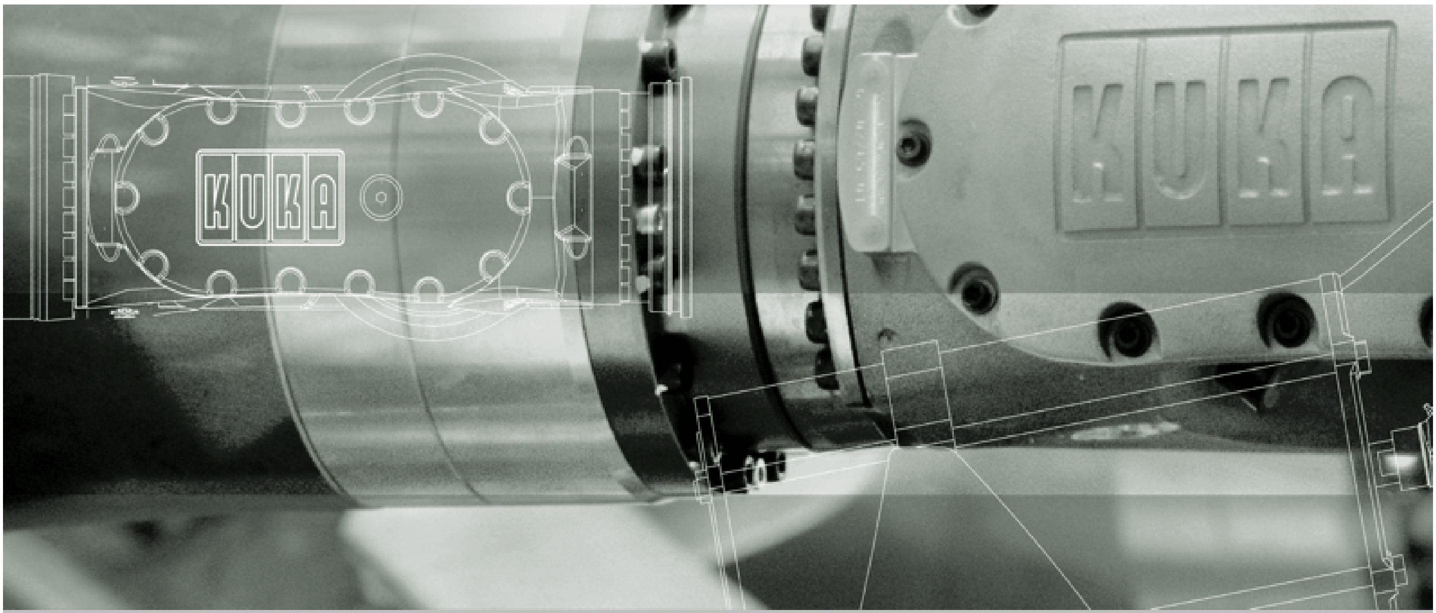


KUKA Robot Group

KUKA System Software (KSS)

KUKA System Software 5.2, 5.3, 5.4

Operating and Programming Instructions for Systems Integrators

Issued: 26.02.2007 Version: 0.3

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.

KIM-PS4-DOC

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 Robot system documentation

The robot system documentation consists of the following parts:

- Operating instructions for the robot
- Operating instructions for the robot controller
- Operating and programming instructions for the KUKA System Software
- Documentation relating to options and accessories

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.

2 Product description

2.1 Overview of the robot system

A robot system consists of the following components:

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

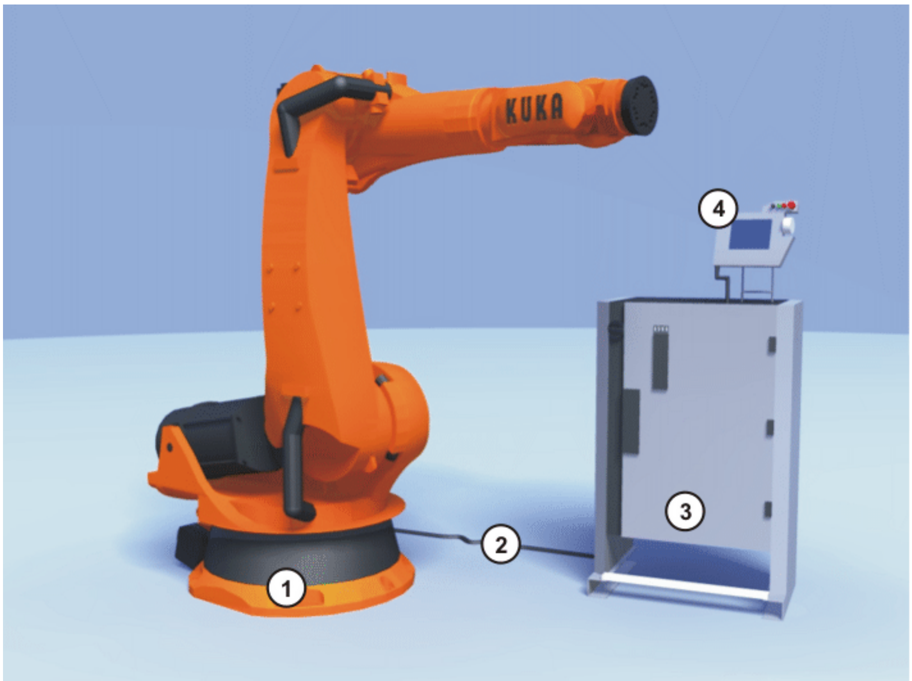


Fig. 2-1: Example of a robot system

1	Robot	3	Robot controller
2	Connecting cables	4	Teach pendant (KCP)

2.2 Overview of the software components

Overview

The following software components are used:

- KUKA System Software KSS V5.x
- Windows XP embedded
- incl. Windows Service Pack 1.0
- VxWin RT V3.0

It is not possible to upgrade from Windows Service Pack 1.0 to Windows Service Pack 2.0.

2.3 Overview of KUKA System Software (KSS)

Description

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

- 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.

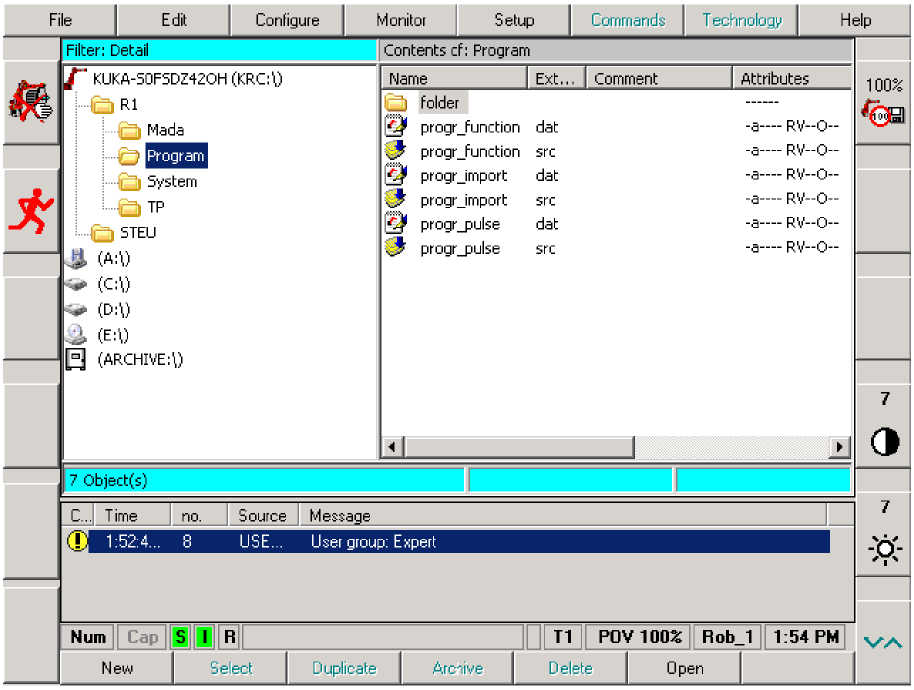


Fig. 2-2: KUKA.HMI user interface

3 Safety

3.1 Stop reactions

Stop reactions of the robot system 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 designations defined in EN 60204.

Trigger	T1, T2	AUT, AUT EXT
EMERGENCY STOP	Path-oriented braking (STOP 0)	Path-maintaining braking (STOP 1)
Start key released	Ramp-down braking (STOP 2)	-
Enabling switch released	Path-oriented braking (STOP 0)	-
Safety gate opened	-	Path-maintaining braking (STOP 1)
"Drives OFF" key pressed	Path-oriented braking (STOP 0)	
Operating mode change	Path-oriented braking (STOP 0)	
Encoder error (DSE-RDC connection broken)	Short-circuit braking	
Motion enable canceled	Ramp-down braking (STOP 2)	
STOP key pressed	Ramp-down braking (STOP 2)	
Controller shut down Power failure	Short-circuit braking	

Stop reaction	Drives	Brakes	Software
Ramp-down braking (STOP 2)	Remain on.	Remain open.	Normal ramp which is used for acceleration and deceleration.
Path-maintaining braking (STOP 1)	Switched off after 1 second hardware delay.	Applied after 1 s at latest.	In this time the controller brakes the robot on the path using a steeper stop ramp.
Path-oriented braking (STOP 0)	Switched off immediately.	Applied immediately.	The controller attempts to brake the robot on the path with the remaining energy. If the voltage is not sufficient, the robot leaves the programmed path.
Short-circuit braking	Switched off immediately.	Applied immediately.	-

3.2 Labeling on the robot system

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

Labeling on the robot system consists of:

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

3.3 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 robot system will not cause personal injuries or material damage.

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

3.4 System planning

3.4.1 EC declaration of conformity and declaration of incorporation

EC declaration of conformity

The system integrator must issue a declaration of conformity for the overall system in accordance with Directive 98/37/EC (Machinery Directive). The declaration of conformity forms the basis for the CE mark for the system. The robot system must be operated in accordance with the applicable national laws, regulations and standards.

The robot controller has a CE mark in accordance with Directive 89/336/EEC (EMC Directive) and Directive 73/23/EEC (Low Voltage Directive).

Declaration of incorporation

A declaration of incorporation is provided for the robot system. This declaration of incorporation contains the stipulation that the robot system must not be commissioned until it complies with the provisions of 98/37/EC (Machinery Directive).

3.4.2 Installation site

Robot

When planning the system, it must be ensured that the installation site (floor, wall, ceiling) has the required grade of concrete and load-bearing capacity. The principal loads acting on the mounting base are indicated in the specifications.

Further information is contained in the robot operating instructions.

Robot controller

It is imperative to comply with the minimum clearances of the robot controller from walls, cabinets and other system components.

Further information is contained in the robot controller operating instructions.

3.4.3 Simulation

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

3.4.4 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 danger zone consists of the workspace and the braking distances of the robot. It must be safeguarded by means of protective barriers to prevent danger to persons or the risk of material damage.

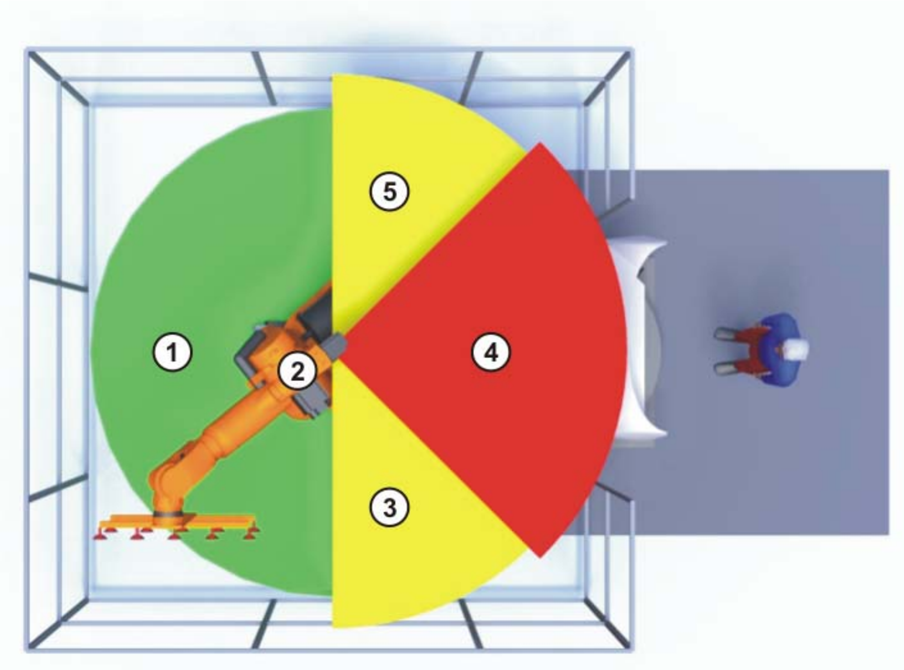


Fig. 3-1: Example of axis range A1

1	Workspace	4	Safety zone
2	Robot	5	Braking distance
3	Braking distance		

3.4.5 External safeguards

EMERGENCY STOP

Additional Emergency Stop devices can be connected via interface X11 or linked together by means of higher-level controllers (e.g. PLC).

