline segments
10	End of the spline block

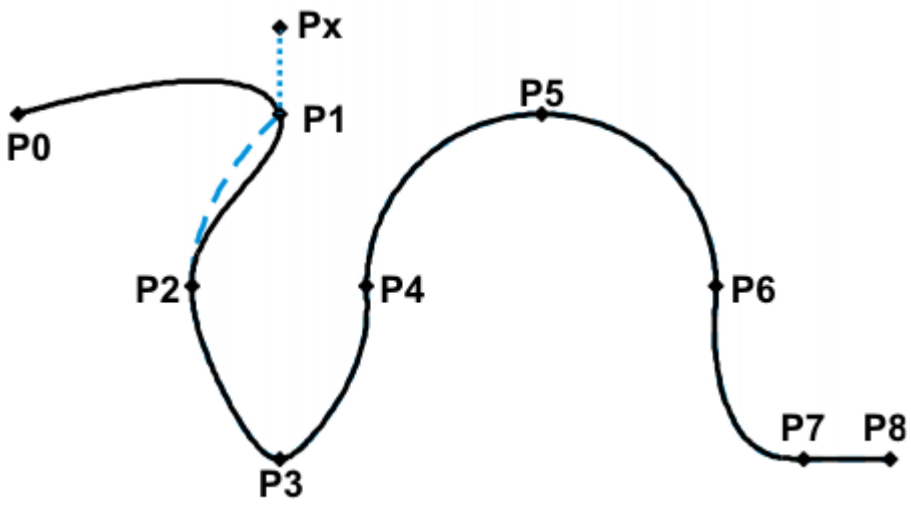


Fig. 8-18: Example: modified path in the case of block selection to P1

8.7.3 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
```

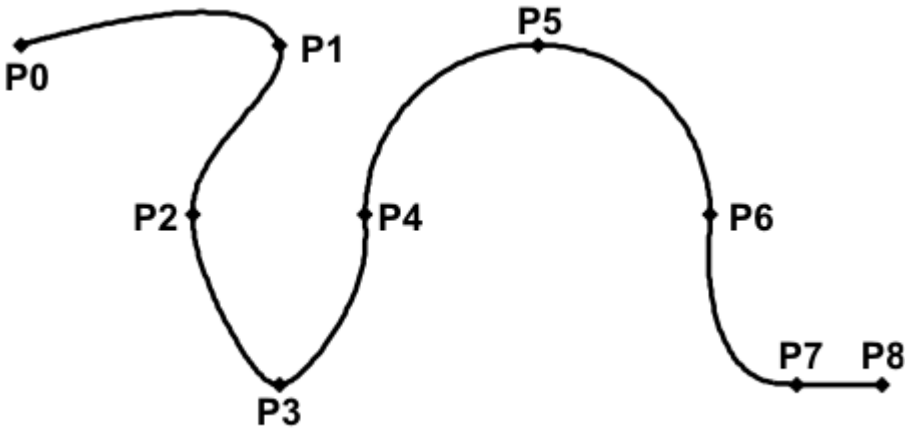


Fig. 8-19: Original path

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.

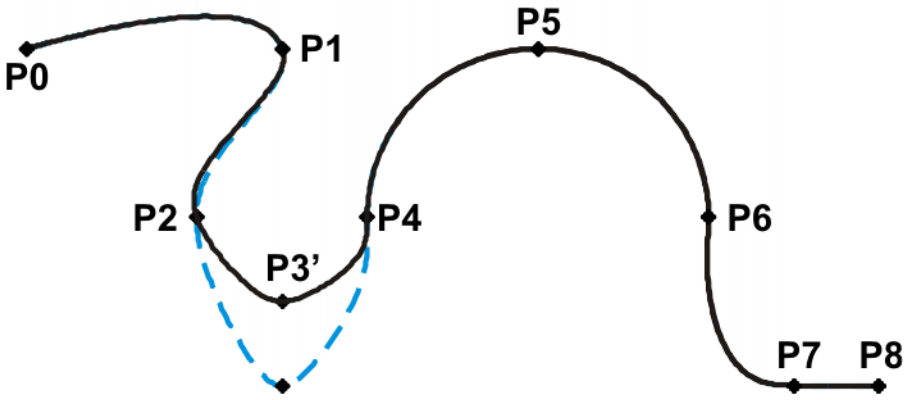


Fig. 8-20: Point has been offset

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
```

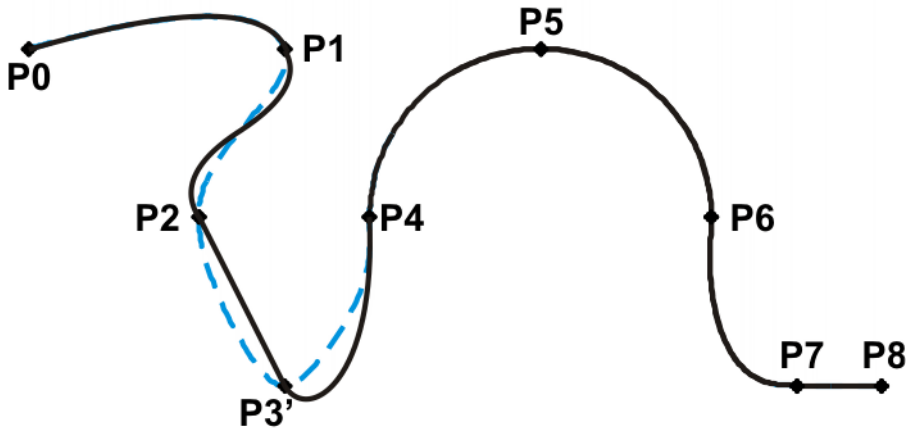


Fig. 8-21: Segment type has been changed

Example 2

```
...
SPLINE
 SPL {X 100, Y 0, ...}
 SPL {X 102, Y 0}
 SPL {X 104, Y 0}
 SPL {X 204, Y 0}
ENDSPLINE
```

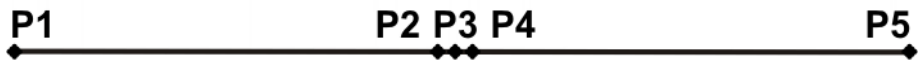


Fig. 8-22: Original path

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. 8-23: Point has been offset

Remedy:

- Distribute the points more evenly
- Program straight lines (except very short ones) as SLIN segments

8.7.4 Approximate positioning with spline motions

Approximate positioning is not currently supported for spline motions.

```
1  LIN P1 C_DIS
2  CIRC P2, P3 C_DIS
3  SPLINE
4   SPL P4
5   SCIRC P5, P6
6   SPL P7
7   SLIN P8
8  ENDSPLINE
9  LIN P9 C_DIS
10 LIN P10 C_DIS
```

Line	Description
2	P3 is the start point for the spline motion. It is addressed with exact positioning.
8	Approximate positioning cannot be programmed for P8.
9	P9 is approximated.

8.7.5 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

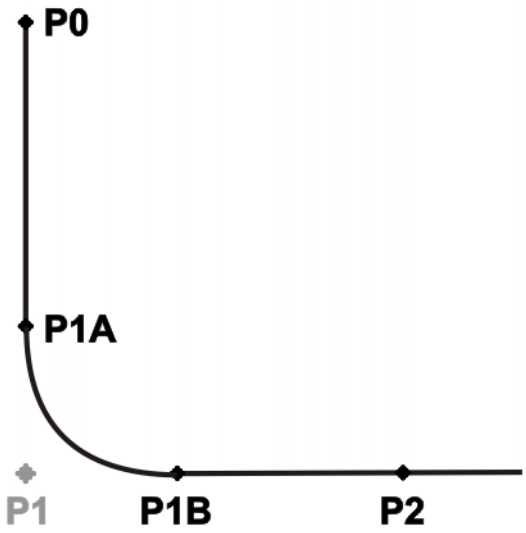


Fig. 8-24: Approximated motion - spline motion

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.