The input/output signals and any necessary external power supplies must ensure a safe state in the case of an Emergency Stop.

Further information is contained in the robot controller operating instructions.

Safety fences

Requirements on safety fences are:

- Safety fences must withstand all forces that are likely to occur in the course of operation, whether from inside or outside the enclosure.
- Safety fences must not, themselves, constitute a hazard.
- It is imperative to comply with the minimum clearances from the danger zone.

Further information is contained in the corresponding standards and regulations.

Safety gates

Requirements on safety gates are:

- The number of safety gates in the fencing must be kept to a minimum.
- All safety gates must be safeguarded by means of an operator safety system (interface X11).
- Automatic mode must be prevented until all safety gates are closed.
- In Automatic mode, the safety gate can be mechanically locked by means of a safety system.
- If the safety gate is opened in Automatic mode, it must trigger an Emergency Stop function.
- If the safety gate is closed, the robot cannot be started immediately in Automatic mode. The message on the control panel must be acknowledged.

Further information is contained in the corresponding standards and regulations.

Other safety equipment

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

3.5 Safety features of the robot system

3.5.1 Overview of the safety features

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

Safety features	T1	T2	AUT	AUT EXT
Operator safety	-	-	active	active
Emergency Stop button (STOP 0)	active	active	-	-
Emergency Stop button (STOP 1)	-	-	active	active
Enabling switch	active	active	-	-
Reduced velocity	active	-	-	-
Jog mode	active	active	-	-
Software limit switches	active	active	active	active

Danger!
In the absence of functional safety equipment, the robot can cause personal injury or material damage. No safety equipment may be dismantled or deactivated while the robot is in operation.

3.5.2 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 robot system to a standstill.

The ESC safety logic monitors the following inputs:

- Local EMERGENCY STOP
- External EMERGENCY STOP
- Operator safety
- Enabling
- Drives OFF
- Drives ON
- Operating modes
- Qualifying inputs

Further information is contained in the robot controller operating instructions.

3.5.3 Operator safety input

The operator safety input is used for interlocking fixed guards. Safety equipment, such as safety gates, can be connected to the dual-channel input. If nothing is connected to this input, operation in Automatic mode is not possible. Operator safety is not active for test modes T1 and T2.

In the event of a loss of signal during Automatic operation (e.g. safety gate is opened), the drives are deactivated after 1 s and the robot stops with a STOP 1. Once the signal is active at the input again (e.g. safety gate closed and signal acknowledged), Automatic operation can be resumed.

Operator safety can be connected via interface X11.

Further information is contained in the robot controller operating instructions.

3.5.4 Connection for external enabling switch

An external enabling switch is required if there is more than one person in the danger zone.

The external enabling switch can be connected via interface X11.

An external enabling switch is not included in the scope of supply of KUKA Roboter GmbH.

Further information is contained in the robot controller operating instructions.

3.5.5 EMERGENCY STOP button

The EMERGENCY STOP button for the robot system is located on the KCP. If the EMERGENCY STOP button is pressed, the drives are deactivated immediately in operating modes T1 and T2 and the robot stops with a STOP 0. In the Automatic operating modes, the drives are deactivated after 1 s and the robot stops with a STOP 1. The EMERGENCY STOP button must be pressed as soon as persons or equipment are endangered. Before operation can be resumed, the EMERGENCY STOP button must be turned to release it and the error message must be acknowledged.

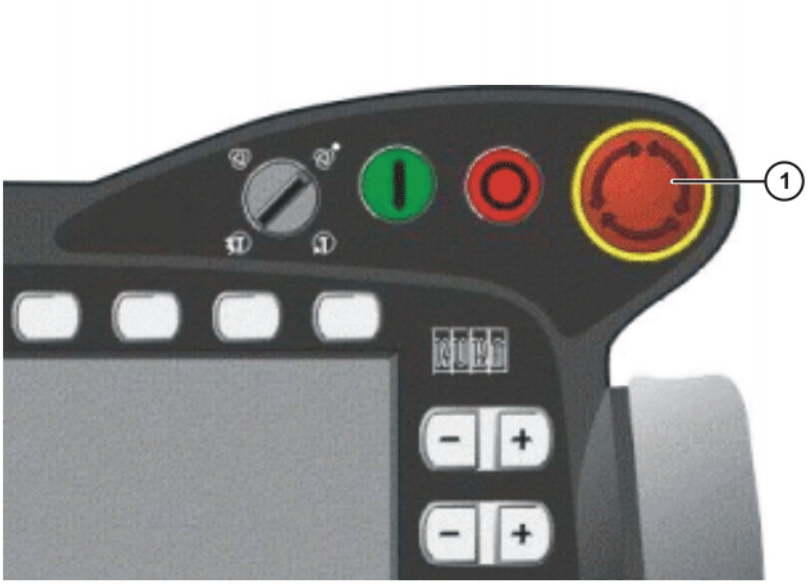


Fig. 3-2: EMERGENCY STOP button on the KCP

1	EMERGENCY STOP button

3.5.6 Enabling switches

There are 3 enabling switches installed on the KCP. These 3-position enabling switches can be used to switch on the drives in modes T1 and T2.

In the test modes, the robot 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 robot stops with a STOP 0.

Fig. 3-3: Enabling switches on the KCP

1 - 3 Enabling switches

3.5.7 Mode selector switch

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 deactivated immediately and the robot stops with a STOP 0.

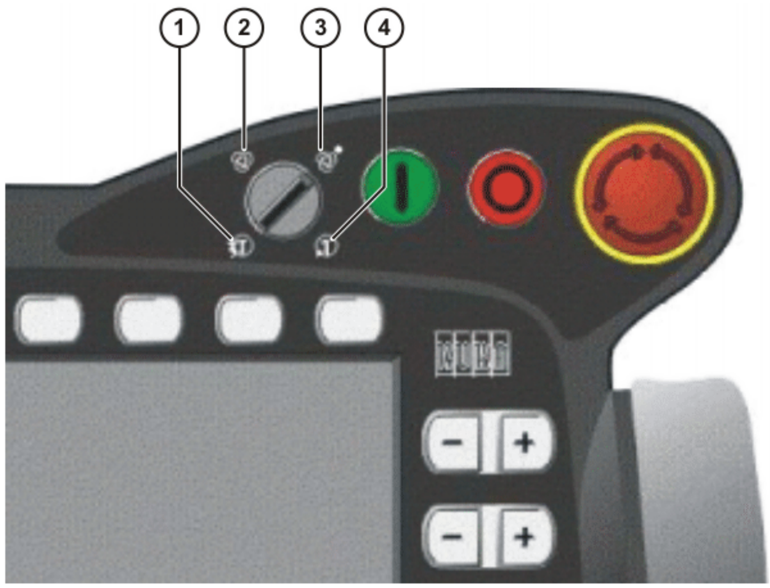


Fig. 3-4: Mode selector switch

1	T2 (Test 2)	3	AUT EXT (Automatic External)
2	AUT (Automatic)	4	T1 (Test 1)

Operating mode	Use	Velocities
T1	For test operation	■ Program mode: Programmed velocity, maximum 250 mm/s ■ Jog mode: Jog velocity, maximum 250 mm/s
T2	For test operation	■ Program mode: Programmed velocity ■ Jog mode: Jog velocity, maximum 250 mm/s
AUT	For robot systems without higher-level controllers Only possible with a connected safety circuit	■ Program mode: Programmed velocity ■ Jog mode: not possible
AUT EXT	For robot systems with higher-level controllers, e.g. PLC Only possible with a connected safety circuit	■ Program mode: Programmed velocity ■ Jog mode: not possible

3.5.8 Jog mode

In modes T1 and T2, the robot can only be moved in jog mode. For this, an enabling switch and the Start key must be kept held down. If the enabling switch is released or pressed fully down (panic position), the drives are deactivated immediately and the robot stops with a STOP 0. Releasing the Start key causes the robot to be stopped with a STOP 2.

3.5.9 Mechanical end stops

The axis ranges of main axes A 1 to A 3 and wrist axis A 5 are limited by means of mechanical limit stops with a buffer.

Danger!
If the robot hits an obstruction or a buffer on the mechanical end stop or axis range limitation, this can result in material damage to the robot. The KUKA Robot Group must be consulted before the robot is put back into operation (>>> 11 "KUKA Service" page 247). The affected buffer must immediately be replaced with a new one. If a robot collides with a buffer at more than 250 mm/s, the robot must be exchanged or recommissioning must be carried out by the KUKA Robot Group.

3.5.10 Software limit switches

The axis ranges of all robot 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 robot cannot hit the mechanical limit stops.

Further information is contained in the operating and programming instructions.

3.5.11 Axis range monitoring (option)

Most robots can be fitted with dual-channel axis range monitoring systems in main axes A1 to A3. 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 can be retrofitted.

Further information is contained in the working range monitoring operating instructions.

3.5.12 Mechanical axis range limitation (option)

Most robots can be fitted with mechanical axis range limitation in main axes A1 to A3. The adjustable axis range limitation systems restrict the working range to the required minimum. This increases personal safety and protection of the system.

This option can be retrofitted.

Further information is contained in the working range limitation operating instructions.

3.5.13 Release device (option)

Description

The release device can be used to move the robot mechanically 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). After use of the release device, the affected motors must be exchanged.

Caution!
The motors reach temperatures during operation which can cause burns to the skin. Appropriate safety precautions must be taken.

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.

4. Replace the protective cap on the motor
5. Remaster all robot axes

3.5.14 KUKA.SafeRobot (option)

KUKA.SafeRobot is an option with software and hardware components.

This option may only be retrofitted after consultation with the KUKA Robot Group.

Properties

- Connection to an external safety logic
- Monitoring that can be activated using safe inputs
- Freely definable axis-specific monitoring
- Safe monitoring of axis-specific and Cartesian velocities and accelerations
- Safe standstill monitoring
- Safe stop via Electronic Safety Circuit (ESC) with safe disconnection of the drives
- Monitoring of the mastering
- Brake test

Functional principle

The robot moves within the limits that have been configured and activated. The actual position is continuously calculated and monitored against the safety parameters that have been set.

The SafeRDC monitors the robot system by means of the safety parameters that have been set. If the robot violates a monitoring limit or a safety parameter, it is stopped.

The safe inputs and outputs of the SafeRDC are of a redundant design and LOW active.

Further information is contained in the KUKA System Technology **KUKA.SafeRobot** documentation.

3.6 Personnel

User

The user of a robot system is responsible for its use. The user must ensure that it can be operated in complete safety and define all safety measures for personnel.

System integrator

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

The system integrator is responsible for the following tasks:

- Installing the robot system
- Connecting the robot system
- Implementing the required facilities
- Issuing the declaration of conformity
- Attaching the CE mark

Operator

The operator must meet the following preconditions:

- The operator must have read and understood the robot system documentation, including the safety chapter.

- The operator must be trained for the work to be carried out.
- Work on the robot system 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 dangers.

Example

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

Tasks	Operator	Programmer	Maintenance technician
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 robot		x	x
Configuration		x	x
Programming		x	x
Start-up			x
Maintenance			x
Repair			x
Shut-down			x
Transportation			x

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

3.7 Safety measures

3.7.1 General safety measures

The robot system may only be used in technically perfect condition in accordance with its designated 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 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 robot to sag. If work is to be carried out on a switched-off robot, the robot must first be moved into a position in which it is unable to move on its own, whether the payload is mounted or not. If this is not possible, the robot must be secured by appropriate means.

KCP

The KCP must be removed from the system if it is not connected, as the EMERGENCY STOP button is not functional in such a case.

If there are several KCPs in a system, it must be ensured that they are not mixed up.

No mouse or keyboard may be connected to the robot controller.

Faults

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

- 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.
- Keep a record of the faults.
- Eliminate the fault and carry out a function test.

3.7.2 Transportation

Robot

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

Further information is contained in the robot operating instructions.

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.

Further information is contained in the robot controller operating instructions.

3.7.3 Start-up

The robot controller must not be put into operation until the internal temperature of the cabinet has adjusted to the ambient temperature. Otherwise, condensation could cause damage to electrical components.

Function test

It must be ensured that no persons or objects are present within the danger zone of the robot during the function test.

The following must be checked during the function test:

- The robot system is installed and connected. There are no foreign bodies or destroyed, loose parts on the robot or in the robot controller.
- All safety devices and protective measures are complete and fully functional.
- All electrical connections are correct.
- The peripheral devices are correctly connected.
- The external environment corresponds to the permissible values indicated in the operating instructions.

Further information is contained in the robot operating instructions and in the robot controller operating instructions.

Setting

It must be ensured that the ratings plate on the robot controller has the same machine data as those entered in the declaration of incorporation. The machine data on the ratings plate of the robot must be entered during start-up.

The robot must not be moved unless the correct machine data are not loaded. Otherwise, damage to property could occur.

Further information is contained in the operating and programming instructions.

3.7.4 Virus protection and network security

The user of the robot system 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.

3.7.5 Programming

The following safety measures must be carried out during programming:

- It must be ensured that no persons are present within the danger zone of the robot during programming.
- New or modified programs must always be tested first in operating mode T1.
- If the drives are not required, they must be switched off to prevent the robot from being moved unintentionally.
- The motors reach temperatures during operation which can cause burns to the skin. Contact should be avoided if at all possible. If necessary, appropriate protective equipment must be used.
- The robot and its tooling must never touch or project beyond the safety fence.
- Components, tooling and other objects must not become jammed as a result of the robot motion, nor must they lead to short-circuits or be liable to fall off.

The following safety measures must be carried out if programming in the danger zone of the robot:

- The robot must only be moved at reduced velocity (max. 250 mm/s). In this way, persons have enough time to move out of the way of hazardous robot motions or to stop the robot.
- To prevent other persons from being able to move the robot, the KCP must be kept within reach of the programmer.
- If two or more persons are working in the system at the same time, they must all use an enabling switch. While the robot is being moved, all persons must remain in constant visual contact and have an unrestricted view of the robot system.

3.7.6 Automatic mode

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

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

If the robot comes to a standstill for no apparent reason, the danger zone must not be entered until the EMERGENCY STOP function has been triggered.

3.7.7 Maintenance and repair

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

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

- 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 robot controller and secure it (e.g. with a padlock) to prevent unauthorized persons from switching it 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.
- 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 equipment is deactivated during maintenance or repair work, it must be reactivated immediately after the work is completed.
- Work on the robot system must be carried out in T1 mode.

Faulty components must be replaced using new components with the same article numbers or equivalent components approved by KUKA Roboter 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 or isolated 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 the KPS (KUKA Power Supply), the KSDs (KUKA Servo Drives) and the intermediate-circuit connecting cables up to 5 minutes after the robot controller has been switched off. To prevent life-threatening injuries, no work may be carried out on the robot system in this time.

Foreign matter, such as swarf, water and dust, must be prevented from entering the robot controller.

Further information is contained in the robot controller operating instructions.

Counterbalancing system

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

The counterbalancing systems correspond to category III, fluid group 2 of directive 97/23/EC (Pressure Equipment Directive).

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

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

- The robot 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. Information about the hazardous substances used can be found in the document **Consumables, Safety Data Sheet**.

3.7.8 Decommissioning, storage and disposal

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

Further information is contained in the robot operating instructions and in the robot controller operating instructions.

4 Operation

4.1 KCP teach pendant

4.1.1 Front view

Function

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

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	(>>> 4.8 "Operating modes" page 41)
Drives ON **SSB GUI**	Switches the robot drives on. Only with Shared Pendant (KCP for KUKA.RoboTeam): SSB GUI calls the user interface of the Safety Selection Board
Drives OFF	Switches the robot drives off.
EMERGENCY STOP pushbutton	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 35)
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 32)
Numeric keypad	(>>> 4.1.3 "Numeric keypad" page 33)
Softkeys	(>>> 4.2.1 "Status keys, menu keys, softkeys" page 35)
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 35)
Menu keys	(>>> 4.2.1 "Status keys, menu keys, softkeys" page 35)

4.1.2 Keypad

Fig. 4-2: KCP 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 38).
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 38).
SYM	SYM must be pressed to enter the secondary characters assigned to the letter keys, e.g. the "#" character on the "A" 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 on KCP

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 38).

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.

4.2 KUKA.HMI user interface

4.2.1 Status keys, menu keys, softkeys

Overview

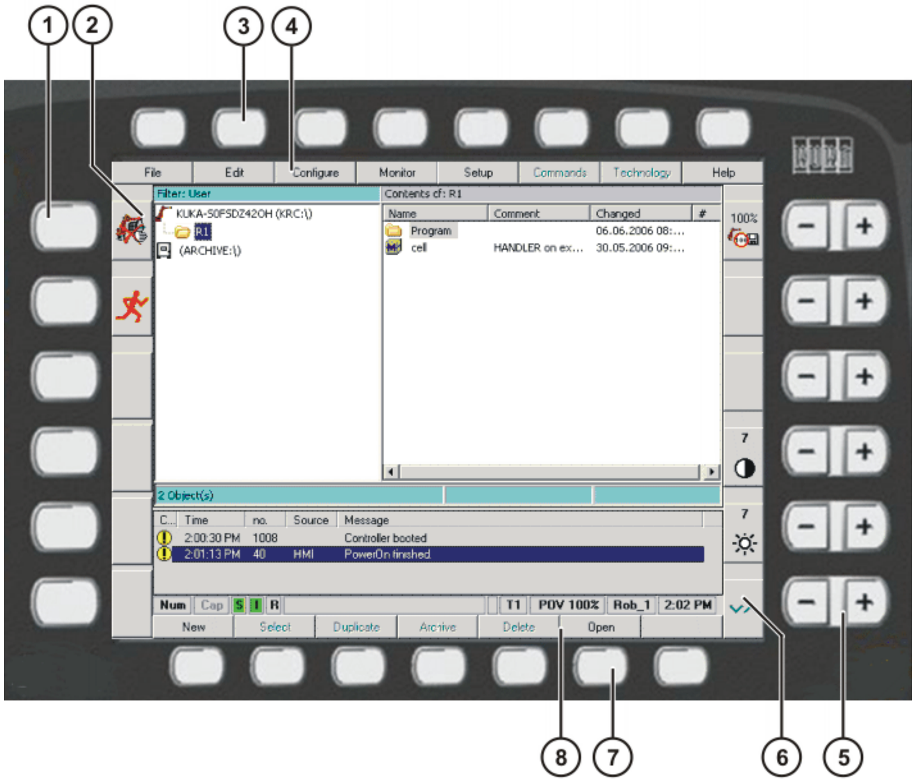


Fig. 4-5: Status keys, menu keys and softkeys in the user interface

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

Description

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

4.2.2 Windows in the user interface

Overview

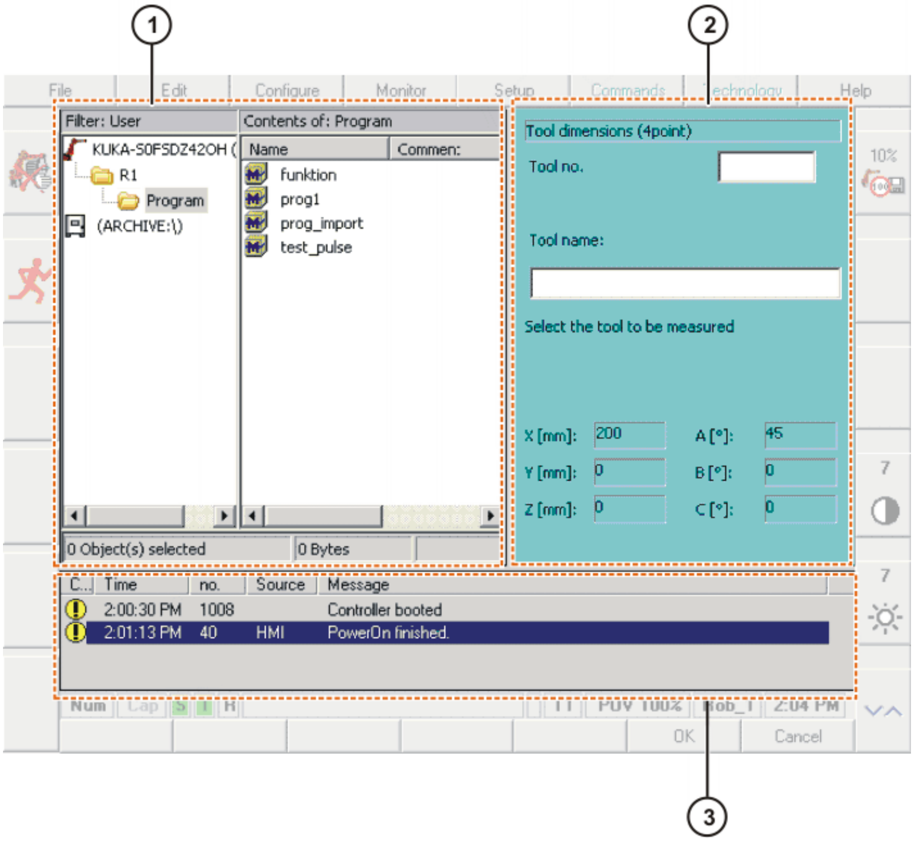


Fig. 4-6: Windows in the user interface

1	Main window	3	Message window
2	Option window		

Description

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

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

4.2.3 Elements in the user interface

Input box

A value or a text can be entered.

Fig. 4-7: Example of an input box

List box

A parameter can be selected from a list.

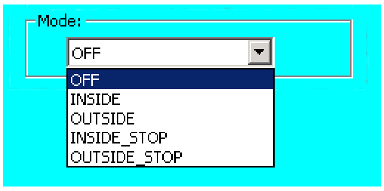


Fig. 4-8: Example of an opened list box

Check box

One or more options can be selected.

Fig. 4-9: Example of a check box

Option box

One option can be selected.

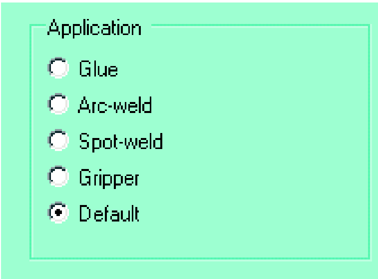


Fig. 4-10: Example of an option box

Slider

A value on a scale can be set.