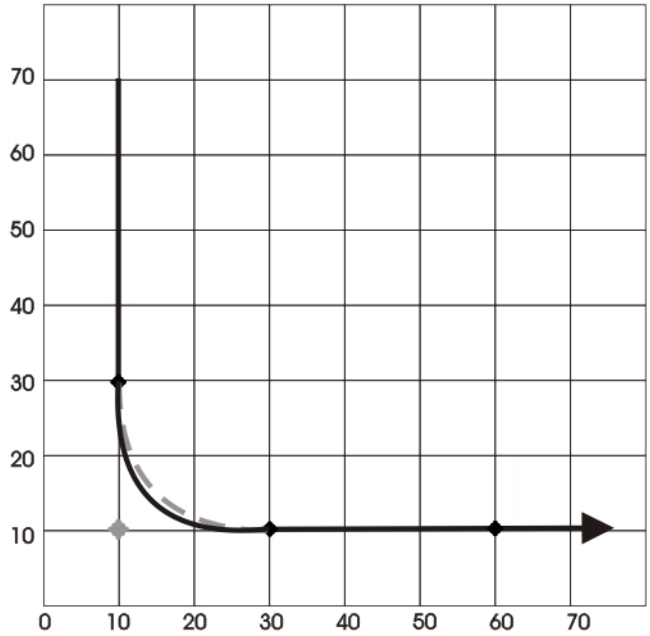


Fig. 8-25: Example: Approximated motion - spline motion 1

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
```

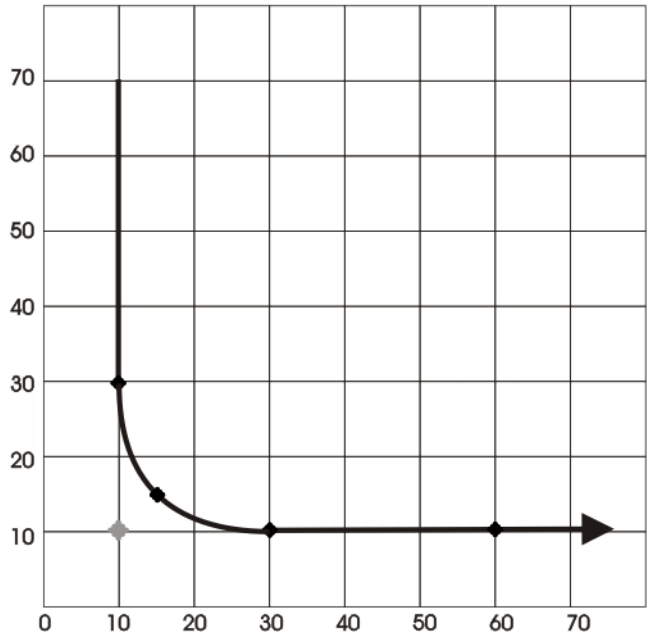


Fig. 8-26: Example: Approximated motion - spline motion 2

With the additional point, the path now corresponds to the approximate positioning arc.

8.8 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 parameter**

(>>> 9.3.4 "Option window “Motion parameter” (spline motion)" Page 266)

The orientation control can be defined in the spline block or in the individual segment. Settings in the segment overwrite the setting in the spline block.

SLIN segment

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 individual spline segments, not for the spline block. It is used if no specific orientation is required at a point. (>>> "#IGNORE" Page 250)

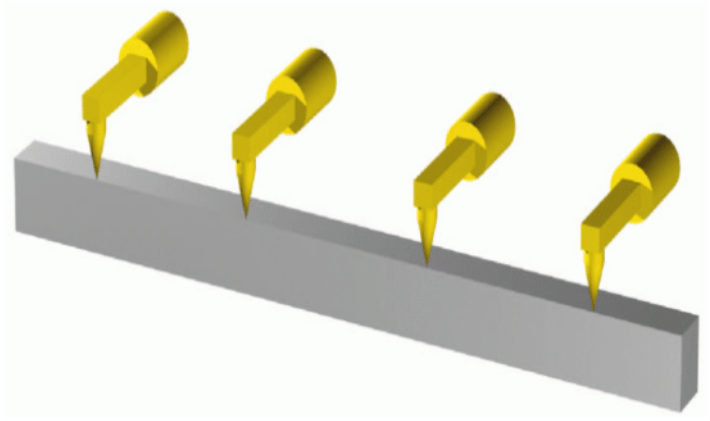


Fig. 8-27: Orientation control - Constant

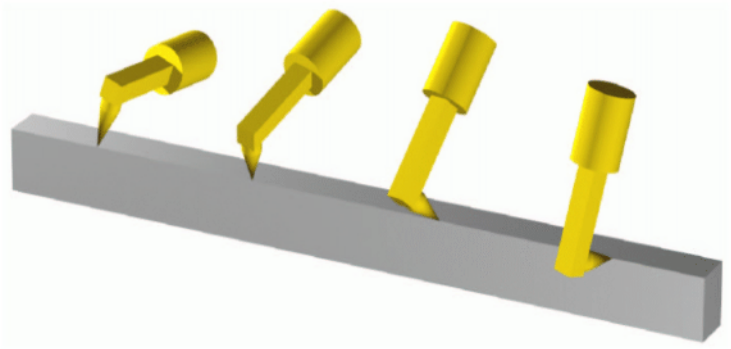


Fig. 8-28: Standard

#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 segment

During SCIRC motions, the robot controller takes the programmed orientation of the auxiliary point into consideration. The same orientation control options are available for selection for SCIRC motions as for SLIN motions.

It is also possible to define for SCIRC motions whether the orientation control is to be space-related or path-related.

(>>> 8.8.1 "SCIRC: reference system for the orientation control" Page 250)

8.8.1 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

(>>> 8.6.1 "Combinations of $ORI_TYPE and $CIRC_TYPE" Page 237)

8.9 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.

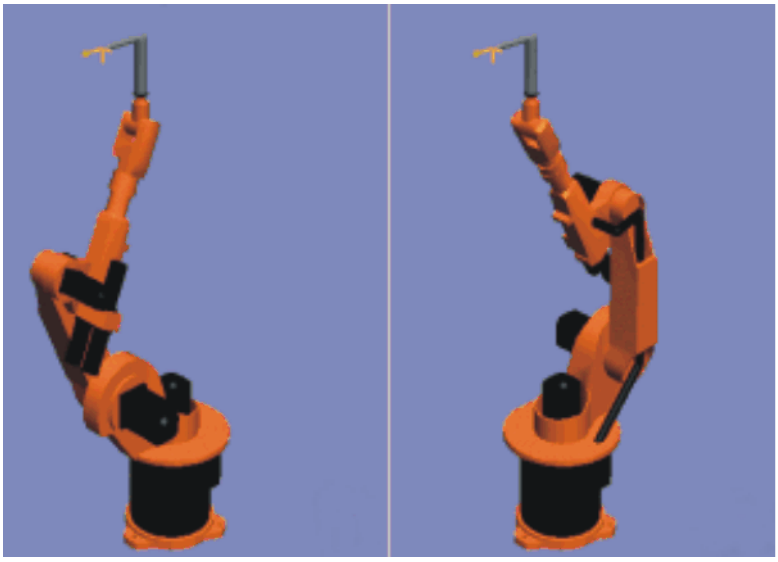


Fig. 8-29: Example: Same TCP position, different axis position

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.

8.9.1 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.

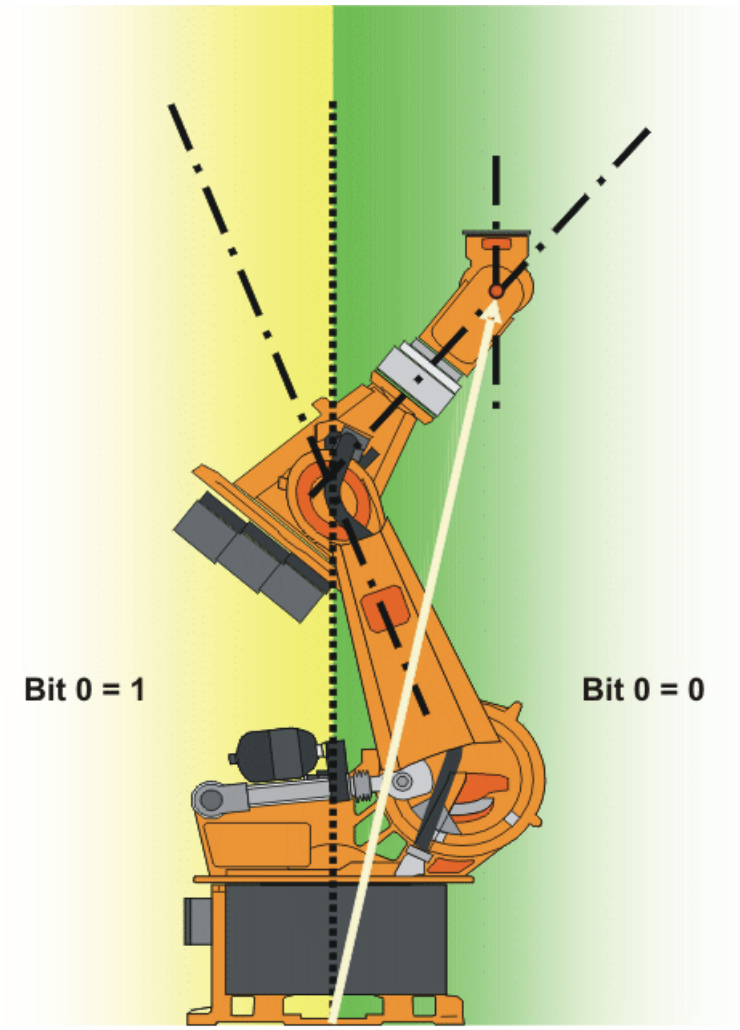


Fig. 8-30: Example: The intersection of the wrist axes (red dot) is in the basic area.

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.

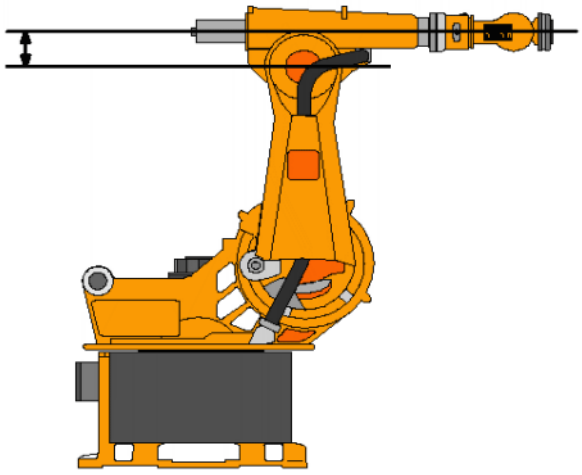


Fig. 8-31: Offset between A3 and A4 – example: KR 30

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

8.9.2 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 x = 0: angle of axis x+1 ≥ 0°

Bit x = 1: angle of axis x+1 < 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

8.10 Singularities

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.

In the case of SCARA robots, only the extended position singularity can arise. In this case, the robot starts to move extremely fast.

9 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. (>>> 10 "Programming for user group “Expert” (KRL syntax)" Page 285)

9.1 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.

9.2 Programming PTP, LIN and CIRC motions

9.2.1 Programming a PTP motion

Caution!
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.
   (>>> 9.2.2 "Inline form for PTP motions" Page 257)
5. Save the instruction by pressing the **Cmd Ok** softkey.

9.2.2 Inline form for PTP motions

Fig. 9-1: 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. (>>> 9.1 "Names in inline forms" Page 257) Position the cursor in this box to edit the point data. The corresponding option window is opened. (>>> 9.2.7 "Option window “Frames”" Page 261)
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. (>>> 9.1 "Names in inline forms" Page 257) Position the cursor in this box to edit the motion data. The corresponding option window is opened. (>>> 9.2.8 "Option window “Motion parameter” (PTP)" Page 262)

9.2.3 Programming a LIN motion

Caution!
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.
   (>>> 9.2.4 "Inline form for LIN motions" Page 258)
5. Save the instruction by pressing the **Cmd Ok** softkey.

9.2.4 Inline form for LIN motions

Fig. 9-2: 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. (>>> 9.1 "Names in inline forms" Page 257) Position the cursor in this box to edit the point data. The corresponding option window is opened. (>>> 9.2.7 "Option window “Frames”" Page 261)
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. (>>> 9.1 "Names in inline forms" Page 257) Position the cursor in this box to edit the motion data. The corresponding option window is opened. (>>> 9.2.9 "Option window “Motion parameter” (LIN, CIRC)" Page 263)

9.2.5 Programming a CIRC motion

Caution!
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.
   (>>> 9.2.6 "Inline form for CIRC motions" Page 260)
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.

9.2.6 Inline form for CIRC motions

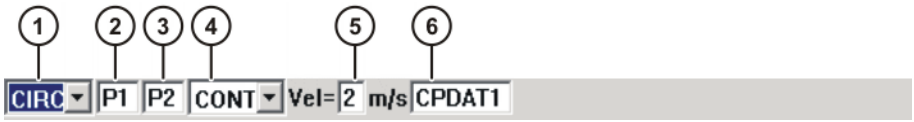


Fig. 9-3: Inline form for CIRC motions

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. (>>> 9.1 "Names in inline forms" Page 257)
3	Name of the end point The system automatically generates a name. The name can be overwritten. (>>> 9.1 "Names in inline forms" Page 257) Position the cursor in this box to edit the point data. The corresponding option window is opened. (>>> 9.2.7 "Option window “Frames”" Page 261)
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. (>>> 9.1 "Names in inline forms" Page 257) Position the cursor in this box to edit the motion data. The corresponding option window is opened. (>>> 9.2.9 "Option window “Motion parameter” (LIN, CIRC)" Page 263)

9.2.7 Option window “Frames”

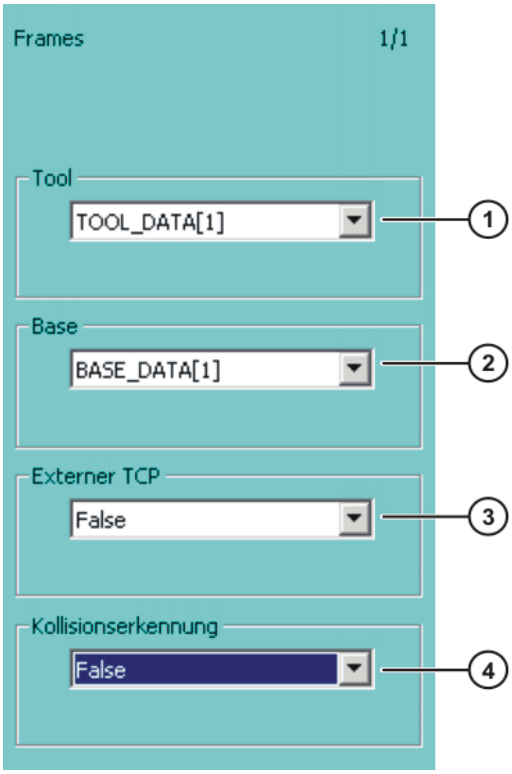


Fig. 9-4: Option window: Frames

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	■ **True**: For this motion, the robot controller calculates the axis torques. These are required for collision detection. ■ **False**: For this motion, the robot controller does not calculate the axis torques. Collision detection is thus not possible for this motion.

9.2.8 Option window “Motion parameter” (PTP)

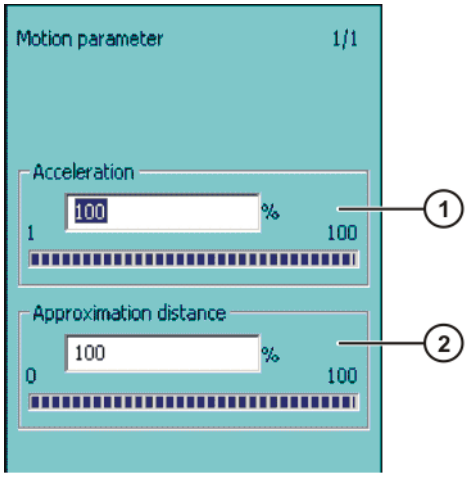


Fig. 9-5: Option window “Motion parameter” (PTP)

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. (>>> 6.14.2 "Changing the unit of the approximation distance for PTP" Page 140) 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**

9.2.9 Option window “Motion parameter” (LIN, CIRC)

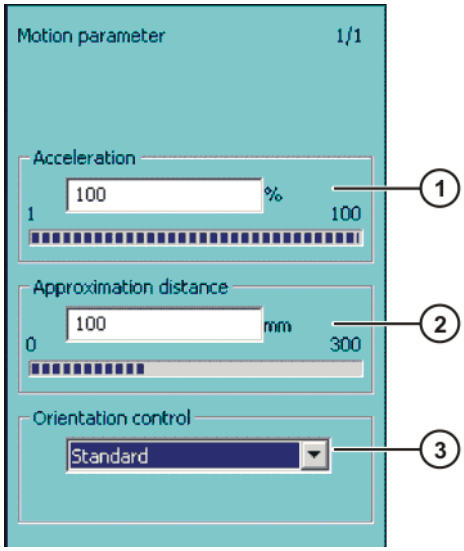


Fig. 9-6: Option window “Motion parameter” (LIN, CIRC)

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.

9.3 Programming spline motions

Overview

Step	Description
1	Program the spline block. (>>> 9.3.2 "Programming a spline block" Page 265)
2	Program the spline segments. (>>> 9.3.5 "Programming an SPL segment" Page 267) (>>> 9.3.6 "Programming an SLIN segment" Page 267) (>>> 9.3.7 "Programming an SCIRC segment" Page 268)
3	If required, and only in the user group “Expert”: Program PATH trigger. (>>> 10.11.3 "TRIGGER WHEN PATH (for SPLINE)" Page 331)

As for all motion instructions, the start point of a spline motion is the end point of the previous motion.

9.3.1 Programming tips for spline motions

- A spline block should cover one process (e.g. an 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 these trigger exact positioning. Exact positioning can be avoided as follows:
  - Program SPL segments between SLIN and SCIRC segments. The length of the SPL segments must be greater than 0.5 mm.
  - Replace an SLIN segment with several SPL segments. If SPL points are situated on a straight line, then the path will also be a straight line.
- Avoid successive points with identical Cartesian coordinates, as these trigger exact positioning.
- 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.

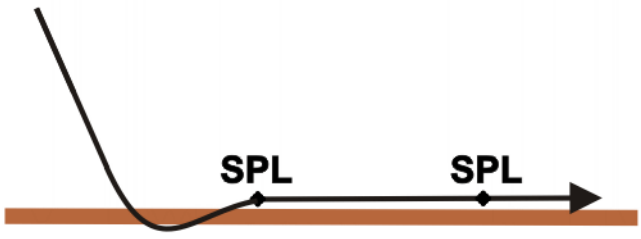


Fig. 9-7: Collision with work surface

A collision can be avoided as follows:

- Program the first segment on the work surface as SLIN.
- Additionally, if required: Insert an SLIN segment before the first point. The following precondition must be met: $2/3 \leq a/b \leq 3/2$

a = distance from start point of the SPL segment to intersection of the SLIN segments

b = distance from intersection of the SLIN segments to end point of the SPL segment

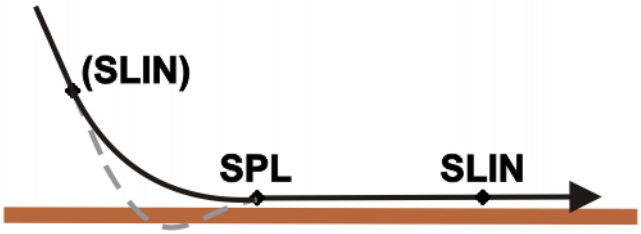


Fig. 9-8: Avoiding a collision with the work surface

9.3.2 Programming a spline block

Precondition

- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Position the cursor in the line **after** which the spline motion is to be inserted.
2. Select the menu sequence **Commands** > **Motion** > **SPLINE Block**.
3. Set the parameters in the inline form.
   (>>> 9.3.3 "Inline form for spline block" Page 265)
4. Press the **Cmd OK** softkey.
5. Press the **Fold open/cls** softkey. Lines can now be inserted into the spline block.

Description

A spline block may contain the following:

- Spline segments (only limited by the memory capacity. As a rule, at least 1,000 segments are possible.)
- PATH trigger
- Comments
- Blank lines
- Inline commands from technology packages that support the spline function

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.

9.3.3 Inline form for spline block

Fig. 9-9: Inline form for spline block

Item	Description
1	Name of the spline motion. 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. (>>> 9.2.7 "Option window “Frames”" Page 261)
2	The velocity is valid by default for the entire spline motion. It can also be defined separately for individual segments. ■ **0.001 … 2 m/s**
3	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. (>>> 9.3.4 "Option window “Motion parameter” (spline motion)" Page 266) The motion data are valid by default for the entire spline motion. They can also be defined separately for individual segments.

9.3.4 Option window “Motion parameter” (spline motion)

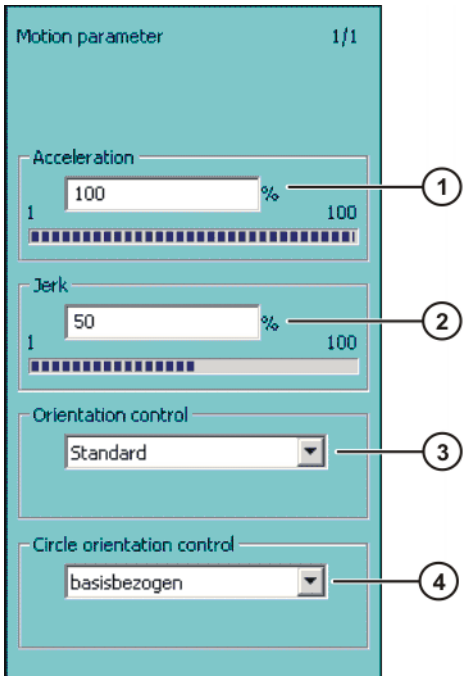


Fig. 9-10: Option window “Motion parameter” (spline motion)

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 (>>> 8.8 "Orientation control SPLINE" Page 249)
4	Only for spline block and SCIRC segments: Circle orientation control selection ■ **Base-related** ■ **Path-related**

9.3.5 Programming an SPL segment

Caution!
When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

Description

The robot guides the TCP to the end point. The robot controller selects a suitable path.

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**.
4. Set the parameters in the inline form.
   (>>> 9.3.8 "Inline form “Spline Segment”" Page 268)
5. Press the **Cmd OK** softkey.

9.3.6 Programming an SLIN segment

Caution!
When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

Description

The robot guides the TCP along the shortest path to the end point. The shortest path is always a straight line.

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** > **SLIN**.
4. Set the parameters in the inline form.
   (>>> 9.3.8 "Inline form “Spline Segment”" Page 268)
5. Press the **Cmd OK** softkey.

9.3.7 Programming an SCIRC segment

Caution!
When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

Description

The robot guides the TCP along a circular path to the end point. The circular path is defined by a start point, auxiliary point and end point.

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.
   (>>> 9.3.8 "Inline form “Spline Segment”" Page 268)
5. Press the **Teach Aux** softkey.
6. Move the TCP to the end point.
7. Press the **Cmd OK** softkey.

9.3.8 Inline form “Spline Segment”

Fig. 9-11: Inline form “Spline Segment”

The boxes in the inline form can be displayed or hidden one by one using the **Toggle Param** softkey.

Item	Description
1	Select the motion type for the spline segment using the **Toggle Cmd** softkey. ■ **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. Position the cursor in the box of the end point to edit the point data. The corresponding option window is opened. (>>> 9.3.9 "Option window “Frames” (spline segment)" Page 269)
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 motion data. The corresponding option window is opened. (>>> 9.3.4 "Option window “Motion parameter” (spline motion)" Page 266) The motion data only refer to the segment to which they belong. They have no effect on subsequent segments.

9.3.9 Option window “Frames” (spline segment)

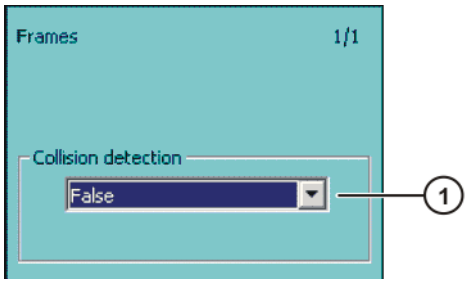


Fig. 9-12: Inline form “Frames” (spline segment)

Item	Description
1	■ **True**: For this motion, the robot controller calculates the axis torques. These are required for collision detection. ■ **False**: For this motion, the robot controller does not calculate the axis torques. Collision detection is thus not possible for this motion.

9.4 Modifying motion parameters

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 the **Change** softkey. The inline form for this instruction is opened.
3. Modify parameters.
4. Save changes by pressing the **Cmd Ok** softkey.

9.5 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.

9.6 Programming logic instructions

9.6.1 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/out-

put 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]

9.6.2 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.
   (>>> 9.6.3 "Inline form “OUT”" Page 271)
4. Save the instruction by pressing the **Cmd Ok** softkey.

9.6.3 Inline form “OUT”

The instruction sets a digital output.

Fig. 9-13: 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

9.6.4 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.
   (>>> 9.6.5 "Inline form “PULSE”" Page 272)
4. Save the instruction by pressing the **Cmd Ok** softkey.

9.6.5 Inline form “PULSE”

The instruction sets a pulse of a defined length.

Fig. 9-14: 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**

9.6.6 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.
   (>>> 9.6.7 "Inline form “ANOUT” (static)" Page 273)
   (>>> 9.6.8 "Inline form “ANOUT” (dynamic)" Page 273)
4. Save the instruction by pressing the **Cmd Ok** softkey.

9.6.7 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. 9-15: Inline form "ANOUT" (static)

Item	Description
1	Analog output number ■ **CHANNEL_1 … CHANNEL_32**
2	Factor for the voltage ■ **0 … 1** (intervals: 0.01)

9.6.8 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. 9-16: 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.
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**

9.6.9 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.
   (>>> 9.6.10 "Inline form “WAIT”" Page 274)
4. Save the instruction by pressing the **Cmd Ok** softkey.

9.6.10 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.

Fig. 9-17: Inline form "WAIT"

Item	Description
1	Wait time ■ **≥ 0** s

9.6.11 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.
   (>>> 9.6.12 "Inline form “WAITFOR”" Page 275)
4. Save the instruction by pressing the **Cmd Ok** softkey.

9.6.12 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.

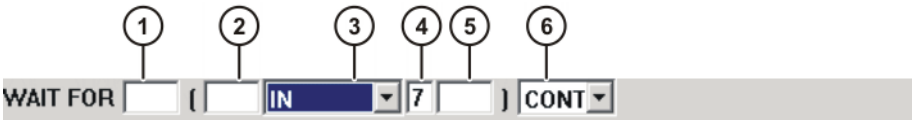


Fig. 9-18: Inline form "WAITFOR"

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

9.6.13 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.
   (>>> 9.6.14 "Inline form “SYN OUT”, option “START/END”" Page 277)
   (>>> 9.6.15 "Inline form “SYN OUT”, option “PATH”" Page 280)
4. Save the instruction by pressing the **Cmd Ok** softkey.

9.6.14 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.

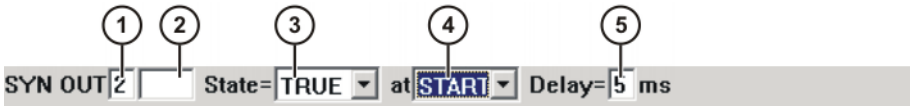


Fig. 9-19: Inline form SYN OUT, option START/END

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**: (>>> 9.6.15 "Inline form “SYN OUT”, option “PATH”" Page 280)
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
```