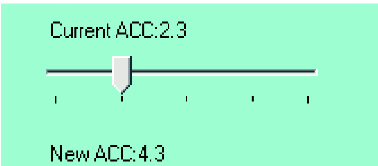


Fig. 4-11: Example of a slider

Group

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

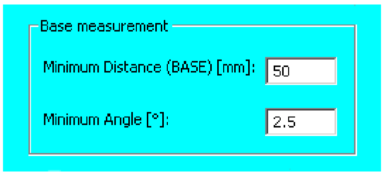


Fig. 4-12: Example of a group

4.2.4 Status bar

Overview

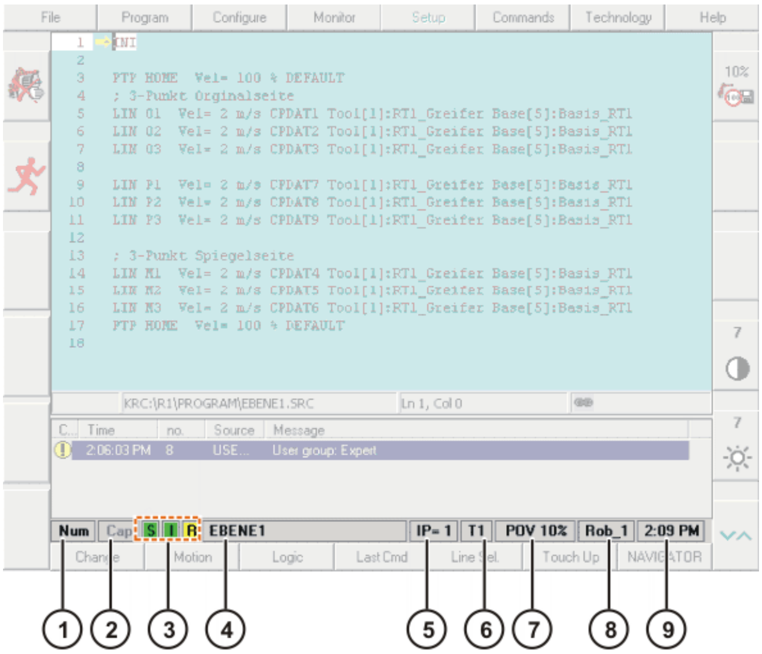


Fig. 4-13: Status bar in the user interface

1 Status of the numeric keypad
2 Upper-case/lower-case status
3 **S**: Status of the Submit interpreter

I/O: Status of the drives

R: Status of the program
4 Name of the selected program
5 Number of the current block
6 Current operating mode
7 Current override setting
8 Robot name
9 System time

Description

Icon	Description
	The numeric function of the numeric keypad is active.
	The control function of the numeric keypad is active.

Icon	Description
Cap	Upper-case characters are active.
Cap	Lower-case characters are active.

Icon	Color	Description
S	gray	Submit interpreter is deselected.
S	red	Submit interpreter has been stopped.
S	green	Submit interpreter is running.
I	green	Drives ready.
O	red	Drives not ready.
R	gray	No program is selected.
R	yellow	The block pointer is situated on the first line of the selected program.
R	green	The program is selected and is being executed.
R	red	The selected and started program has been stopped.
R	black	The block pointer is situated on the last line of the selected program.

Information about the Submit interpreter is contained in the Expert documentation "Submit-Interpreter".

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 the robot controller on.

Procedure

- Turn the main switch on the robot controller to ON.
  The operating system and the KSS start automatically.

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

4.4 Switching the robot controller off

Procedure

- Turn the main switch on the robot controller to OFF.
  The robot controller automatically backs up data.

4.5 Setting the user interface language

Procedure

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

4.6 Changing user group

Description

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

- **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.

When the system is booted, the user group "User" is selected by default. The user groups "Expert" and "Administrator" are password-protected.

Depending on customer-specific settings, additional user groups may also be available.

Procedure

1. Select the menu sequence **Configure** > **User group**.
   The current user group is displayed.
2. Select the new user group by pressing the corresponding softkey.
3. If prompted:
   Enter password and confirm with **OK**.

4.7 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.

4.8 Operating modes

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 deactivated immediately and the robot stops with a STOP 0.

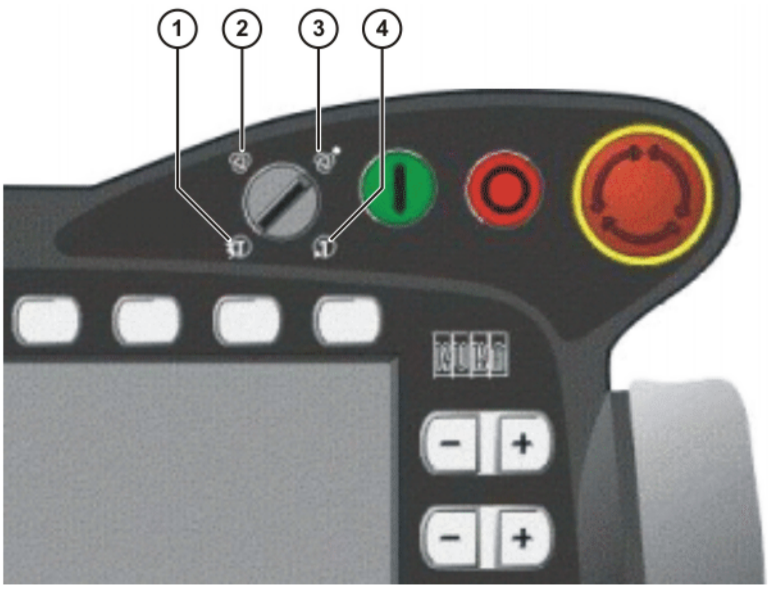


Fig. 4-14: Mode selector switch

1	T2 (Test 2)	3	AUT EXT (Automatic External)
2	AUT (Automatic)	4	T1 (Test 1)

Operating mode	Use	Velocities
T1	For test operation	■ Program mode: Programmed velocity, maximum 250 mm/s ■ Jog mode: Jog velocity, maximum 250 mm/s
T2	For test operation	■ Program mode: Programmed velocity ■ Jog mode: Jog velocity, maximum 250 mm/s
AUT	For robot systems without higher-level controllers Only possible with a connected safety circuit	■ Program mode: Programmed velocity ■ Jog mode: not possible
AUT EXT	For robot systems with higher-level controllers, e.g. PLC Only possible with a connected safety circuit	■ Program mode: Programmed velocity ■ Jog mode: not possible

4.9 Coordinate systems

Overview

The following Cartesian coordinate systems are defined in the robot system:

- WORLD
- ROBROOT
- BASE
- TOOL

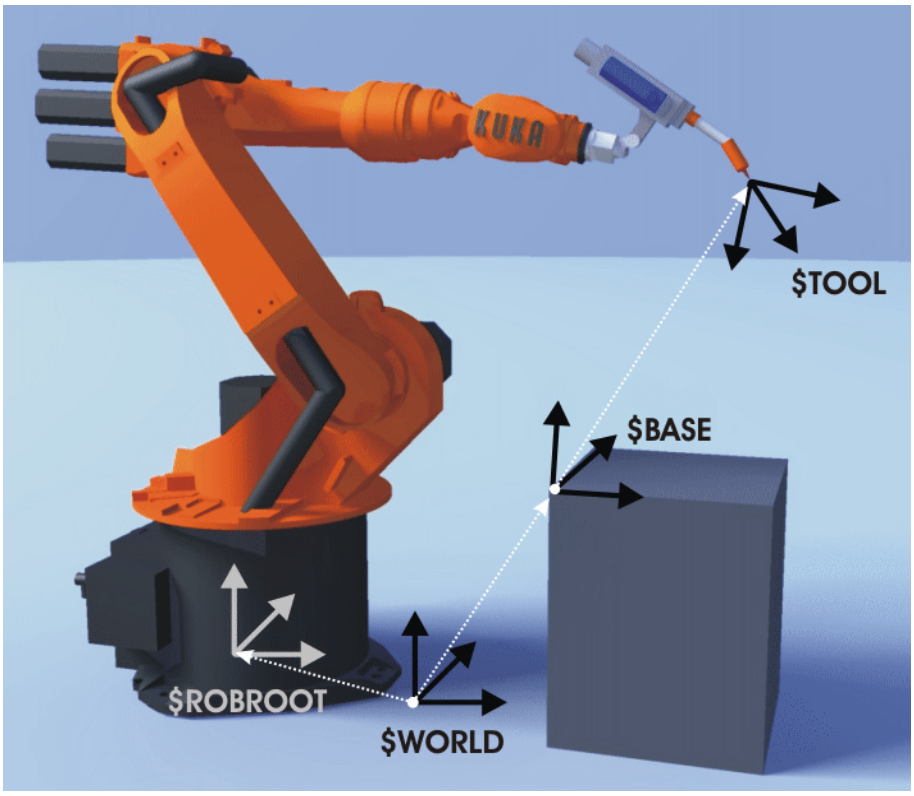


Fig. 4-15: Overview of coordinate systems

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.

(>>> 5.4.3 "Base calibration" page 99)

TOOL

The TOOL coordinate system is a Cartesian coordinate system which is located at the tool center point. It is relative to the BASE coordinate system.

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.

(>>> 5.4.1 "Tool calibration" page 88)

4.10 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.

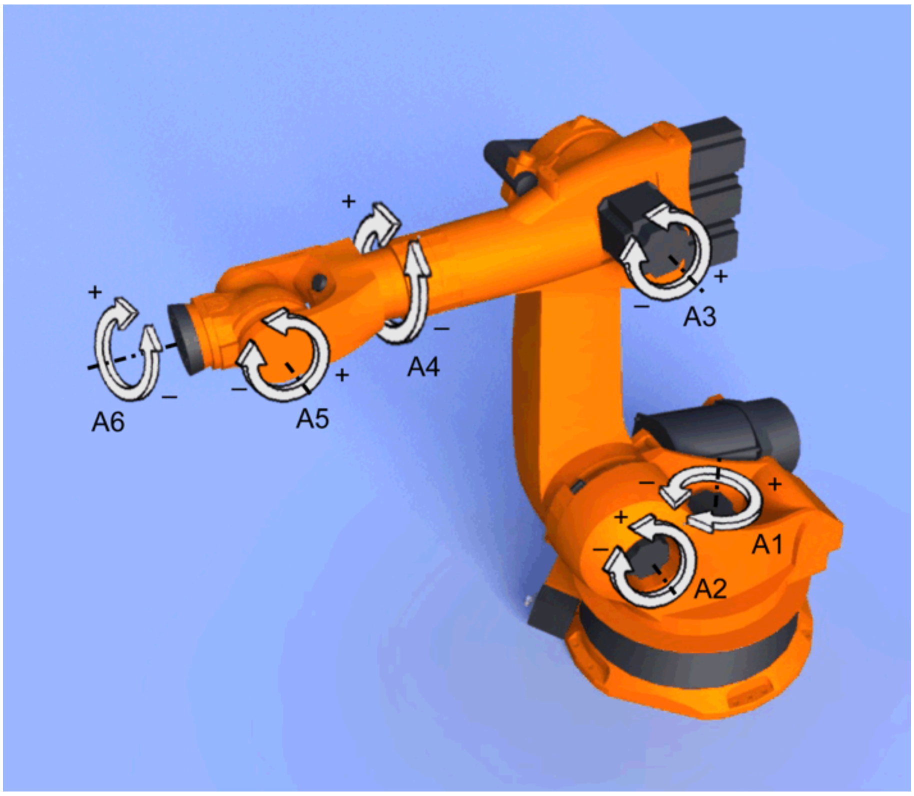


Fig. 4-16: Axis-specific jogging

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	(>>> 4.10.4 "Cartesian jogging with the jog keys" page 46)	(>>> 4.10.3 "Axis-specific jogging with the jog keys" page 45)
Space Mouse	(>>> 4.10.7 "Cartesian jogging with the Space Mouse" page 49)	Axis-specific jogging with the Space Mouse is possible, but is not described here.

4.10.1 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.

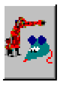
Preparation

- Define the jog override intervals:
  Select the menu sequence **Configure** > **Jogging** > **Jog OV Steps**.

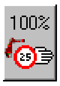
Active	Meaning
No	The override can be adjusted in 1% steps.
Yes	Intervals: 100%, 75%, 50%, 30%, 10%, 3%, 1%

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.

4.10.2 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** > **Cur. 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**.

4.10.3 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 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.

4.10.4 Cartesian jogging with the jog keys

Precondition

- Tool and base have been selected.
  (>>> 4.10.2 "Selecting the tool and base" page 45)
- 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**.