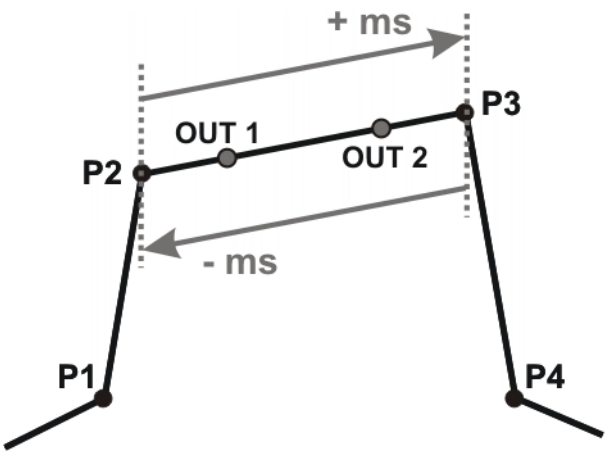


Fig. 9-20

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
```

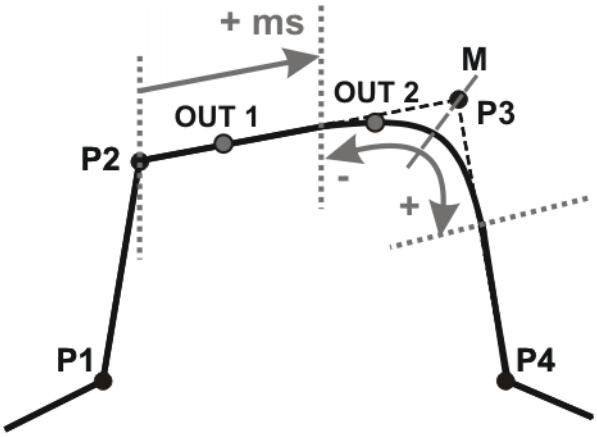


Fig. 9-21

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
```

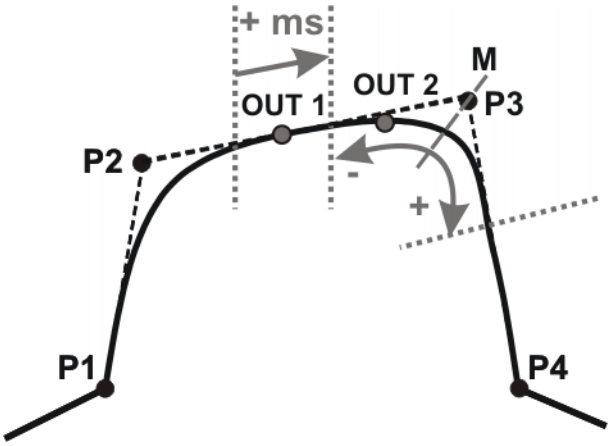


Fig. 9-22

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.

9.6.15 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.

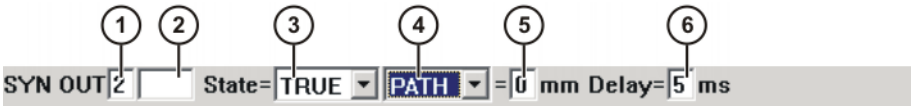


Fig. 9-23: Inline form SYN OUT, option PATH

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**: (>>> 9.6.14 "Inline form “SYN OUT”, option “START/END”" Page 277) ■ **END**: (>>> 9.6.14 "Inline form “SYN OUT”, option “START/END”" Page 277)
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
```

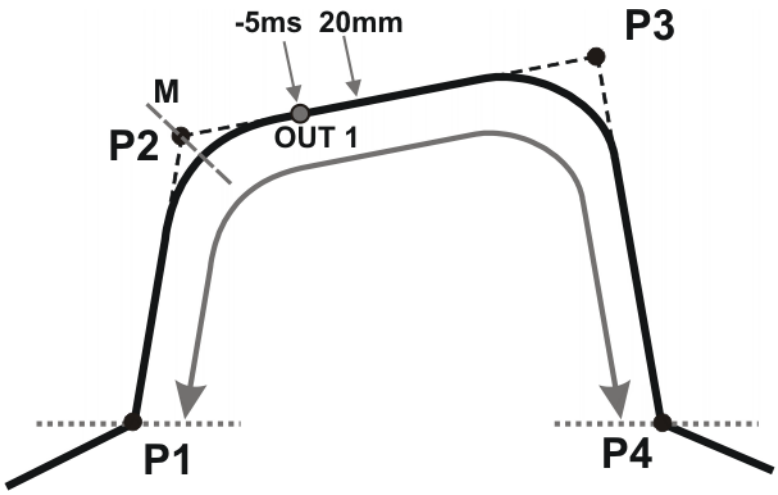


Fig. 9-24

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
```

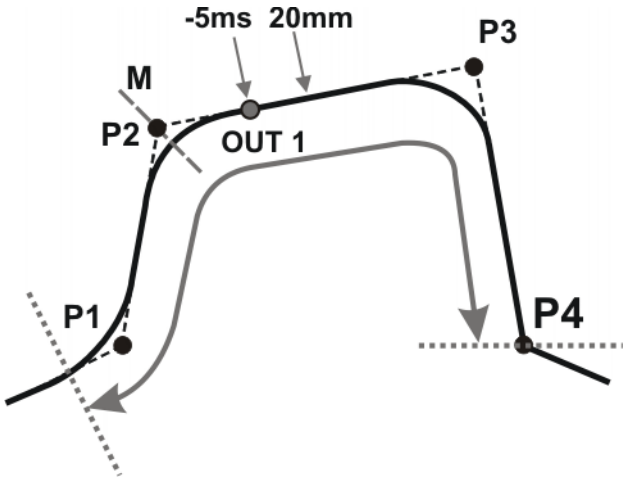


Fig. 9-25

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.

9.6.16 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.
   (>>> 9.6.17 "Inline form “SYN PULSE”" Page 283)
4. Save the instruction by pressing the **Cmd Ok** softkey.

9.6.17 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.

Fig. 9-26: 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. (>>> 9.6.14 "Inline form “SYN OUT”, option “START/END”" Page 277) ■ **PATH**: The pulse is triggered at the end point of the motion block. See SYN OUT for examples and switching limits. (>>> 9.6.15 "Inline form “SYN OUT”, option “PATH”" Page 280)
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.

9.6.18 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 the **Change** softkey. The inline form for this instruction is opened.
3. Modify parameters.
4. Save changes by pressing the **Cmd Ok** softkey.

10 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.

10.1 Overview of KRL syntax

Variables and declarations	
DECL	(>>> 10.4.1 "DECL" Page 291)
ENUM	(>>> 10.4.2 "ENUM" Page 292)
IMPORT … IS	(>>> 10.4.3 "IMPORT ... IS" Page 293)
STRUC	(>>> 10.4.4 "STRUC" Page 294)

Motion programming	
CIRC	(>>> 10.5.5 "CIRC" Page 300)
CIRC_REL	(>>> 10.5.6 "CIRC_REL" Page 301)
LIN	(>>> 10.5.3 "LIN" Page 297)
LIN_REL	(>>> 10.5.4 "LIN_REL" Page 298)
PTP	(>>> 10.5.1 "PTP" Page 295)
PTP_REL	(>>> 10.5.2 "PTP_REL" Page 296)
SPLINE … ENDSPLINE	(>>> 10.6.1 "SPLINE ... ENDSPLINE" Page 303)
SCIRC	(>>> 10.6.4 "SCIRC" Page 304)
SLIN	(>>> 10.6.3 "SLIN" Page 304)
SPL	(>>> 10.6.2 "SPL" Page 304)

Program execution control	
CONTINUE	(>>> 10.7.1 "CONTINUE" Page 305)
EXIT	(>>> 10.7.2 "EXIT" Page 305)
FOR … TO … ENDFOR	(>>> 10.7.3 "FOR ... TO ... ENDFOR" Page 306)
GOTO	(>>> 10.7.4 "GOTO" Page 307)
HALT	(>>> 10.7.5 "HALT" Page 307)
IF … THEN … ENDIF	(>>> 10.7.6 "IF ... THEN ... ENDIF" Page 308)
LOOP … ENDLOOP	(>>> 10.7.7 "LOOP ... ENDLOOP" Page 308)
REPEAT … UNTIL	(>>> 10.7.8 "REPEAT ... UNTIL" Page 309)
SWITCH … CASE … ENDSWITCH	(>>> 10.7.9 "SWITCH ... CASE ... ENDSWITCH" Page 309)
WAIT … FOR	(>>> 10.7.10 "WAIT FOR" Page 311)
WAIT … SEC	(>>> 10.7.11 "WAIT SEC" Page 311)
WHILE … ENDWHILE	(>>> 10.7.12 "WHILE ... ENDWHILE" Page 311)

Inputs/outputs	
ANIN	(>>> 10.8.1 "ANIN" Page 312)
ANOUT	(>>> 10.8.2 "ANOUT" Page 313)
PULSE	(>>> 10.8.3 "PULSE" Page 314)
SIGNAL	(>>> 10.8.4 "SIGNAL" Page 318)

Subprograms and functions	
RETURN	(>>> 10.9.1 "RETURN" Page 319)

Interrupt programming	
BRAKE	(>>> 10.10.1 "BRAKE" Page 320)
INTERRUPT	(>>> 10.10.3 "INTERRUPT" Page 322)
INTERRUPT … DECL … WHEN … DO	(>>> 10.10.2 "INTERRUPT ... DECL ... WHEN ... DO" Page 320)
RESUME	(>>> 10.10.4 "RESUME" Page 324)

Path-related switching actions (=Trigger)	
TRIGGER WHEN DISTANCE	(>>> 10.11.1 "TRIGGER WHEN DISTANCE" Page 325)
TRIGGER WHEN PATH	(>>> 10.11.2 "TRIGGER WHEN PATH" Page 328)
TRIGGER WHEN PATH (for spline)	(>>> 10.11.3 "TRIGGER WHEN PATH (for SPLINE)" Page 331)

Communication
(>>> 10.12 "Communication" Page 333)

System functions	
VARSTATE()	(>>> 10.13.1 "VARSTATE()" Page 333)

Manipulating string variables
(>>> 10.14 "Editing string variables" Page 335)

10.2 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`

10.3 Important KRL terms

10.3.1 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.

10.3.2 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 (>>> 10.3.5 "Areas of validity" Page 289).
- **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.

10.3.3 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.
- 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.

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	ENDFCT
BOOL	ENDFOR
CHAR	ENDIF
CAST_FROM	ENDLOOP
CAST_TO	ENDSWITCH
CCLOSE	ENDWHILE
CHANNEL	EXT
CIOCTL	EXTFCT
CONFIRM	FALSE
CONST	FRAME
COPEN	GLOBAL
CREAD	INT
CWRITE	MAXIMUM
DEF	MINIMUM
DEFAULT	POS
DEFDAT	PRIO
DEFFCT	PUBLIC
E6AXIS	SREAD
E6POS	SWRITE
END	REAL
ENDDAT	TRUE

10.3.4 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 ■ **-2³¹-1 … 2³¹-1** 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 characters ■ **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
```

10.3.5 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.

Data object	Area of validity
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.
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 ()*

10.3.6 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

10.4 Variables and declarations

10.4.1 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	(>>> 10.3.5 "Areas of validity" Page 289)
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"
```

10.4.2 ENUM

Description

Definition of an enumeration type (= ENUM data type)

Syntax

<GLOBAL> ENUM *NameEnumType Constant1*<, ..., *ConstantN*>

Explanation of the syntax

Element	Description
GLOBAL	(>>> 10.3.5 "Areas of validity" Page 289)
NameEnum-Type	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
```

10.4.3 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 DataListName PUBLIC
```

Syntax

```
IMPORT DataType Name IS /R1/DataSource..NameOld
```

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.
NameOld	Name in the external data list of the data object to be imported. Unlike with *Name*, no brackets are specified for arrays.

DataSource and *NameOld* 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
```

10.4.4 STRUC

Description

Definition of a structure type (= STRUC data type). Several data types are combined to form a new data type.

Syntax

`<GLOBAL> STRUC` *NameStructureType DataType1 Component1A*<, *Component1B*, ...> < , *DataType2 Component2A*<, *Component2B*, ...>>

Explanation of the syntax

Element	Description
GLOBAL	(>>> 10.3.5 "Areas of validity" Page 289)
NameStructureType	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}}
```

10.5 Motion programming: PTP, LIN, CIRC

10.5.1 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` <*Approximate_Positioning*>>

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, *Approximate_Positioning* must also be specified.
Approximate_Positioning	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}
```

10.5.2 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` *<Approximate_Positioning>>*

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, *Approximate_Positioning* must also be specified.
Approximate_ Positioning	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}
```

10.5.3 LIN

Description

Executes a linear motion to the end point. The coordinates of the end point are absolute.

Syntax

`LIN` *End_Point* <*Approximate_Positioning*>

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.
Approximate_ Positioning	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}
```

10.5.4 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* <*Approximate_Positioning*> <`#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.
Approximate_ Positioning	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
```

10.5.5 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*`>` `<`*Approximate positioning*`>`

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.
Approximate positioning	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
```

10.5.6 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*`>` `<`*Approximate positioning*`>`

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.
Approximate positioning	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
```

10.6 Motion programming: spline

Overview

Step	Description
1	Program the spline block. (>>> 10.6.1 "SPLINE ... ENDSPLINE" Page 303)
2	Program the spline segments. (>>> 10.6.2 "SPL" Page 304) (>>> 10.6.3 "SLIN" Page 304) (>>> 10.6.4 "SCIRC" Page 304)
3	If required, and only in the user group “Expert”: Program PATH trigger. (>>> 10.11.3 "TRIGGER WHEN PATH (for SPLINE)" Page 331)

As for all motion instructions, the start point of a spline motion is the end point of the previous motion.

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)

Explanations of the motion type SPLINE can be found in the following section of this documentation:
Programming tips relating to the motion type SPLINE can be found in the following section of this documentation: (>>> 9.3.1 "Programming tips for spline motions" Page 264)

10.6.1 SPLINE ... ENDSPLINE

Syntax

```
SPLINE <WITH SysVarSpline = Value>
Spline-Segment
<Trigger>
<Spline-Segment2
<Trigger2>>
…
<Spline-SegmentN
<TriggerN>>
ENDSPLINE
```

Explanation of the syntax

Element	Description
SysVar-Spline	The following system variables may be used: $BASE, $TOOL, $IPO_MODE, $LOAD, $VEL, $VEL_EXTAX[], $ACC, $ACC_EXTAX[], $JERK (=Jerk limitation), $ORI_TYPE, $CIRC_TYPE
Value	Value of the system variable. The unit and the range of values depend on the specific system variable. The values are valid for every segment in the spline block unless values are assigned separately to a segment.

Description

A spline block may contain the following:

- Spline segments (only limited by the memory capacity. As a rule, at least 1,000 segments are possible.)
- PATH trigger
- Comments
- Blank lines
- Inline commands from technology packages that support the spline function

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.

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
```

10.6.2 SPL

Description

The robot guides the TCP to the end point. The robot controller selects a suitable path.

This statement can only be used within a spline block.

Syntax

`SPL` *End_Point* <`WITH` *SysVarSpl* = *Value*>

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 coordinates refer to the BASE coordinate system.
SysVarSpl	The following system variables may be used: $VEL, $VEL_EXTAX[], $ACC, $ACC_EXTAX[], $JERK (=Jerk limitation), $ORI_TYPE
Value	Value of the system variable. The unit and the range of values depend on the specific system variable. The assignment applies only for this spline segment. It has no effect on subsequent segments.

Example

```
SPL P4 WITH $ACC={CP 2.0, ORI1 200, ORI2 200}
```

10.6.3 SLIN

Description

The robot guides the TCP along the shortest path to the end point. The shortest path is always a straight line.

This statement can only be used within a spline block.

Syntax

`SLIN` *End_Point* <`WITH` *SysVarSlin* = *Value*>

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 coordinates refer to the BASE coordinate system.
SysVarSlin	The following system variables may be used: $VEL, $VEL_EXTAX[], $ACC, $ACC_EXTAX[], $JERK (=Jerk limitation), $ORI_TYPE
Value	Value of the system variable. The unit and the range of values depend on the specific system variable. The assignment applies only for this spline segment. It has no effect on subsequent segments.

Example

```
SLIN P8 WITH ORI_TYPE = #IGNORE, $JERK = 30
```

10.6.4 SCIRC

Description

The robot guides the TCP along a circular path to the end point. The circular path is defined by a start point, auxiliary point and end point.

This statement can only be used within a spline block.

Syntax

`SCIRC` *Auxiliary_Point*, *End_Point* <`WITH` *SysVarScirc* = *Value*>

Explanation of the syntax

Element	Description
Auxiliary_Point	Type: POS, E6POS, FRAME 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 coordinates refer to the BASE coordinate system.
SysVarScirc	The following system variables may be used: $VEL, $VEL_EXTAX[], $ACC, $ACC_EXTAX[], $JERK (=Jerk limitation), $ORI_TYPE, $CIRC_TYPE
Value	Value of the system variable. The unit and the range of values depend on the specific system variable. The assignment applies only for this spline segment. It has no effect on subsequent segments.

Example

```
SCIRC P2, P3 WITH $CIRC_TYPE = #PATH
```

10.7 Program execution control

10.7.1 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
```

Caution!
In this case, the outputs are set in the advance run. The exact point at which they are set cannot be foreseen.

10.7.2 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
```

10.7.3 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
```

10.7.4 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` *Marker*

`...`

Marker`:`

Explanation of the syntax

Element	Description
Marker	Position to which a jump is made. At the destination position, *Marker* 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
```

10.7.5 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`

10.7.6 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
```

10.7.7 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
```

10.7.8 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
```

10.7.9 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.

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
```

10.7.10 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
```

10.7.11 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
```

10.7.12 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 execu-

tion. 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
```

10.8 Inputs/outputs

10.8.1 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
```

- 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 $CONFIG.DAT).
- The robot controller has 32 analog inputs ($ANIN[1] … $ANIN[32]).

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
```

10.8.2 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*

- 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 $CONFIG.DAT).
- The robot controller has 32 analog outputs ($ANOUT[1] … $ANOUT[32]).

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
```

10.8.3 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 Logical 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

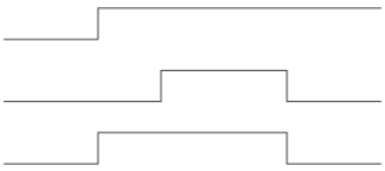


Fig. 10-1: $OUT+PULSE, example 1

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

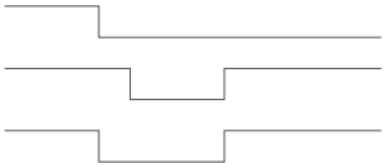


Fig. 10-2: $OUT+PULSE, example 2

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

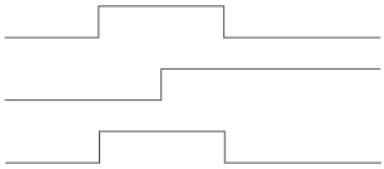


Fig. 10-3: PULSE+$OUT, example 1

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

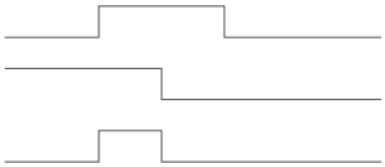


Fig. 10-4: PULSE+$OUT, example 2

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

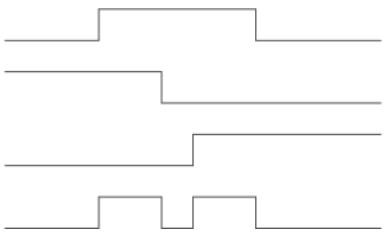


Fig. 10-5: PULSE+$OUT, example 3

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):

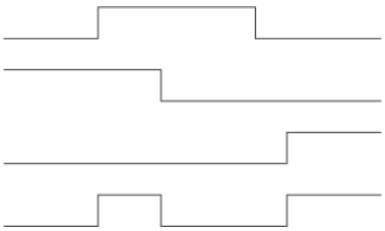


Fig. 10-6: PULSE+$OUT, example 4

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

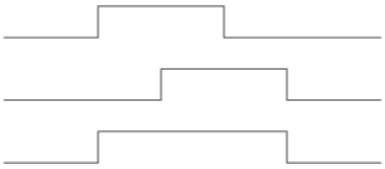


Fig. 10-7: PULSE+PULSE, example 1

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

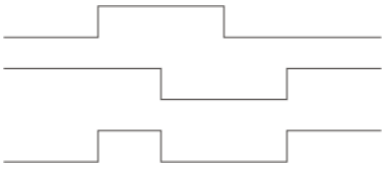


Fig. 10-8: PULSE+PULSE, example 2

```
   PULSE($OUT[50],TRUE,3)
PULSE($OUT[50],FALSE,1)
```

Actual pulse characteristic at output 50

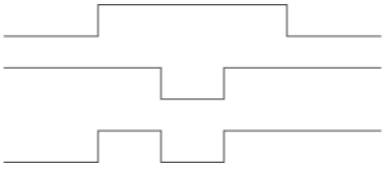


Fig. 10-9: PULSE+PULSE, example 3

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

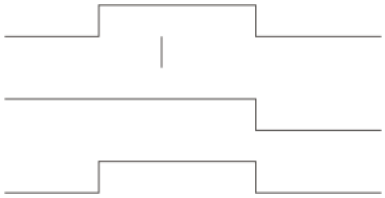


Fig. 10-10: PULSE+END, example

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
```

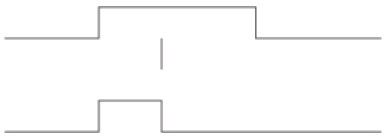


Fig. 10-11: PULSE+RESET, example

10.8.4 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]
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'
```

10.9 Subprograms and functions

10.9.1 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
```

10.10 Interrupt programming

10.10.1 BRAKE

Description

Brakes the robot motion.

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

(>>> 10.10.3 "INTERRUPT" Page 322)

10.10.2 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 (>>> 10.10.4 "RESUME" Page 324).

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! (>>> 10.10.3 "INTERRUPT" Page 322)

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
```

10.10.3 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. (>>> 10.10.2 "INTERRUPT ... DECL ... WHEN ... DO" Page 320)

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
```

10.10.4 RESUME

Description

RESUME may only occur in an interrupt program. 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
```

10.11 Path-related switching actions (=Trigger)

10.11.1 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

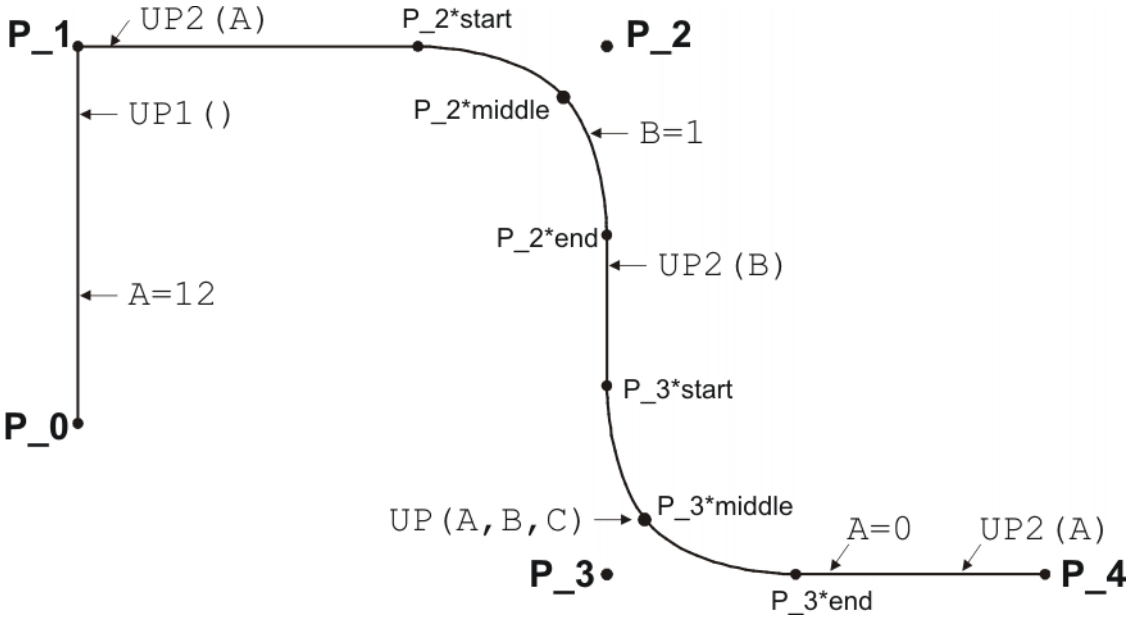


Fig. 10-12: Example of TRIGGER WHEN DISTANCE

10.11.2 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 end point must be LIN or CIRC. It must not be PTP.

The statement is executed parallel to the robot motion.

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.

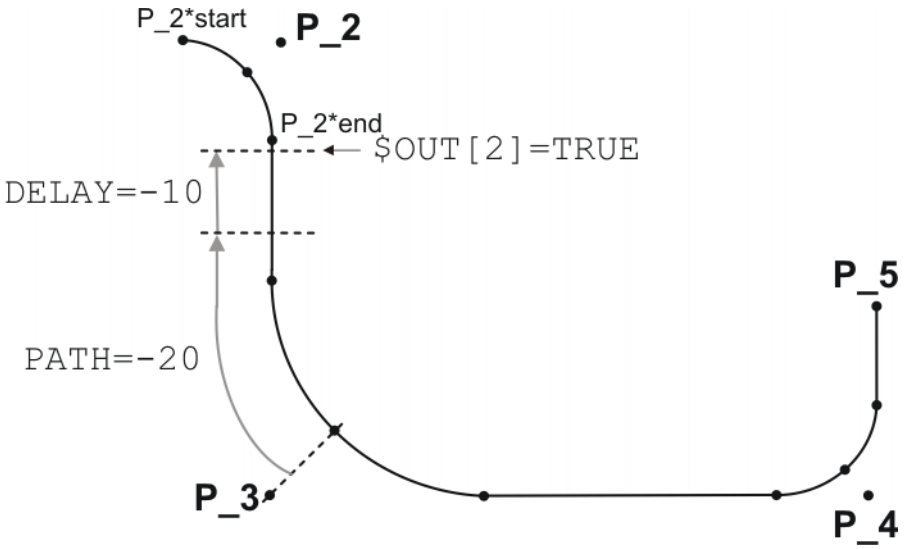


Fig. 10-13: Example of TRIGGER WHEN PATH

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.

10.11.3 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. The statement can be shifted, at most, as far as the first point before the spline block or the last point of the spline block.

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 with the function must have the attribute **Hidden**. (>>> 7.1.2 "Displaying or modifying file properties" Page 208)
- The function must be globally valid.
- The functions may only contain the following statements:
  - Write to user-defined variable
  - IF statement
  - Assignment
  - Read system variable

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 statement refers to the start point. Without ONSTART: the statement refers to the end point.
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.

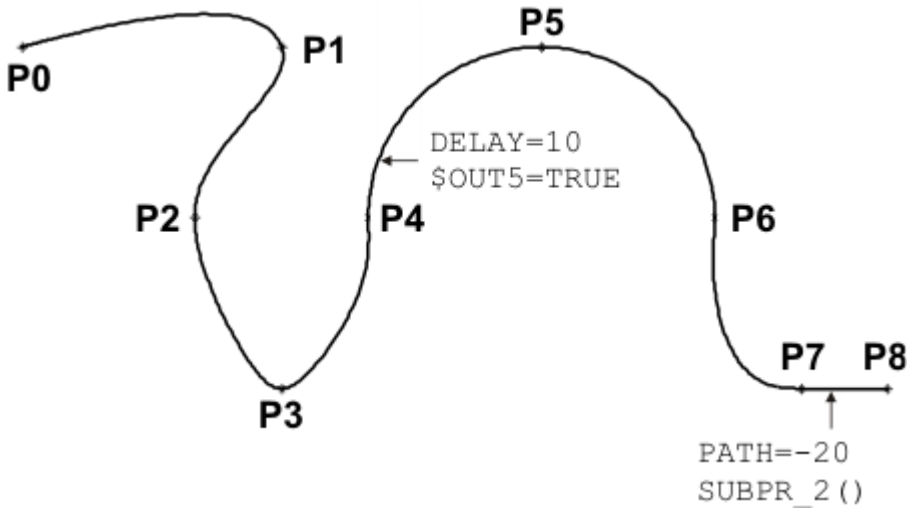


Fig. 10-14: Example of TRIGGER WHEN PATH (for spline)

10.12 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

10.13 System functions

10.13.1 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.

10.14 Editing string variables

Overview

Various functions are available for editing string variables. The functions can be used in SRC files, in SUB files and in the variable display.

- String variable length in the declaration

  (>>> 10.14.1 "String variable length in the declaration" Page 335)

- String variable length after initialization

  (>>> 10.14.2 "String variable length after initialization" Page 336)

- Deleting the contents of a string variable

  (>>> 10.14.3 "Deleting the contents of a string variable" Page 336)

- Extending a string variable

  (>>> 10.14.4 "Extending a string variable" Page 337)

- Searching a string variable

  (>>> 10.14.5 "Searching a string variable" Page 338)

- Comparing the contents of string variables

  (>>> 10.14.6 "Comparing the contents of string variables" Page 338)

- Copying a string variable

  (>>> 10.14.7 "Copying a string variable" Page 339)

10.14.1 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

10.14.2 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

10.14.3 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.

10.14.4 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