4.10.5 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

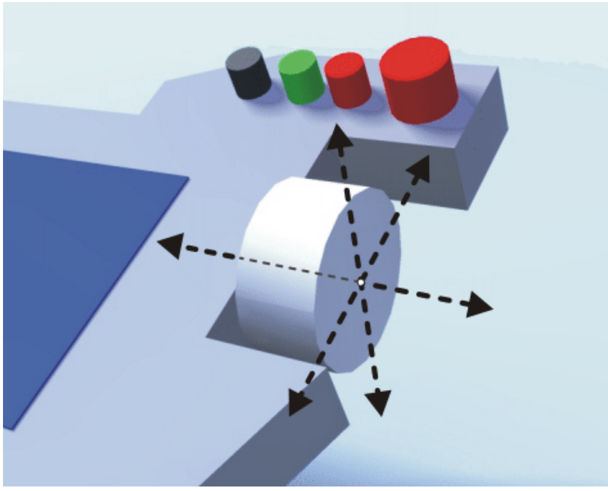


Fig. 4-17: Pushing and pulling the Space Mouse

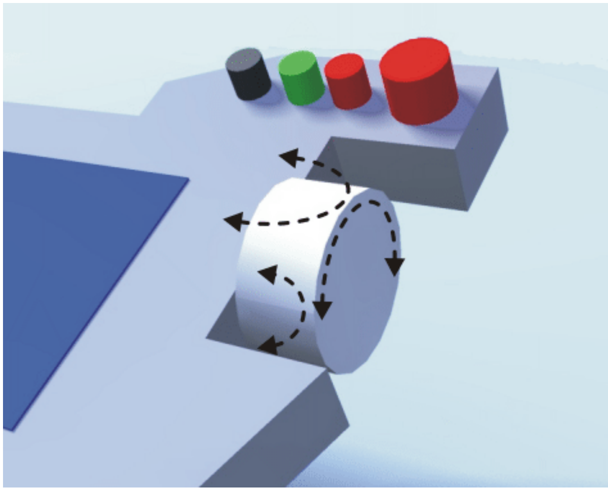


Fig. 4-18: Rotating and tilting the Space Mouse

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.

4.10.6 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.

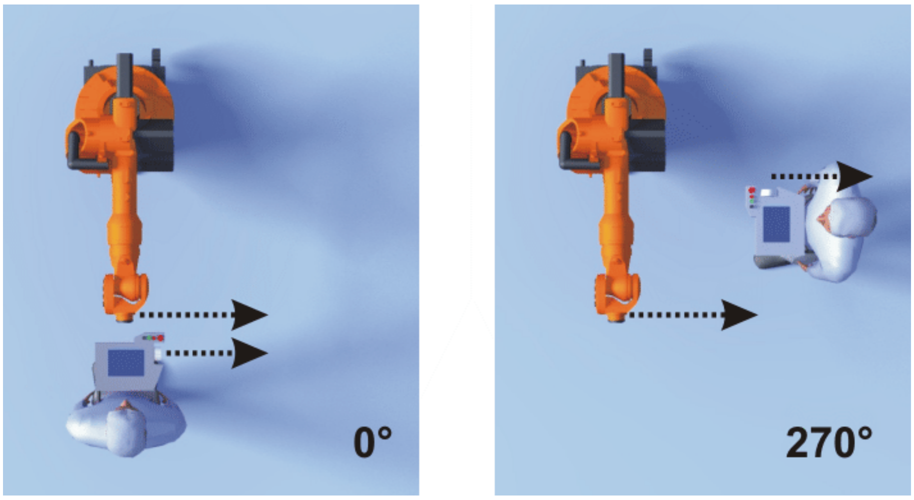


Fig. 4-19: Space Mouse: 0° and 270°

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.

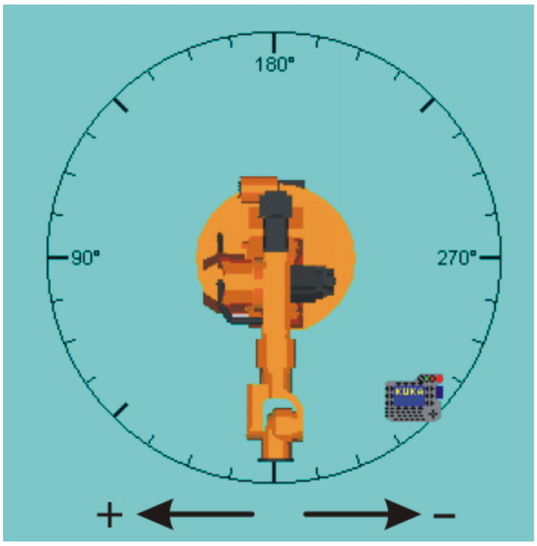


Fig. 4-20: Option window for aligning the Space Mouse

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°.

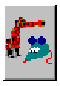
4.10.7 Cartesian jogging with the Space Mouse

Precondition

- Tool and base have been selected.
  (>>> 4.10.2 "Selecting the tool and base" page 45)
- The Space Mouse is configured.
  (>>> 4.10.5 "Configuring the Space Mouse" page 46)
- The alignment of the Space Mouse has been defined.
  (>>> 4.10.6 "Defining the alignment of the Space Mouse" page 48)
- 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 **Monitor** > **Rob. Position**.

4.10.8 Incremental jogging

Description

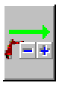
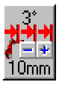
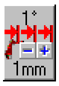
Incremental jogging makes it possible to move the robot a defined distance,e.g. 10 mm or 3°. The robot then stops by itself.

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.

Area 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

The following status keys are available in the right-hand status key bar for incremental jogging:

Status key	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.

   (>>> 4.10.3 "Axis-specific jogging with the jog keys" page 45)
   (>>> 4.10.4 "Cartesian jogging with the jog keys" page 46)

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.

4.11 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. (>>> 6.7 "Configuring workspaces" page 110)

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 or T2.

Procedure

1. Select the menu sequence **Configure** > **Tools** > **Monitoring working envelope** > **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.

4.12 Monitor functions

4.12.1 Overview of the monitor functions

Topic	Monitor functions
Displaying the current robot position	(>>> 4.12.2 "Displaying the actual position" page 52)
Displaying inputs/outputs	(>>> 4.12.4 "Displaying analog inputs/outputs" page 53) (>>> 4.12.3 "Displaying digital inputs/outputs" page 52) (>>> 4.12.5 "Displaying inputs/outputs for Automatic External" page 54)
Displaying (and modifying) variables	(>>> 4.12.6 "Displaying and modifying the value of a variable" page 56) (>>> 4.12.8 "Displaying the variable overview and modifying variables" page 58) (>>> 4.12.7 "Displaying the state of a variable" page 57)
Displaying information about the robot system	(>>> 4.12.10 "Displaying information about the robot system" page 61) (>>> 4.12.11 "Displaying robot data" page 62)
Displaying information about the hardware	(>>> 4.12.12 "Displaying hardware information" page 62)

Other monitor functions: (>>> 9 "Diagnosis" page 241)

4.12.2 Displaying the actual position

Procedure

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

Description

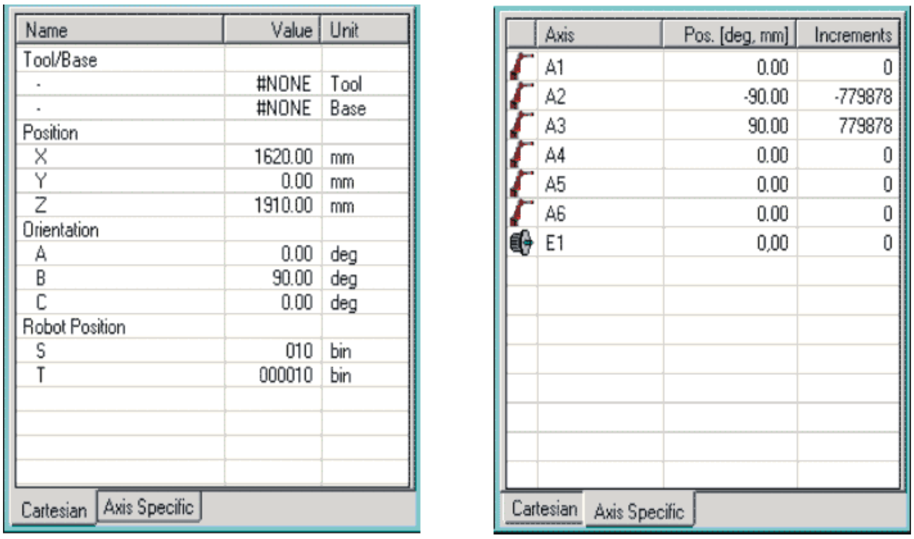


Fig. 4-21: Actual position: Cartesian and axis-specific

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.

4.12.3 Displaying digital inputs/outputs

Procedure

- Select the menu sequence **Monitor** > **I/O** > **Digital Outputs** or **Digital Inputs**.

Description

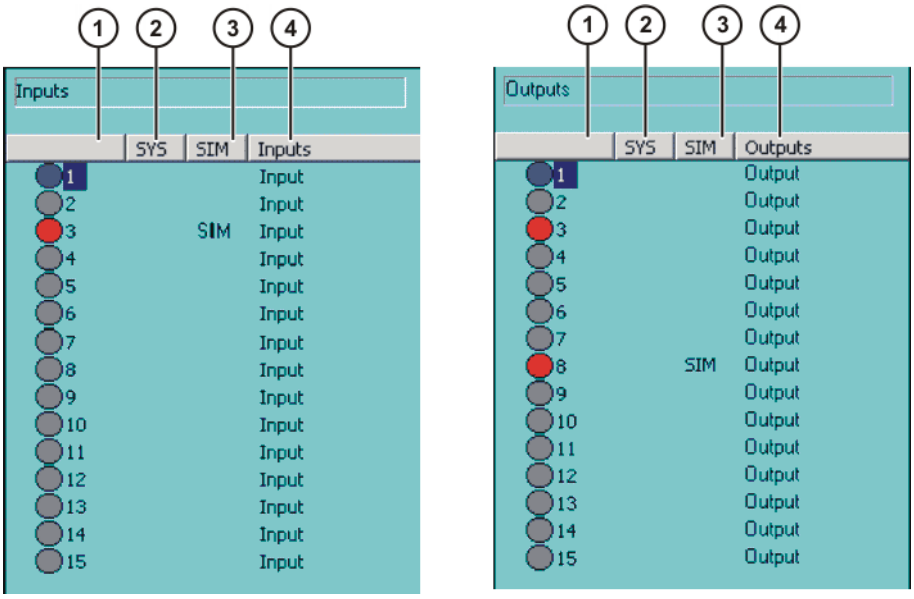


Fig. 4-22: Digital inputs/outputs

Column	Description
1	Input/output number. The icon is red if the input or output is set.
2	SYS entry: input/output whose value is saved in a system variable.
3	SIM entry: simulated input/output. This column is only displayed if I/O simulation is activated.
4	Name of the input/output

The following softkeys are available:

Softkey	Description
Value	Toggles the selected input/output between TRUE and FALSE. Precondition: The enabling switch is pressed. This softkey is not available in AUT mode.
Name	The name of the selected input or output can be changed.