```
1  DECL CHAR A[50], B[50]
2  INT AB, AC
   …
3  A[] = "This is an "
4  B[] = "example"
5  AB = StrAdd(A[],B[])
```

Line	Description
5	A[ ] = “This is an example” AB = 18

10.14.5 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

10.14.6 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[].

Element	Description
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

10.14.7 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

```
1  DECL CHAR A[25], B[25]
2  DECL BOOL C
3  A[] = ""
4  B[] = "Example"
5  C = StrCopy(A[], B[])
```

Line	Description
5	A[ ] = “Example” C = TRUE

11 Submit interpreter

11.1 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.

(>>> 11.2 "Manually stopping or deselecting the Submit interpreter" Page 342)

(>>> 11.3 "Manually starting the Submit interpreter" Page 342)

SUB programs are always files with the extension *.SUB. The program SPS.SUB can be modified and new SUB programs can be created.

(>>> 11.4 "Modifying the program SPS.SUB" Page 342)

(>>> 11.5 "Creating a new SUB program" Page 343)

Caution!
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.

11.2 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
	Red	Submit interpreter has been stopped.
	Gray	Submit interpreter is deselected.

11.3 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
	Green	Submit interpreter is running.

11.4 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

11.5 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.

11.6 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.

Motion commands for external axes can be used in a SUB program. Example:

```
IF (($IN[12] == TRUE) AND ( NOT $IN[13] == TRUE)) THEN
ASYPTP {E2 45}
ASYPTP {E3 200}
...
IF ((NOT $IN[12] == TRUE) AND  ($IN[13] == TRUE)) THEN
ASYPTP {E2 0}
ASYPTP {E3 90}
```

External axes E2 and E3 are moved in accordance with specific inputs.

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.

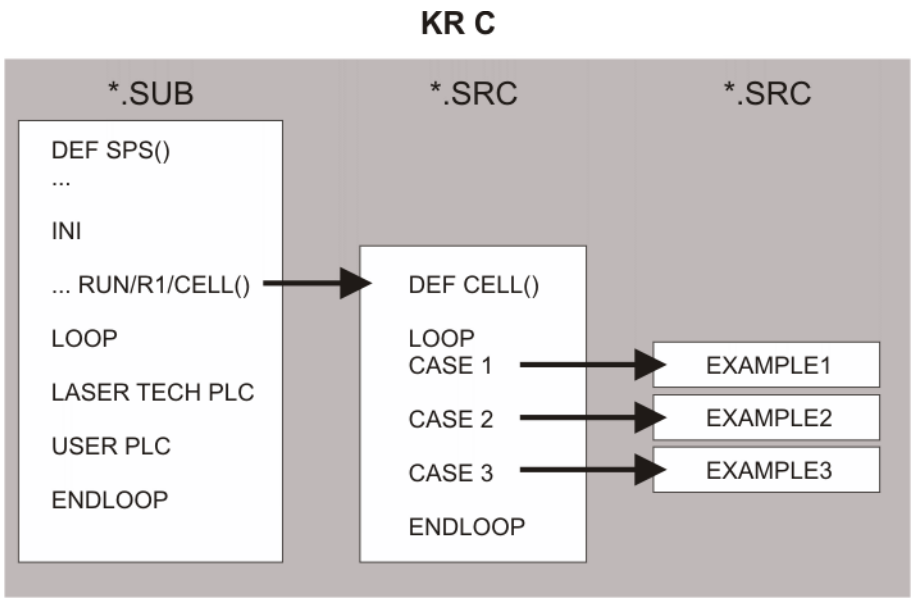


Fig. 11-1: SPS.SUB calls CELL.SRC

Further information about the program CELL.SRC can be found in this documentation.
(>>> 6.20.1 "Configuring CELL.SRC" Page 154)
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.

12 Diagnosis

12.1 Logbook

12.1.1 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 (>>> 12.1.2 ""Log" tab" Page 347)
- Filter (>>> 12.1.3 ""Filter" tab" Page 348)

12.1.2 "Log" tab

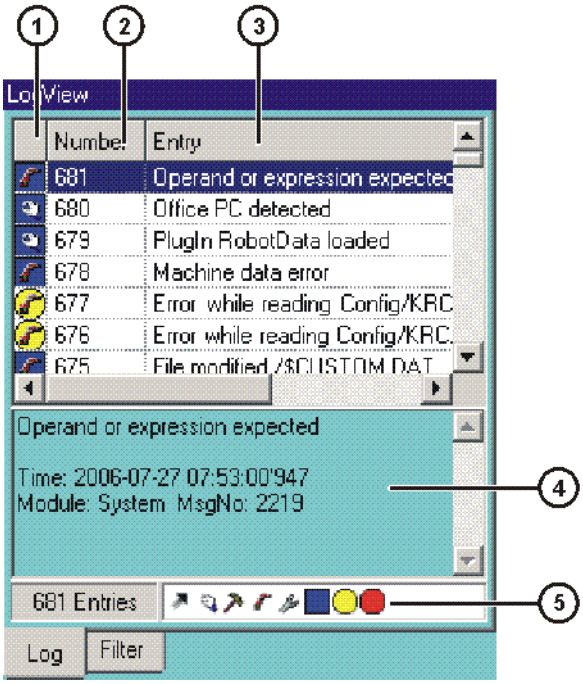


Fig. 12-1: Logbook, Log tab

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.

12.1.3 "Filter" tab

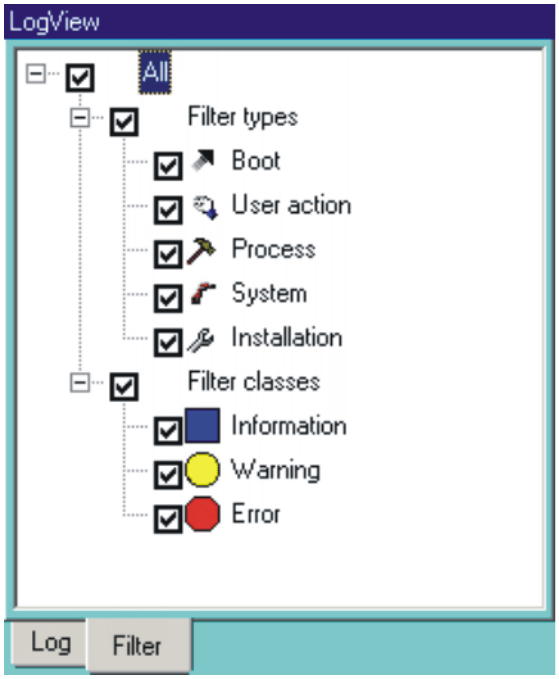


Fig. 12-2: Logbook, Filter tab

The following softkeys are available:

Softkey	Description
Tab +	Toggles between the **Log** and **Filter** tabs.
Mark	Activates or deactivates the selected filter.

12.2 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

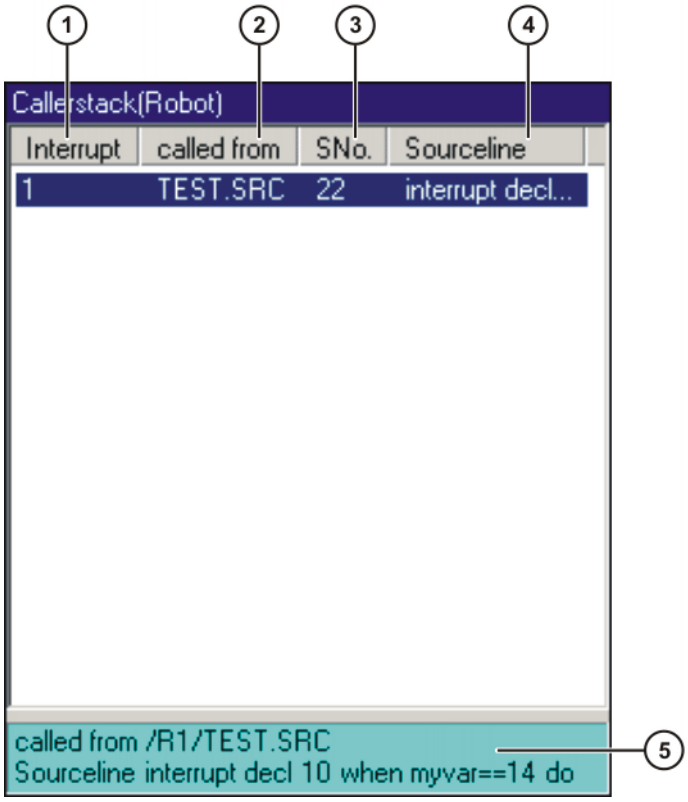


Fig. 12-3: Caller Stack window

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

12.3 Displaying interrupts

Precondition
- User group “Expert”

Procedure
- Select the menu sequence **Monitor** > **Diagnosis** > **Interrupts**.

Description

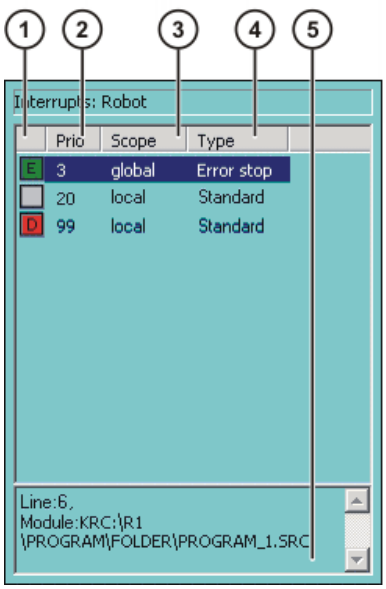


Fig. 12-4: Interrupts

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.

12.4 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.

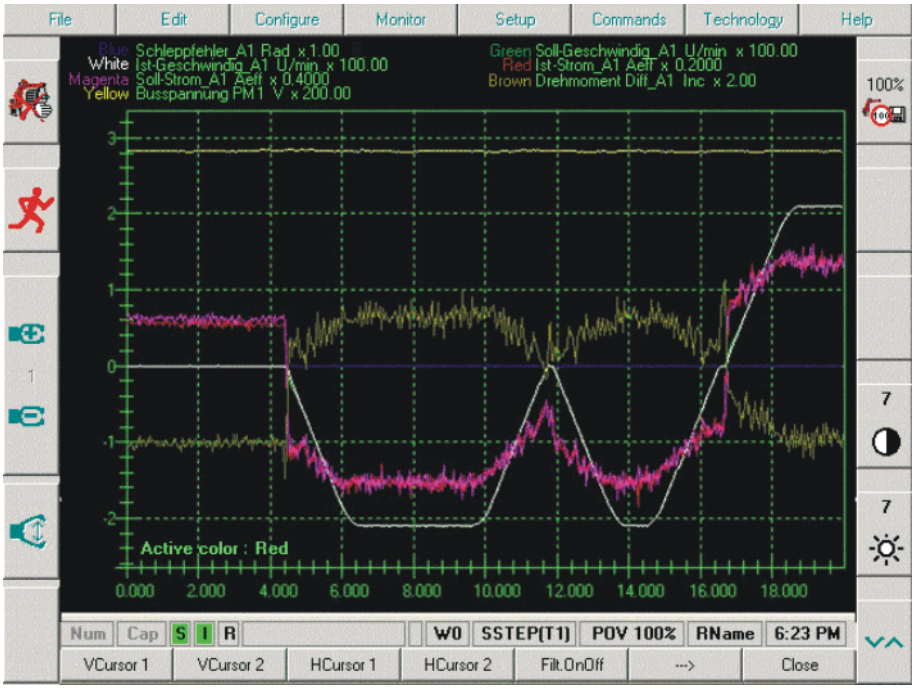


Fig. 12-5: Oscilloscope

12.4.1 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.

   (>>> 12.4.1.1 "Main window" Page 352)

3. If DSE data are to be recorded:
   Press the **DSE Table** softkey and configure the DSE recording.

   (>>> 12.4.1.2 "DSE table" Page 354)

4. If I/O data are to be recorded:
   Press the **I/O Table** softkey and configure the I/O recording.

   (>>> 12.4.1.3 "I/O table" Page 356)

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.

12.4.1.1 Main window

Description

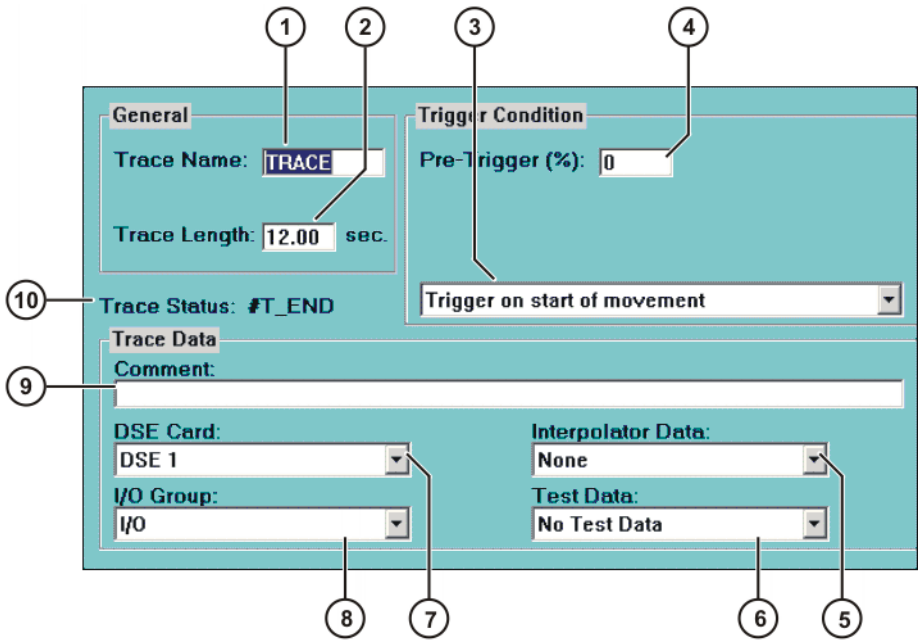


Fig. 12-6: Oscilloscope configuration – main window

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 353) 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.
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.

Item	Description
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.
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	Description
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.

12.4.1.2 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.

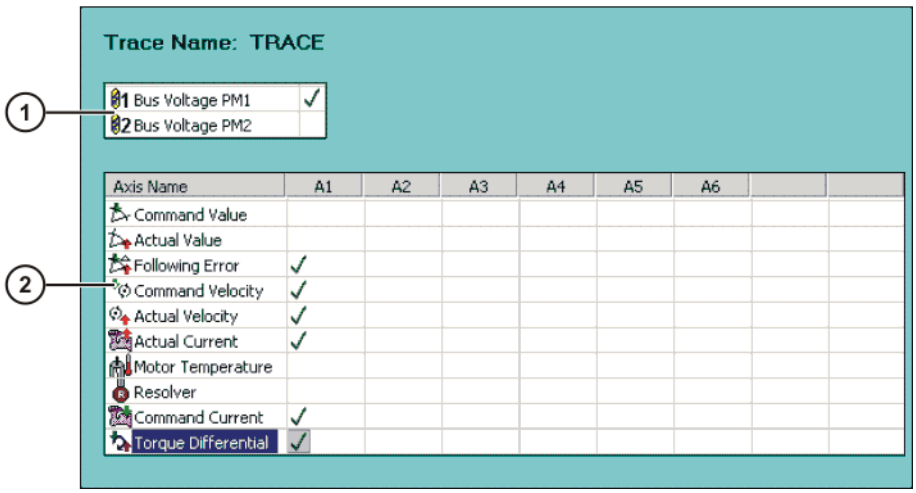


Fig. 12-7: Oscilloscope configuration – DSE table

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.

12.4.1.3 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.

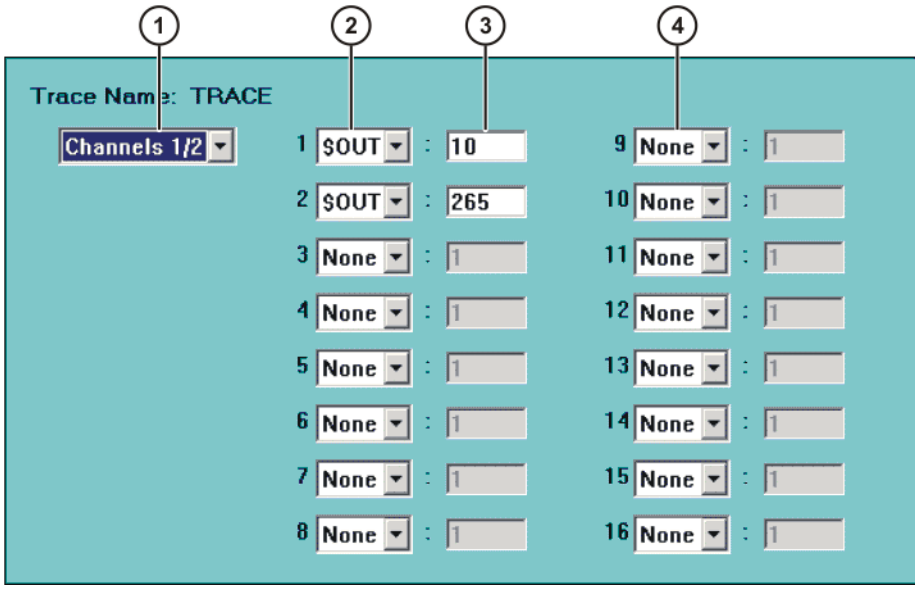


Fig. 12-8: Example: 2 outputs are recorded, 1 channel = 1 color

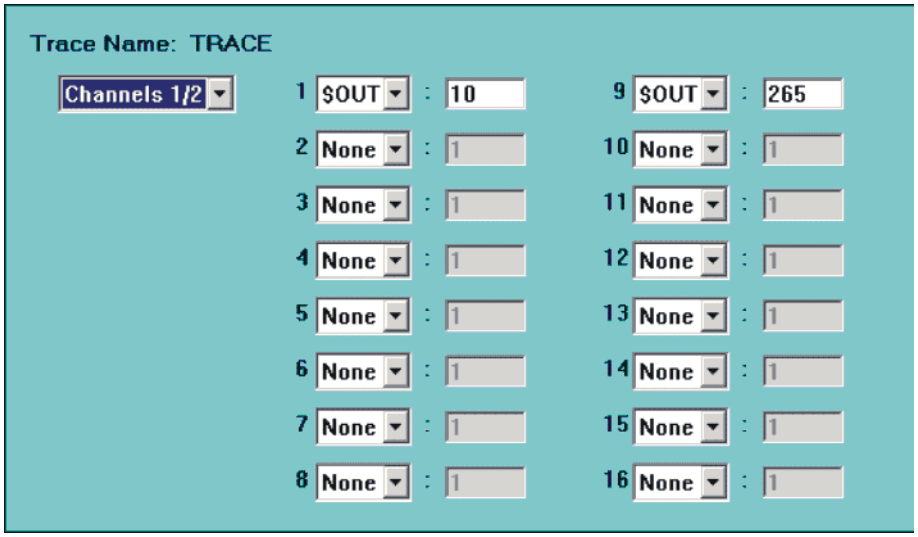


Fig. 12-9: Example: 2 outputs are recorded, 2 channels = 2 colors

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**

Item	Description
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.

12.4.1.4 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.

Line	Description
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.

12.4.1.5 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:

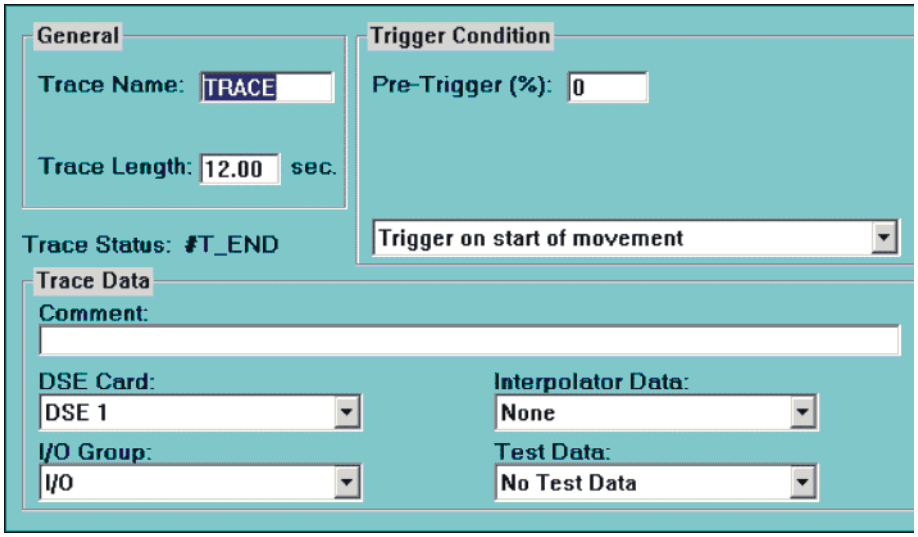


Fig. 12-10: Example 1, main window

DSE table

Make the following settings in the DSE table:

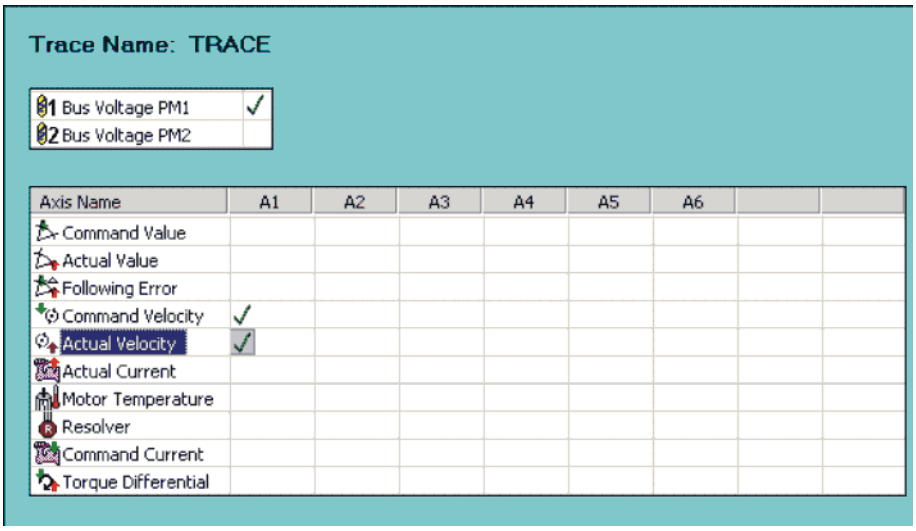


Fig. 12-11: Example 1, DSE table

I/O table

Make the following settings in the I/O table:

The outputs need not be specified in numerical order.

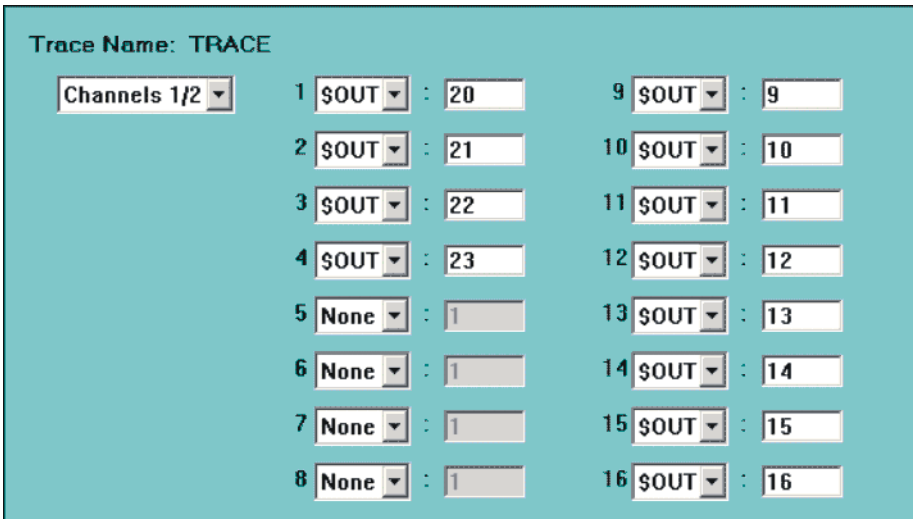


Fig. 12-12: Example 1, I/O table

12.4.1.6 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:

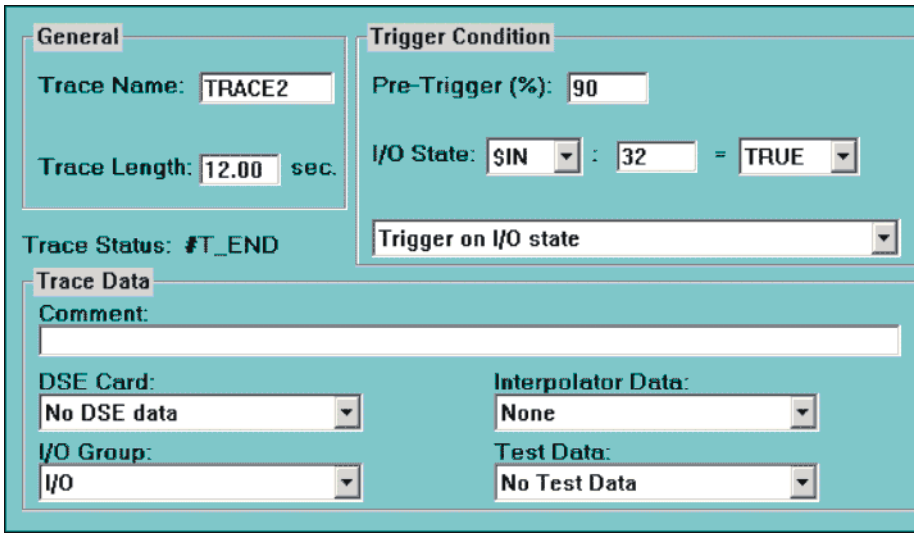


Fig. 12-13: Example 2, main window

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:

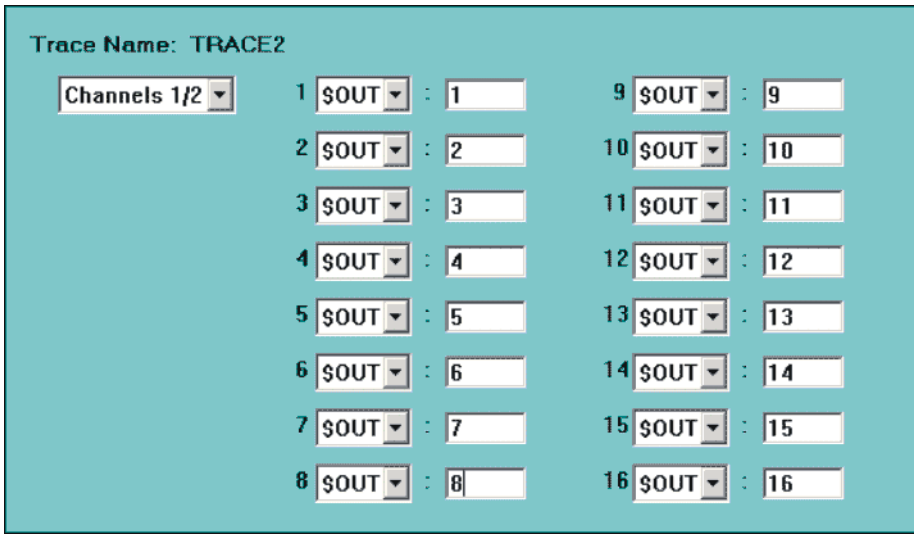


Fig. 12-14: Example 2, I/O table (outputs)

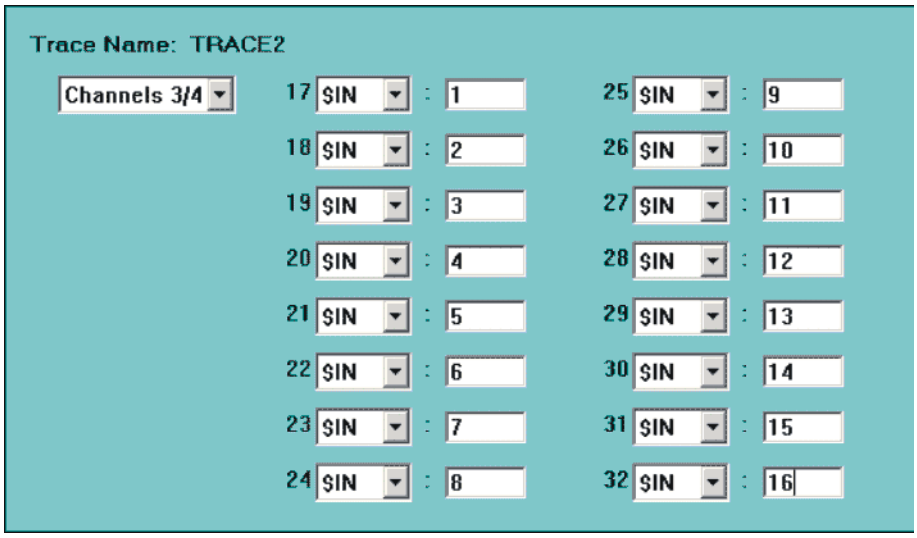


Fig. 12-15: Example 2, I/O table (inputs)

12.4.1.7 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.
   (>>> 12.4.1.4 "Starting the recording via a program" Page 357)
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:

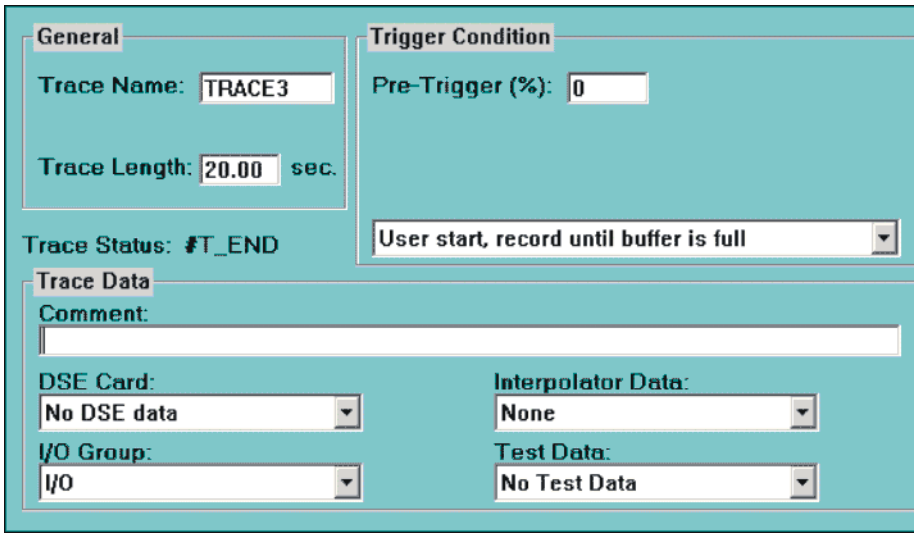


Fig. 12-16: Example 3, main window

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.

12.4.2 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.

12.4.2.1 User interface

Recording

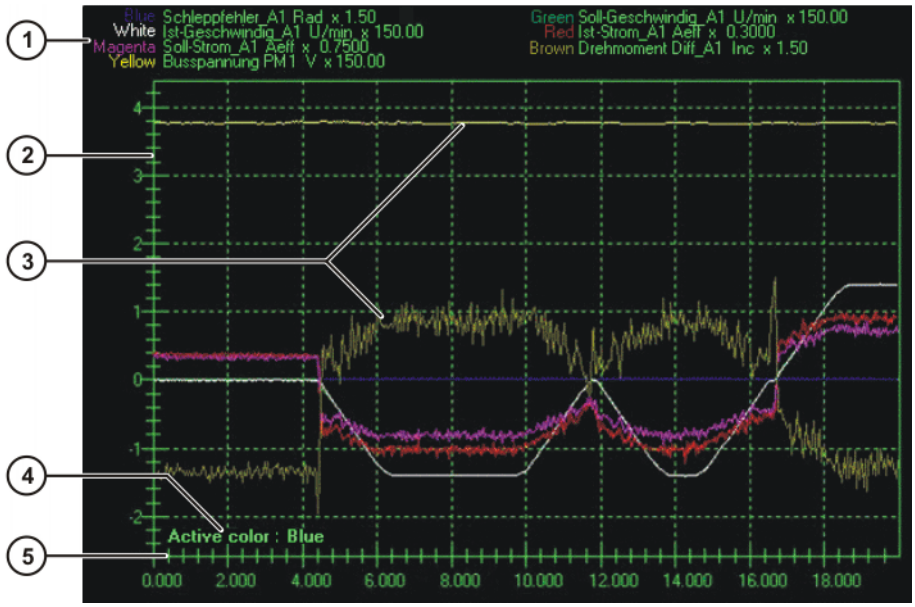


Fig. 12-17: Example of a trace recording

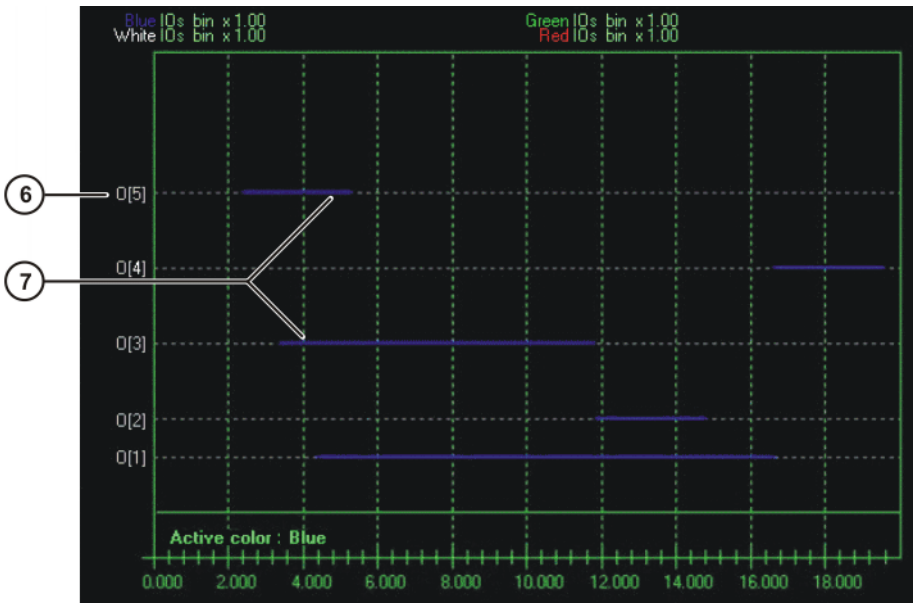


Fig. 12-18: Example of a trace recording of outputs

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.

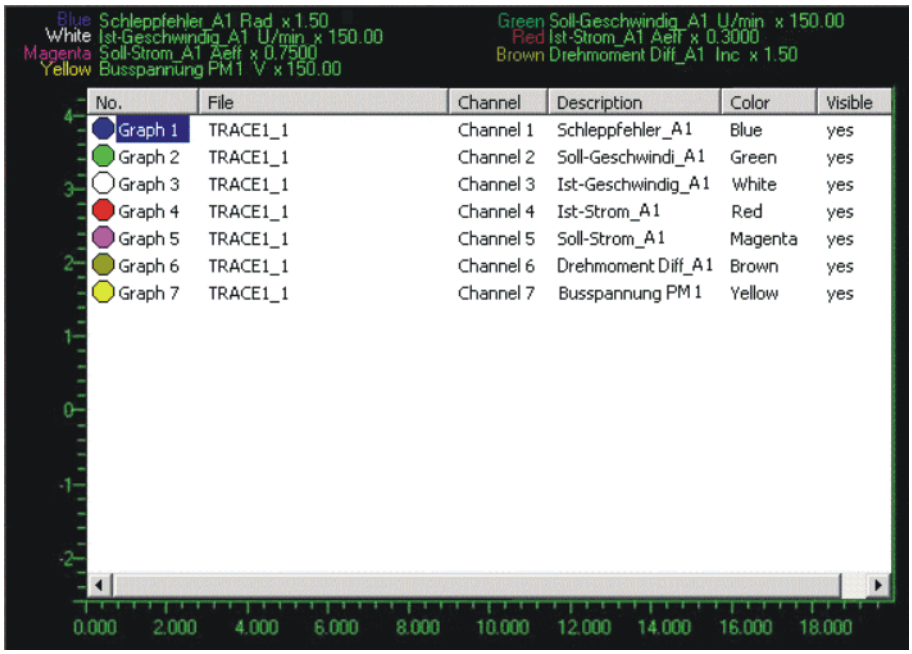


Fig. 12-19: Info window

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. (>>> 12.4.2.7 "Filtering the display" Page 370)	**Filt.OnOff**
E	Activates or deactivates the RMS function. (>>> 12.4.2.8 "Determining the r.m.s. value" Page 371)	**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. (>>> 12.4.2.10 "Saving the display as a BMP file" Page 375)	**Save**
Z	Activates the Zoom function. (>>> 12.4.2.6 "Enlarging the display" Page 368)	**Zoom**
U	Deactivates the Zoom function.	**Unzoom**