An input/output can be selected by entering its number via the numeric keypad. In this way, inputs/outputs that are not visible in the option window can be displayed. Preconditions:

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

4.12.4 Displaying analog inputs/outputs

Procedure

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

Description

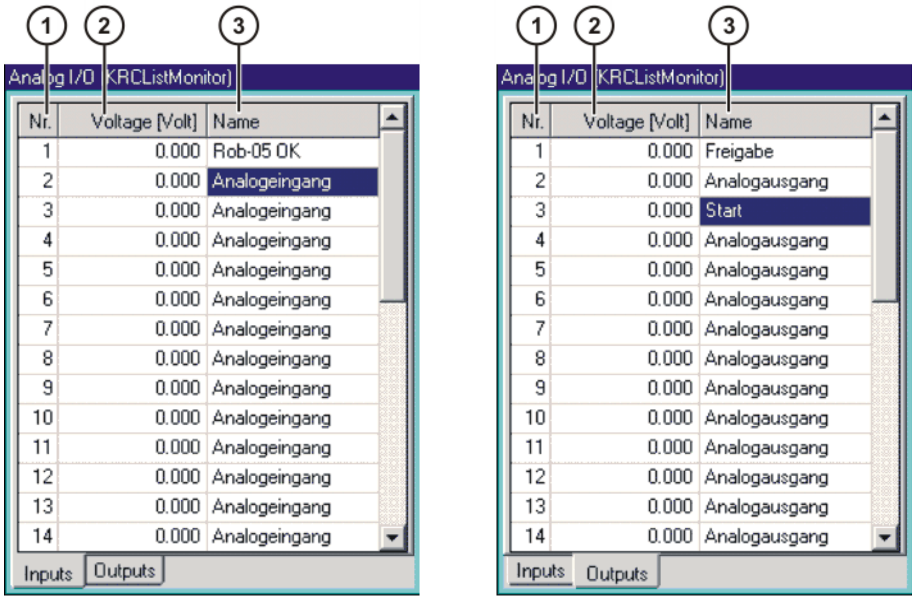


Fig. 4-23: Analog inputs/outputs

Column	Description
1	Input/output number
2	Input/output voltage Range of values: -10 to +10 volts
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. This softkey is not available for inputs.
Name	The name of the selected input or output can be changed.

An input/output can be selected by entering its number via the numeric keypad. In this way, inputs/outputs that are not visible in the option window can be displayed. Preconditions:

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

4.12.5 Displaying inputs/outputs for Automatic External

Procedure

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

Description

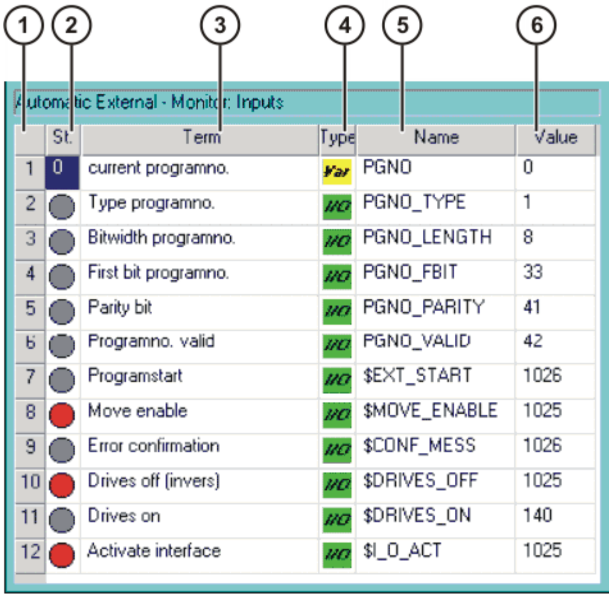


Fig. 4-24: Automatic External inputs (detail view)

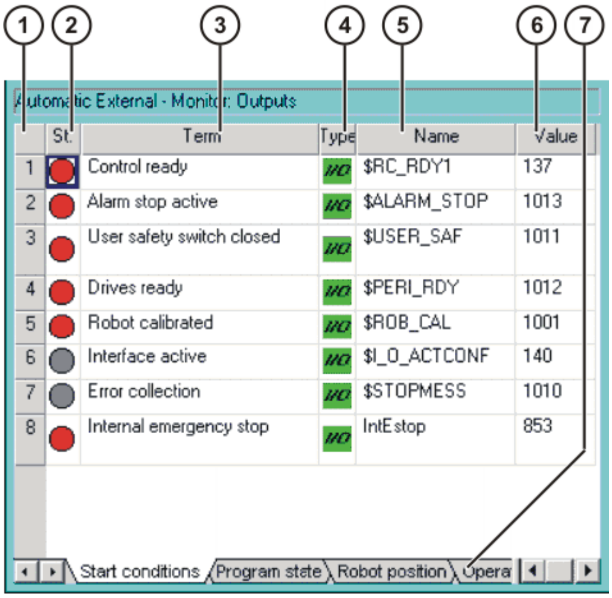


Fig. 4-25: Automatic External outputs (detail view)

Column	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

Column	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
Configure	Switches to the configuration of the Automatic External interface. (>>> 6.13.2 "Configuring Automatic External inputs/outputs" page 128)
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.

4.12.6 Displaying and modifying the value of a variable

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

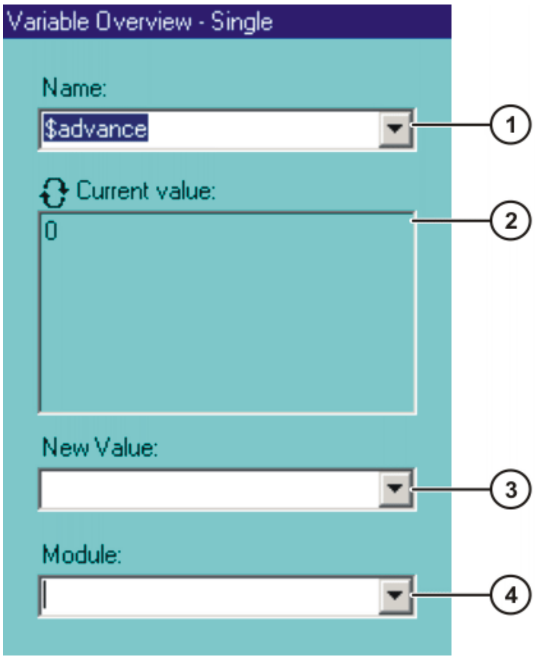
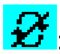
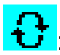


Fig. 4-26: Variable Overview - Single window

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 **New Value** of **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.

4.12.7 Displaying the state of a variable

Description

Variables can have the following states:

- UNKNOWN: The variable is unknown.
- DECLARED: This 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.

4.12.8 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. (>>> 4.12.9 "Variable overview configuration" page 59)

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 using the **Tab +** softkey.
3. Select the cell to be modified. Carry out modification using the softkeys.
4. Press the **OK** softkey to save the change and close the window.

Description

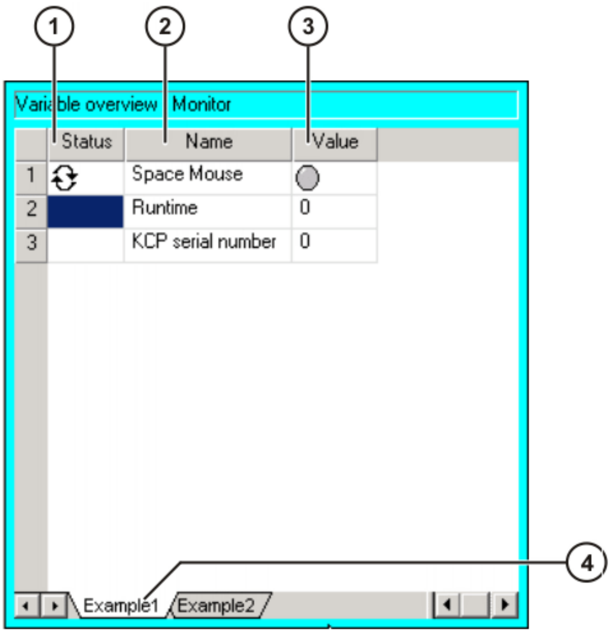


Fig. 4-27: Variable overview - Monitor window

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	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
Configure	Switches to the configuration of the variable overview. 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.

4.12.9 Variable overview configuration

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. Both 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. Making 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

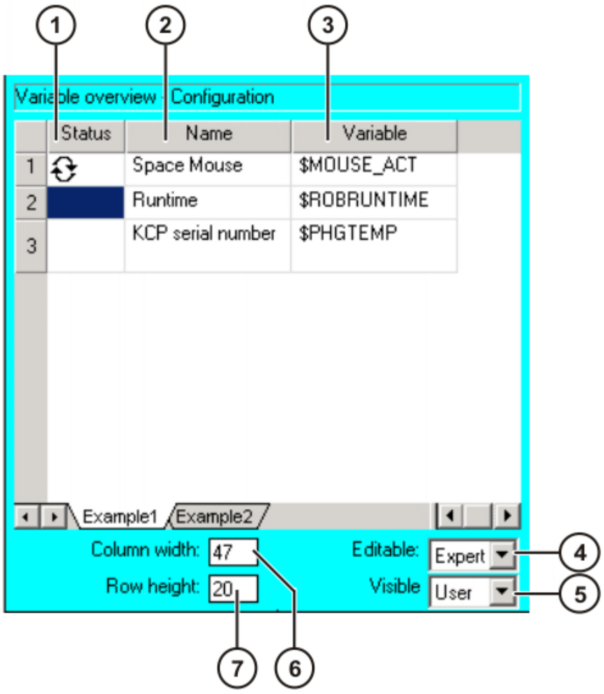


Fig. 4-28: Variable overview - Configuration window

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	Lowest user group in which the variable overview can be modified.
5	Lowest user group in which the variable overview can be displayed.
6	Column width in mm
7	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.
Tab +	Switches to the next group.
Jump	Sets the focus on the next element in the user interface.

Softkey	Description
Paste	Shows additional softkeys: ■ **R. above**: Inserts a new row above the one currently selected. ■ **R. below**: Inserts a new row below the one currently selected. ■ **G. before**: Inserts a new group to the left of the one currently selected. ■ **G. after**: Inserts a new group to the right of the one currently selected.
Delete	Shows additional softkeys: ■ **Row**: The selected row is deleted. ■ **Group**: The current group is deleted.

4.12.10 Displaying information about the robot system

Procedure

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

Description

Information about the robot system 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 ■ Operating hours The operating hours are the time the drives have been switched on. ■ 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 **Save** softkey exports the contents of the **Virus Scanner** tab to the file C:\ KRC\ ROBOTER\ LOG\ VIRUS-INFO.XML.

4.12.11 Displaying robot data

Procedure

1. Select the menu sequence **Setup** > **Robot data**.
2. The following data are displayed:
   - Robot name

     The robot name can be changed.
   - Serial number
   - Operating hours

     The operating hours are the time the drives have been switched on.
   - Machine data

4.12.12 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.

4.13 Program management

4.13.1 Navigator file manager

Overview

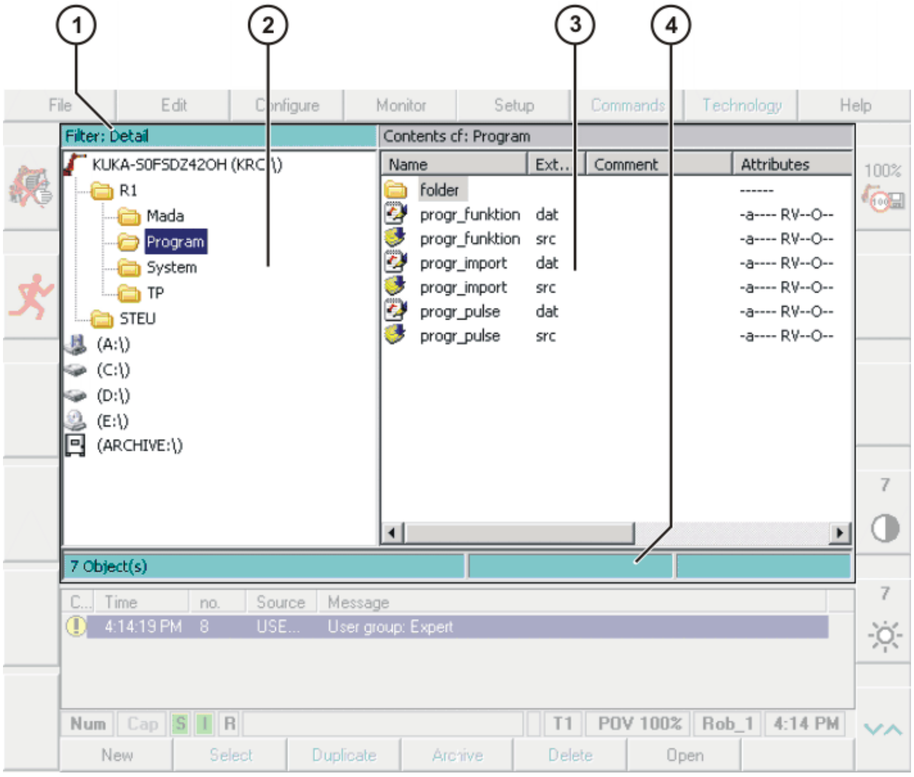


Fig. 4-29: Navigator

1	Header	3	File list
2	Directory structure	4	Status bar

Description

In the Navigator, the user manages programs and all system-specific files.