C	Keys for the cursor functions (>>> 12.4.2.9 "Cursor functions" Page 373)	**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. (>>> 12.4.2.2 "Superimposing traces" Page 367)
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. (>>> 12.4.2.6 "Enlarging the display" Page 368)
Unzoom	Deactivates the Zoom function.
Save	Saves the current oscilloscope display as a BMP file in the directory C:\KRC\ROBOTER\TRACE. (>>> 12.4.2.10 "Saving the display as a BMP file" Page 375)
RMS	Activates or deactivates the RMS function. (>>> 12.4.2.8 "Determining the r.m.s. value" Page 371)

Softkey	Description
Print	Prints the display. (>>> 12.4.2.11 "Printing the display" Page 375)
-->	Displays softkey bar 4.
Close	Closes the oscilloscope.

Softkey bar 4:

Softkey	Description
VCursor 1	Softkeys for the cursor functions (>>> 12.4.2.9 "Cursor functions" Page 373)
VCursor 2	
HCursor 1	
HCursor 2	
Filt.OnOff	Switches the filter on or off. (>>> 12.4.2.7 "Filtering the display" Page 370)
-->	Displays softkey bar 1.
Close	Closes the oscilloscope.

12.4.2.2 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.

12.4.2.3 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.

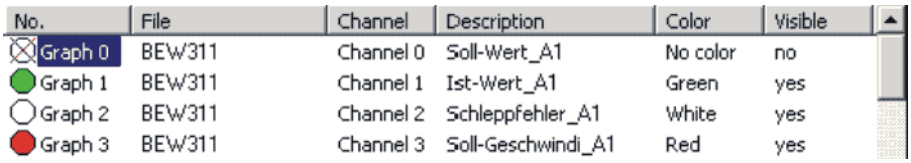
Alternative procedure

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. 12-20: 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. 12-21: Curve is deactivated

4. Press **I** again. The Info window is closed.
5. To reactivate the curve, repeat steps 1 to 4.

12.4.2.4 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.

12.4.2.5 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.

12.4.2.6 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**.

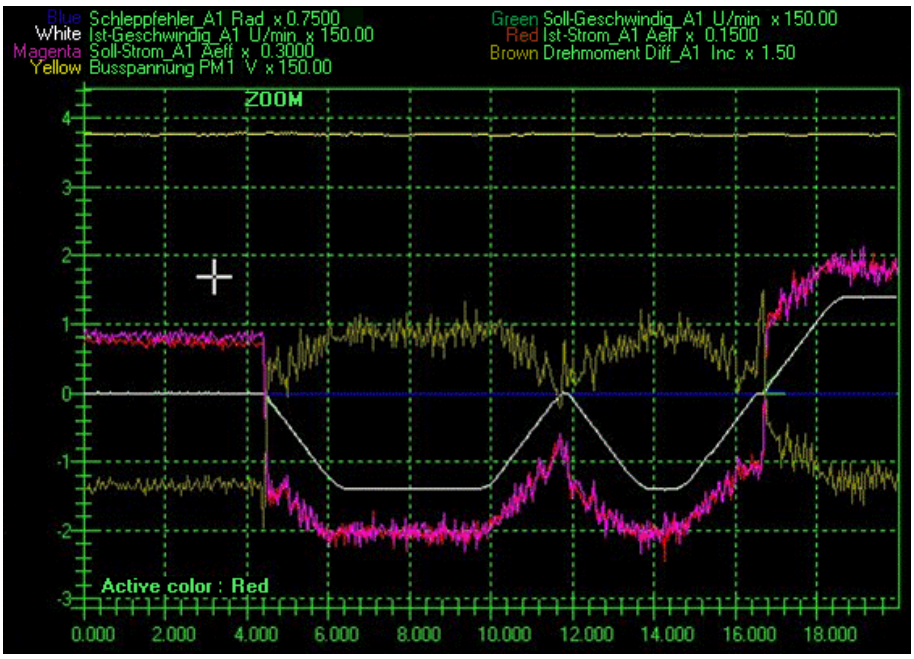


Fig. 12-22: Cross is displayed

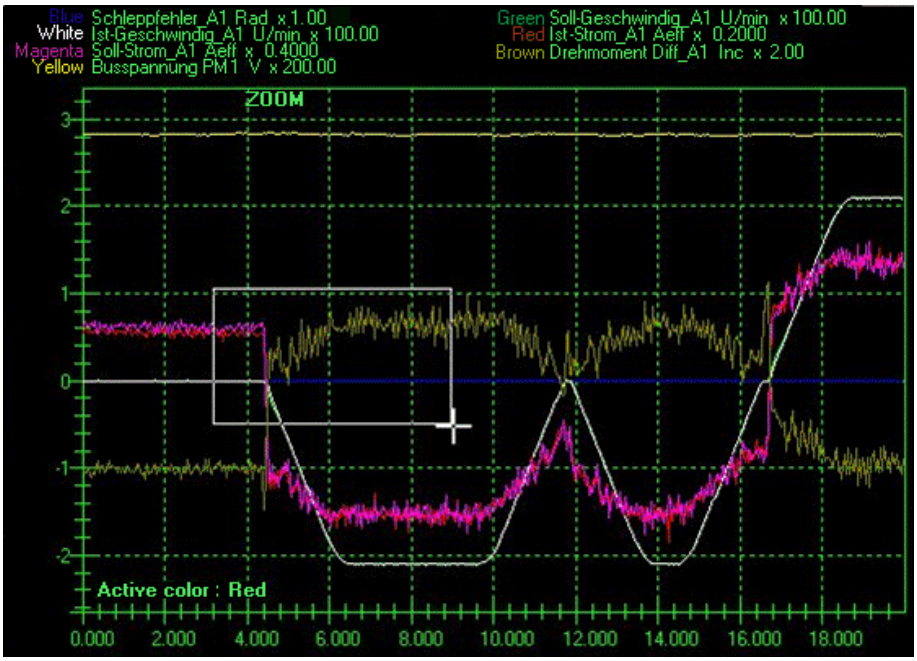


Fig. 12-23: Window has been stretched

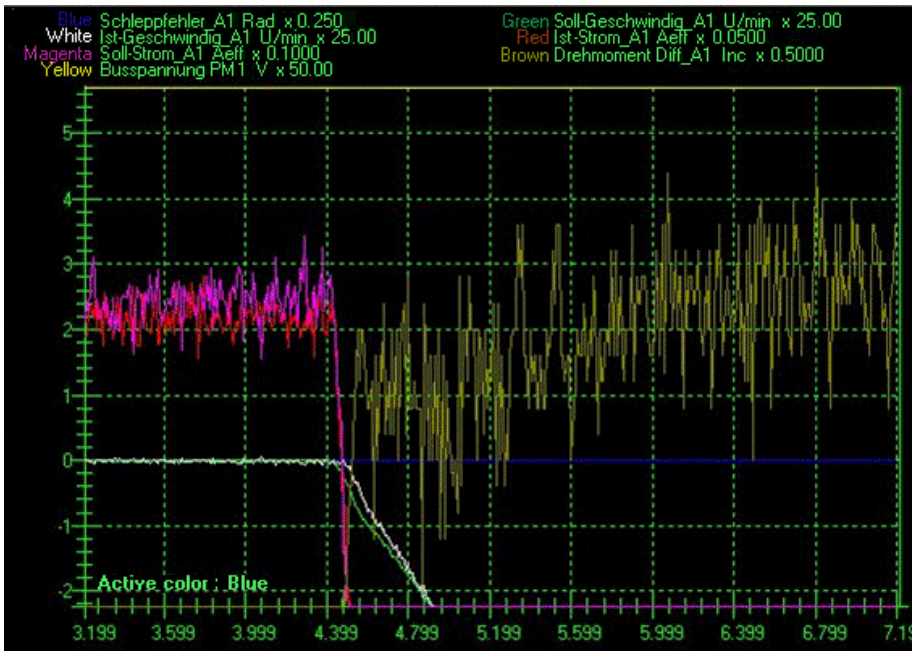


Fig. 12-24: Contents of the window are enlarged

12.4.2.7 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.

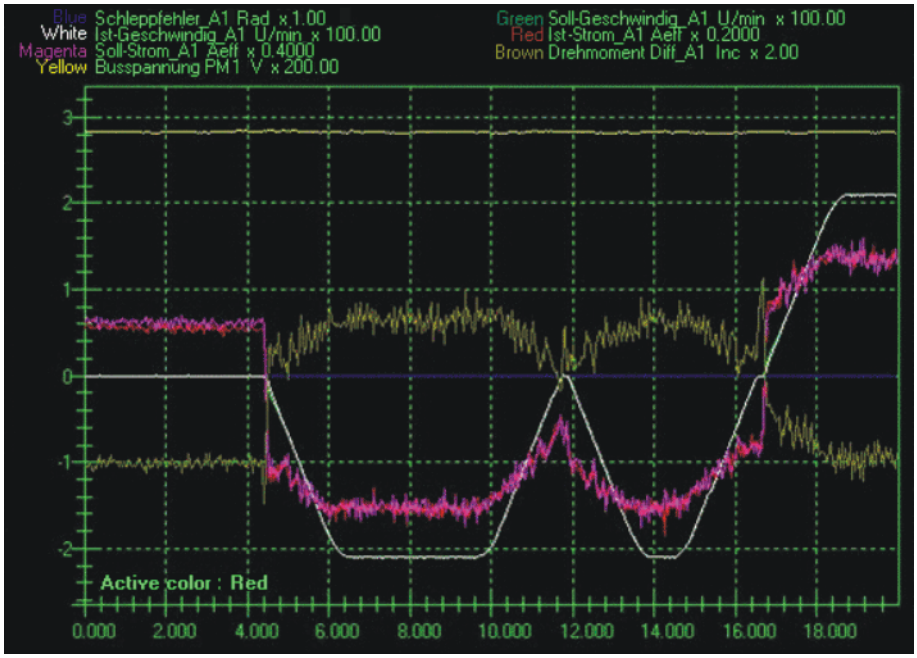


Fig. 12-25: Filter on

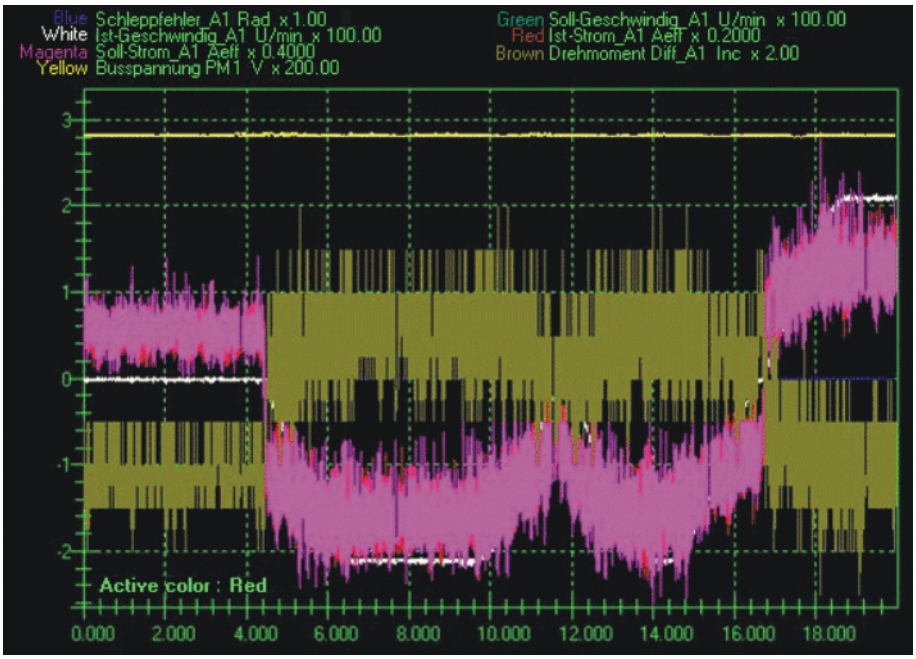


Fig. 12-26: Filter off

12.4.2.8 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.

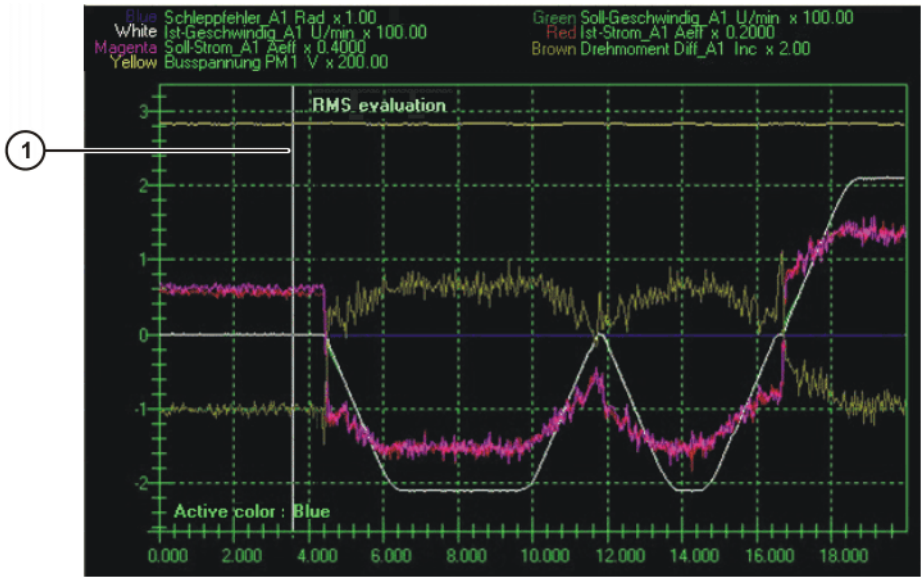


Fig. 12-27: Vertical line is displayed

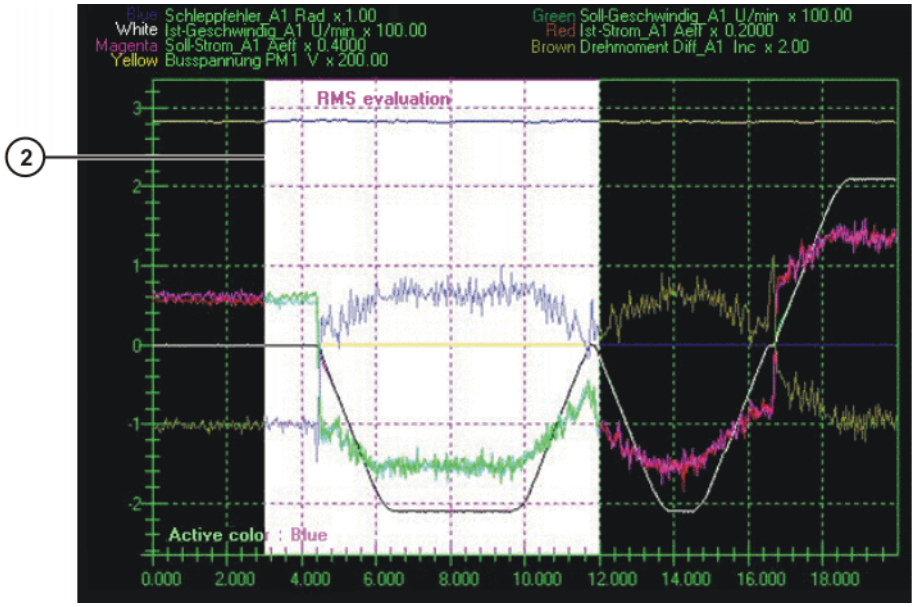


Fig. 12-28: Line has been stretched to form a window

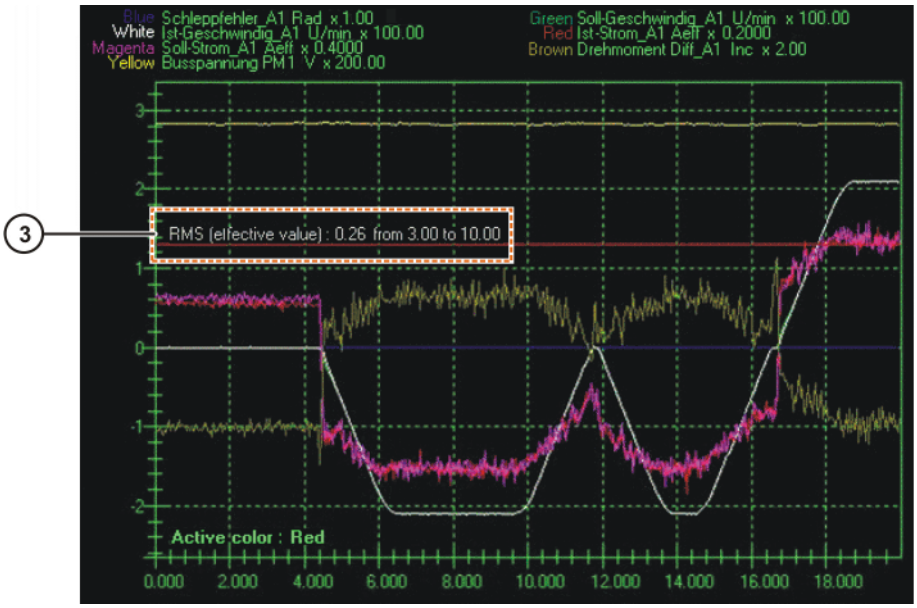


Fig. 12-29: Result is displayed

Item	Description
1	Vertical line
2	Window
3	Result

12.4.2.9 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.

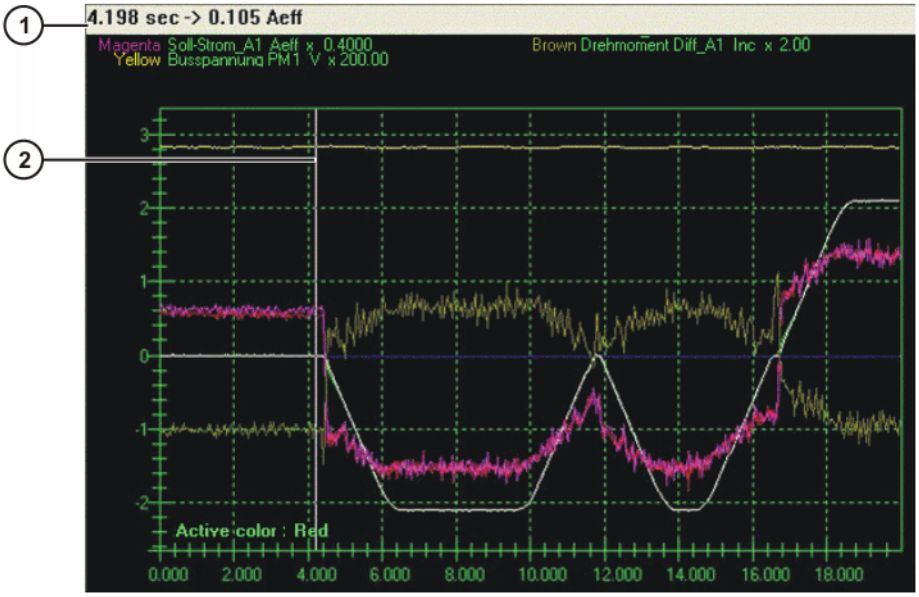


Fig. 12-30: First vertical line

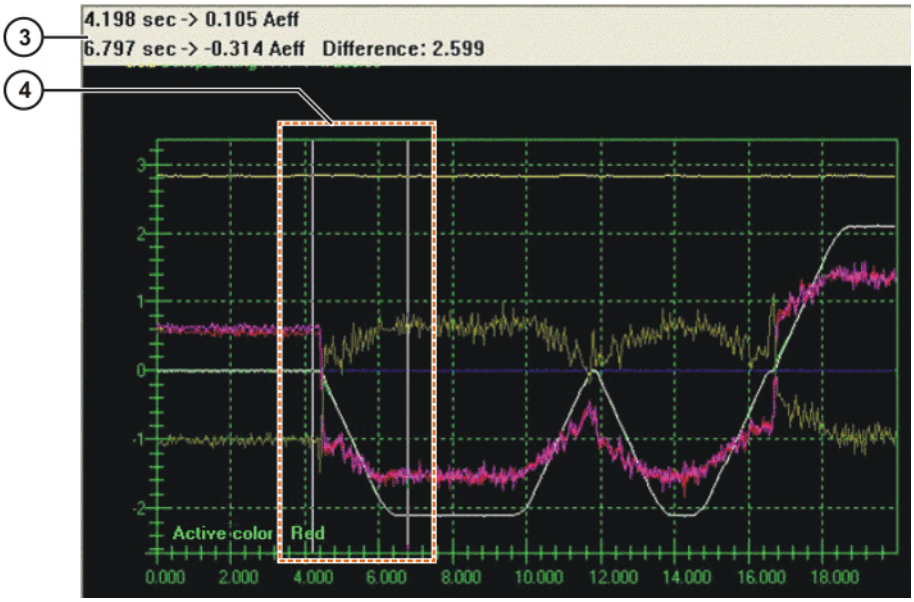


Fig. 12-31: First and second vertical lines

Fig. 12-32: 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

12.4.2.10 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**.

12.4.2.11 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.

13 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.
(>>> 15 "KUKA Service" Page 387)

13.1 Overview of the software components

Overview

The following software components are used:

- KUKA System Software 5.5
- Windows XP embedded 2.x incl. Service Pack 2

It is not possible to upgrade from Windows Service Pack 1 to Windows Service Pack 2.

13.2 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. (>>> 13.2.1 "Adapting BIOS settings for USB CD/DVD drive" Page 377) 2. Install Windows. (>>> 13.2.2 "Installing Windows" Page 378) 3. If required, change the Windows language. (>>> 13.2.3 "Changing the Windows language" Page 378)
2	Install the KUKA System Software. (>>> 13.2.4 "Installing the KUKA System Software" Page 379)
3	Install technology packages. (>>> 13.3 "Installing additional software (via KUKA.HMI)" Page 379)
4	If applicable: load archive. (>>> 7.8.4 "Restoring data" Page 230)

13.2.1 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.

13.2.2 Installing Windows

Warning!
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.

13.2.3 Changing the Windows language

Description

By default, the Windows user interface can be displayed in German or English. Additional languages can be installed.

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.

13.2.4 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.

Caution!
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.

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.

13.3 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 instal-

lation. If a software package that the system has rejected is nonetheless to be installed, KUKA Roboter GmbH must be contacted.

(>>> 15.2 "KUKA Customer Support" Page 387)

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.

(>>> 13.4 "KSS update (via KUKA.HMI)" Page 381)

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.

13.4 KSS update (via KUKA.HMI)

Description

This function can be used to install KSS updates, e.g. from KSS 5.5.3 to KSS 5.5.4.

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 5.4 to KSS 5.5. KUKA Roboter GmbH must be consulted before a new version is installed.

(>>> 15 "KUKA Service" Page 387)

This function cannot be used to install updates of additional software, such as technology packages.

(>>> 13.3 "Installing additional software (via KUKA.HMI)" Page 379)

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 from the CD with the KSS update 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

(>>> 13.4.2 "KSS update from CD-ROM" Page 383)

- From the network

(>>> 13.4.3 "KSS update from the network" Page 383)

13.4.1 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)
```