Header

- Left-hand area: the selected filter is displayed.
  (>>> 4.13.1.1 "Selecting filters" page 64)
- 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
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

4.13.1.1 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**.

4.13.1.2 Changing properties

Description

The properties of objects can be displayed and changed.

Precondition

- To change properties: user group “Expert”.

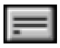
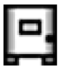
Procedure

1. Select the object in the directory structure or in the file list.

2. Select the menu sequence **File** > **Properties**.
   The properties display is opened. The layout is dependent on the object selected.

4.13.1.3 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.

4.13.1.4 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**.

4.13.1.5 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 **OK**.

It is not possible to select a template in the user group "User". By default, a program of type "Module" is created.

4.13.1.6 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**.

4.13.1.7 Toggling between the Navigator and the program

Description

If a program is selected, it is possible to toggle to the Navigator without having to deselect the program. The user can then return to the program.

Procedure

- Toggling from the program to the Navigator: press the **NAVIGATOR** softkey.
- Toggling from the program to the Navigator: press the **PROGRAM** softkey.

4.13.2 Selecting and deselecting a program

Description

Before a program can be started or edited, it must first be selected. Following program execution or editing, it must be deselected again.

Procedure

1. Select the program in the Navigator.
   If the program is displayed as a SRC file and a DAT file, the SRC file or the DAT file can be selected.
2. Press the **Select** softkey.

3. Execute or edit the program.
4. Select the menu sequence **Program** > **Cancel program**.

4.13.3 Displaying/hiding program sections

4.13.3.1 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.

4.13.3.2 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.

4.13.3.3 Activating/deactivating line breaks

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.

Fig. 4-30: 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.

4.13.3.4 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.
  (>>> 4.13.5.3 "Creating Folds" page 75)

If a program is reset or closed, all Folds are automatically closed.

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

Fig. 4-31: 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. 4-32: 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.

4.13.4 Starting a program

4.13.4.1 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".
	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 display. 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.

4.13.4.2 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.

> **Caution!**
> Altering the advance run may disrupt program execution. The advance run may only be altered after consultation with KUKA!

4.13.4.3 Icons in the program

Line break

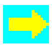
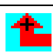
If a line is wider than the program window, the line is broken by default. The part of the line after the break has no line number and is marked with a black, L-shaped arrow. The line break function can be deactivated.
(>>> 4.13.3.3 "Activating/deactivating line breaks" page 67)

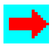
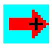
```
25   INTERRUPT DECL 15 WHEN
   ↳ $MEAS_PULSE[TOUCH_I[TOUCH_ACTIVE].IN_NR] DO H70 (6,CD0 )
```

Fig. 4-33: 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".

Icon	Description
	Normal arrow (red): The robot has completed the motion block in the backwards direction
	Normal arrow (red) with plus sign: The robot has completed the motion block in the backwards direction This block pointer is not displayed in the user group "User".

Icon	Description
20 ↑USER_GRP 21 SPOT INI	The block pointer is located higher up in the program.
19 GRIPPER INI ↓	The block pointer is located lower down in the program.

4.13.4.4 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.

4.13.4.5 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 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 always takes place by the direct route from the current position to the destination position. Make sure that there are no obstacles between these positions in order 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.

4.13.4.6 Starting a program forwards (automatic)

Precondition

- 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 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 always takes place by the direct route from the current position to the destination position. Make sure that there are no obstacles between these positions in order to avoid collisions. The velocity is automatically reduced during the BCO run.

4. Press Start key. Program is executed.

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

4.13.4.7 Starting a program backwards

Description

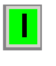
In the case of backward motion, the robot stops at every point. Approximate positioning is not possible.

Exactly how the controller responds during backward motion depends on the configuration.
(>>> 6.11 "Backward motion" page 118)

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 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 always takes place by the direct route from the current position to the destination position. Make sure that there are no obstacles between these positions in order 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.

4.13.4.8 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**.

4.13.4.9 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 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 always takes place by the direct route from the current position to the destination position. Make sure that there are no obstacles between these positions in order 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.

4.13.5 Editing a program

4.13.5.1 Inserting a comment or stamp

Precondition

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

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.

Comment description

Fig. 4-34: Inline form "Comment"

Item	Description
1	Any text

Stamp description

A stamp is a comment that is extended to include the system date and time and the user ID.

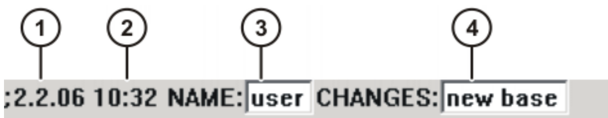


Fig. 4-35: Inline form "Stamp"

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

4.13.5.2 Deleting program lines

Precondition

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

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!

4.13.5.3 Creating Folds

Syntax

`;FOLD` *Name*

Statements

`;ENDFOLD` *<Name>*

The ENDFOLD lines can be assigned more easily if the name of the Fold is entered here as well. Folds can be nested.

Precondition

- User group “Expert”
- Program is selected or open.

Procedure

1. Enter Fold in program. A double semicolon prevents the Fold from closing when edited.

```
;;FOLD outputs
$OUT[1] = TRUE
$OUT[2] = TRUE
$OUT[3] = TRUE
;;ENDFOLD outputs
```

Fig. 4-36: Creating a sample Fold, step 1

2. Delete the second semicolon.

```
;FOLD outputs
$OUT[1] = TRUE
$OUT[2] = TRUE
$OUT[3] = TRUE
;ENDFOLD outputs
```

Fig. 4-37: Creating a sample Fold, step 2

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

```
outputs
```

Fig. 4-38: Creating a sample Fold, step 3

4.13.5.4 Additional editing functions

The following additional program editing functions can be found in the **Program** menu:

- **Copy**
- **Paste**
- **Cut**
- **Find**
- **Replace**

4.13.6 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**.

4.13.7 Archiving

4.13.7.1 Formatting the floppy disk

Procedure

1. Insert floppy disk into the disk 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.

4.13.7.2 Archiving data

Preparation

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

- Format floppy disk if required.

  (>>> 4.13.7.1 "Formatting the floppy disk" page 76)

Procedure

1. Insert floppy disk into the disk drive.
2. Select the menu sequence **File** > **Archive** and the desired menu item.
3. Confirm the request for confirmation with **Yes**.

   A message is generated when archiving is complete or if another floppy disk is required. The file ARCHIVE.ZIP is generated by default.

Description

The following menu items are available for archiving. Exactly which files are archived depends on the configuration in the KRC Configurator (>>> 6.14.8 "Archive Manager tab" page 157).

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 Tech-Pack	The configuration of the installed technology packages is archived. Not available in the user group "User".

Menu item	Description
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 different menu item 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.

Archiving can also be carried out to a network directory. The desired path can be set in the KRC Configurator.

4.13.7.3 Restoring data

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.

Procedure

1. Insert floppy disk containing archived data into the disk drive.
2. Select the menu sequence **File** > **Restore** and the desired menu item.
3. Confirm the request for confirmation with **Yes**.

   A message indicates completion of the restoration process.
4. If data have been archived on more than one disk, insert the next floppy disk and repeat steps 2 and 3.
5. Remove floppy disk from drive and reboot system.

Description

With the exception of **Log Data**, the menu items available for restoring data are the same as those available for archiving.

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.

If the data were archived on a network drive, this drive is accessed for restoring the data.

5 Start-up

5.1 Start-up overview

The robot cannot be started up until the robot system has been installed and connected.
Further information is contained in the robot operating instructions and in the robot controller operating instructions.

Step	Description
1	Configure the inputs/outputs between the robot controller and the periphery in the file IOSYS.INI. Information can be found in the field bus documentation.
2	Check machine data. (>>> 5.2 "Checking the machine data" page 79)
3	Master robot. (>>> 5.3 "Mastering" page 80)
4	Calibrate tool. In the case of a fixed tool: calibrate external TCP. (>>> 5.4.1 "Tool calibration" page 88) (>>> 5.4.2 "Calibration of an external TCP and fixed tool" page 95)
5	Enter load data. (>>> 5.5 "Load data" page 102)
6	Calibrate base. In the case of a fixed tool: calibrate workpiece. (>>> 5.4.3 "Base calibration" page 99) (>>> 5.4.2 "Calibration of an external TCP and fixed tool" page 95)

In addition to this, other start-up functions are also available:

- If the robot is to be controlled from a host computer or PLC: configure Automatic External interface.
  (>>> 6.13 "Configuring Automatic External" page 127)
- 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.
  (>>> 5.6 "Transferring long text names" page 105)

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.

Warning!
The robot must not be moved if incorrect machine data are loaded. Physical injuries or damage to property may result in this case.

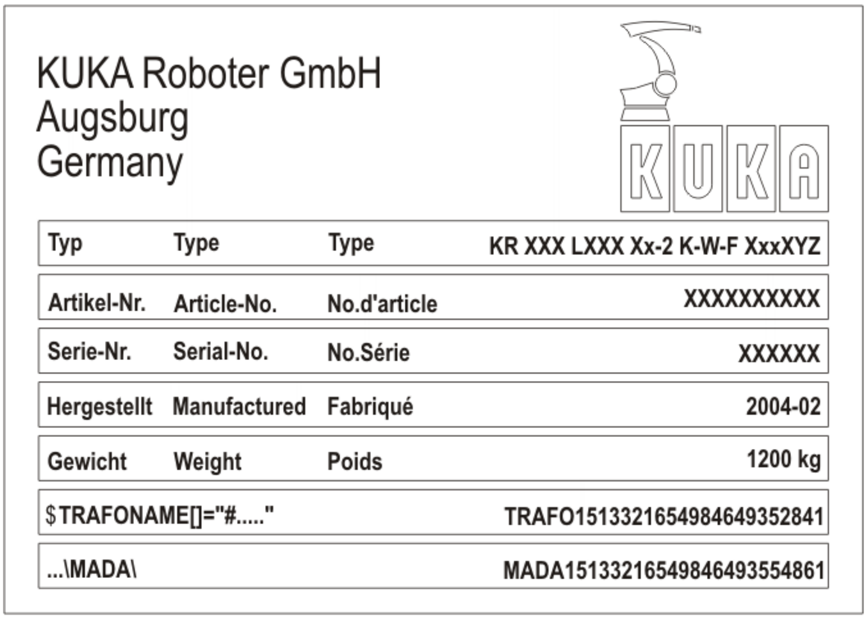


Fig. 5-1: Rating plate

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 Mastering

5.3.1 Mastering overview

During mastering, the robot is moved to the mechanical zero position, and the encoder value for each axis is saved. In this way, the mechanical zero position and the electronic zero position are made to coincide.