13.4.2 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.

13.4.3 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

The network path from which the installation is to be carried out is configured as follows:

It is only necessary to configure the network path once. It remains saved for subsequent updates.

1. Select the menu sequence **Setup** > **Install Additional Software**.
2. Press the **New SW** softkey.
3. Press the **Config.** softkey.

4. Select the **Installationpath for KRC Update from network** box. Press the **Browse** softkey.
5. Select the desired network path. Press the **Apply** softkey.
6. The selected path is displayed in the **Installationpath for KRC Update from network** box. Press the **Apply** softkey.
7. Press the **Close** softkey.

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.

14 Messages

14.1 System messages

Details about the system messages can be found in the online help.

(>>> 4.2.5 "Calling online help" Page 51)

14.2 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.
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”.

15 KUKA Service

15.1 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.

Faults leading to production downtime should be reported to the local KUKA subsidiary within one hour of their occurrence.

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

15.2 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

Marand Precision Engineering Pty. Ltd. (Agency)
153 Keys Road
Moorabbin
Victoria 31 89
Australia
Tel. +61 3 8552-0600
Fax +61 3 8552-0605
robotics@marand.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 Flexible Manufacturing Equipment (Shanghai) Co., Ltd.
Shanghai Qingpu Industrial Zone
No. 502 Tianying Rd.
201712 Shanghai
P.R. China
Tel. +86 21 5922-8652
Fax +86 21 5922-8538
Franz.Poeckl@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, Private Limited
621 Galleria Towers
DLF Phase IV
122 002 Gurgaon
Haryana
India
Tel. +91 124 4148574
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 Robot Automation Korea Co. Ltd.
4 Ba 806 Sihwa Ind. Complex
Sung-Gok Dong, Ansan City
Kyunggi Do
425-110
Korea
Tel. +82 31 496-9937 or -9938
Fax +82 31 496-9939
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
Riedstr. 7
8953 Dietikon
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.
136, Section 2, Huanjung E. Road
Jungli City, Taoyuan
Taiwan 320
Tel. +886 3 4371902
Fax +886 3 2830023
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

T

U

V

W

X