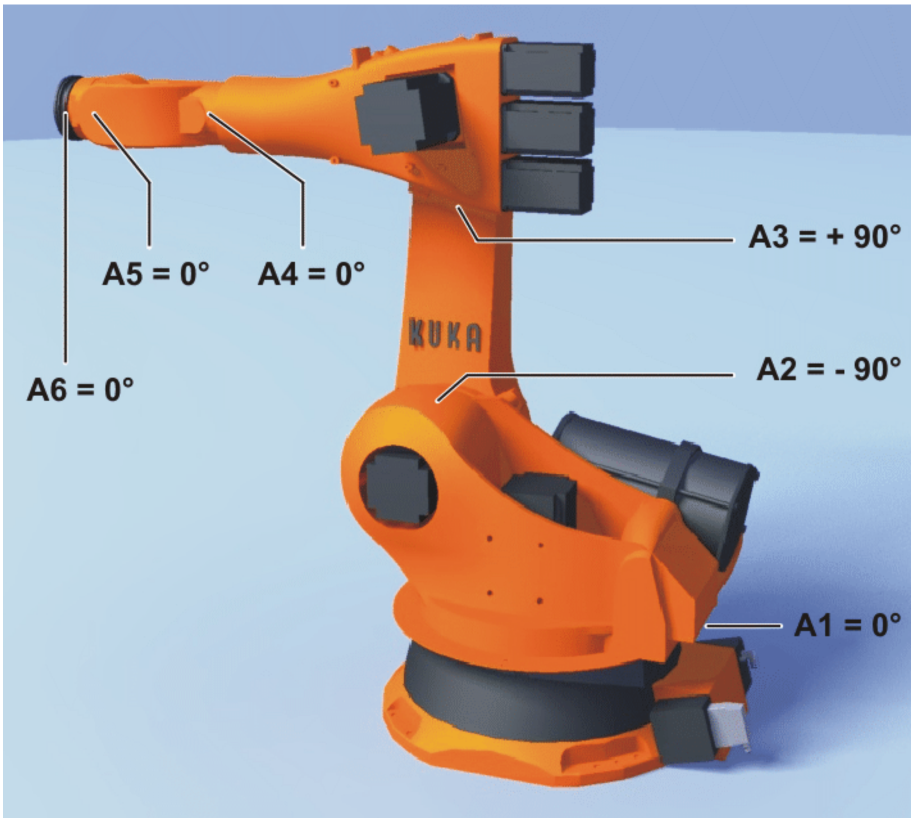


Fig. 5-2: Mechanical zero position

Only a mastered robot can move to programmed positions and be moved using Cartesian coordinates.

A robot must be mastered in the following cases:

- During start-up
- After repairs (e.g. after replacement of a motor or RDC)
- 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!
- After an impact with an end stop at more than 250 mm/s

  Before carrying out a new mastering procedure, the old mastering data must first be deleted!
- After a collision

  Before carrying out a new mastering procedure, the old mastering data must first be deleted!

Mastering data are deleted by manually unmastering the axes.
(>>> 5.3.9 "Manually unmastering axes" page 88)

5.3.2 Mastering methods

Overview

There are 2 ways of mastering a robot:

- With the EMT (electronic mastering tool)
- With the dial gauge

EMT mastering is recommended.

Description of mastering with the EMT

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

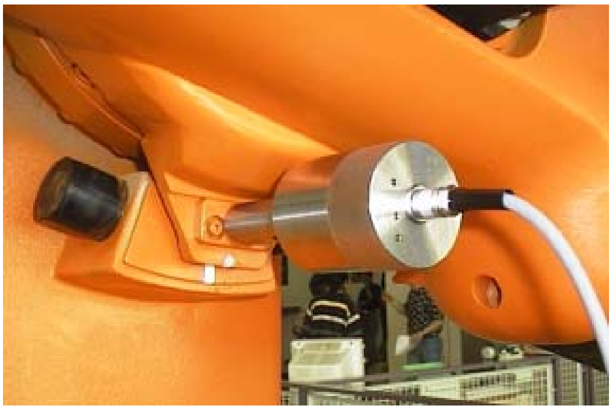


Fig. 5-3: Electronic measuring tool

EMT mastering consists of the following steps:

Step	Description
1.	**First mastering** (>>> 5.3.4 "First mastering with the EMT" page 85) First mastering is carried out without a load.
2.	**Teach offset** (>>> 5.3.5 "Teach offset" page 85) "Teach offset" is carried out with a load. The difference from the first mastering is saved.
3.	If necessary: **Load mastering with offset** (>>> 5.3.6 "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.

Description of mastering with the dial gauge

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

(>>> 5.3.7 "Mastering with the dial gauge" page 87)

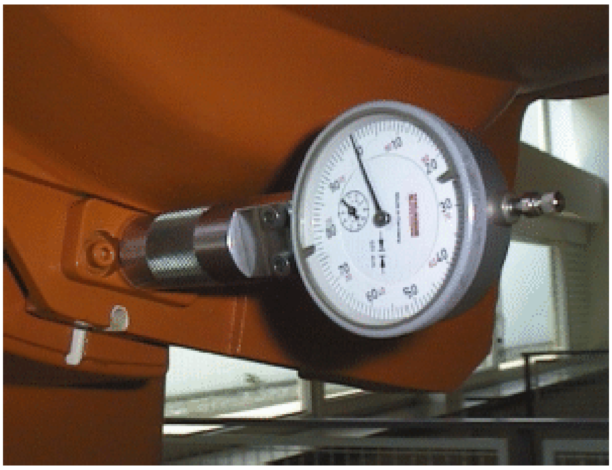


Fig. 5-4: Dial gauge

5.3.3 Moving axes to the pre-mastering position

Description

Each axis is moved so that the mastering marks line up. The pre-mastering position is a prerequisite for every mastering.

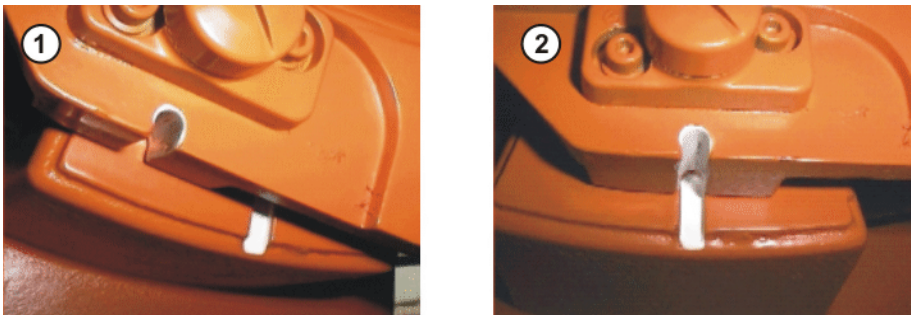


Fig. 5-5: Moving an axis to the pre-mastering position

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

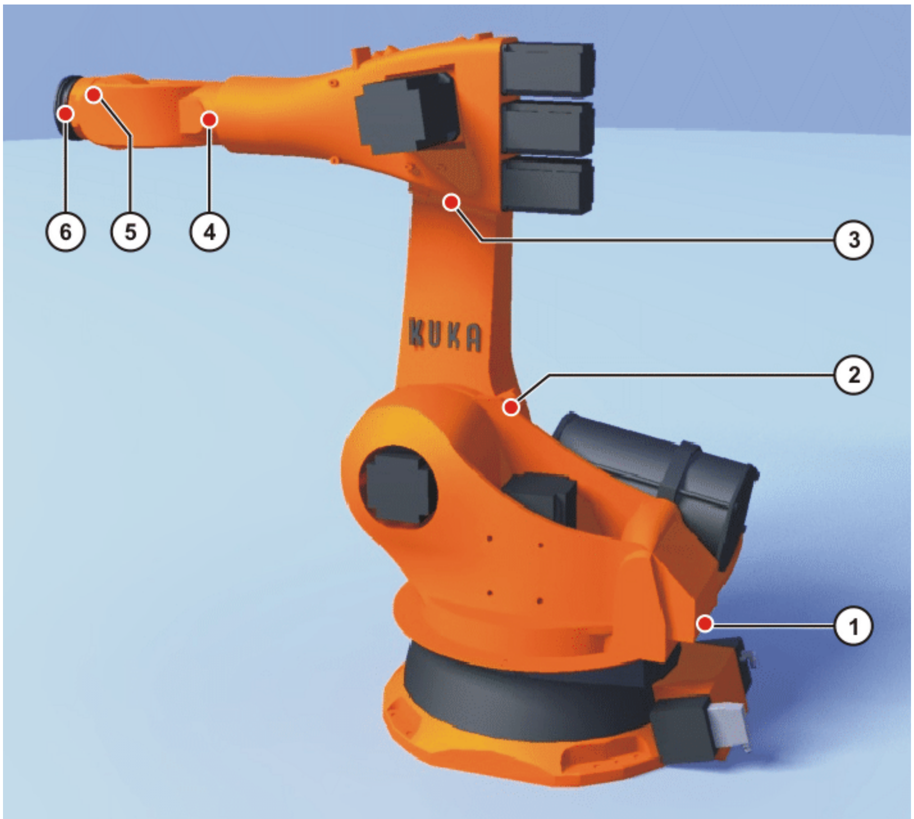


Fig. 5-6: Mastering marks on the robot

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.

Move A4 and A6 into the pre-mastering position so that the following positions are approximately reached:

- A4: 0°
- A6: +90°

In this way, A4 and A6 reach the correct mechanical zero position during mastering and are not rotated.
For verification purposes, display the positions of the axes robot during mastering: select the menu sequence **Monitor** > **Rob. Position** > **Axis specific**.

5.3.4 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.

A4 and A6 must not be rotated.
(>>> 5.3.3 "Moving axes to the pre-mastering position" page 83)

- No program is selected.
- Operating mode T1

Procedure

1. Select the menu **Setup** > **Master** > **EMT** > **With load correction** > **First mastering**.

   An option window is opened. 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 option window. Screw EMT onto gauge cartridge. Then attach signal cable to EMT and plug into connector X32 on the base frame junction box.

Caution!
The EMT must always be screwed onto and removed from the gauge cartridge **without** the signal cable attached. Otherwise, the signal cable could be damaged.

3. Press the **Master** softkey.
4. Press an enabling switch and the Start key.

   When the EMT detects the lowest point of the reference notch, the mechanical zero position is reached. The robot stops automatically. The values are saved. The axis is no longer displayed in the option window.
5. Remove signal cable from EMT. Then remove EMT from the gauge cartridge and replace the protective cap.
6. Repeat steps 2 to 5 for all axes to be mastered.

5.3.5 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.

A4 and A6 must not be rotated.
(>>> 5.3.3 "Moving axes to the pre-mastering position" page 83)

- No program is selected.
- Operating mode T1

Procedure

1. Select the menu **Setup** > **Master** > **EMT** > **With load correction** > **Teach offset**.

2. Enter tool number. Confirm with **Tool OK**.

   An option window is opened. 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 option window. Screw EMT onto gauge cartridge. Then attach signal cable to EMT and plug into connector X32 on the base frame junction box.

Caution!
The EMT must always be screwed onto and removed from the gauge cartridge **without** the signal cable attached. Otherwise, the signal cable could be damaged.

4. Press the softkey **Teach**.
5. Press an enabling switch and the Start key.

   When the EMT detects the lowest point of the reference notch, the mechanical zero position is reached. The robot stops automatically. An option window is opened. 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 option window.
7. Remove signal cable from EMT. Then remove EMT from the gauge cartridge and replace the protective cap.
8. Repeat steps 3 to 7 for all axes to be mastered.

5.3.6 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.

A4 and A6 must not be rotated.
(>>> 5.3.3 "Moving axes to the pre-mastering position" page 83)

- No program is selected.
- Operating mode T1

Procedure

1. Select the menu **Setup** > **Master** > **EMT** > **With load correction** > **Master load** > **With offset**.
2. Enter tool number. Confirm with **Tool OK**.

   An option window is opened. 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 option window. Mount EMT on gauge cartridge. Then attach signal cable to EMT and plug into connector X32 on the base frame junction box.

Caution!
The EMT must always be screwed onto and removed from the gauge cartridge **without** the signal cable attached. Otherwise, the signal cable could be damaged.

4. Press the softkey **Check**.
5. Hold down an enabling switch and press the Start key.

   When the EMT detects the lowest point of the reference notch, the mechanical zero 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 signal cable from EMT. Then remove EMT from the gauge cartridge and replace the protective cap.
8. Repeat steps 3 to 7 for all axes to be mastered.

5.3.7 Mastering with the dial gauge

Description

Dial mastering is always carried out with a load. It is not possible to save mastering data for different loads.

Precondition

- The load is mounted on the robot.
- All axes are in the pre-mastering position.

A4 and A6 must not be rotated.
(>>> 5.3.3 "Moving axes to the pre-mastering position" page 83)

- Axis-specific jogging with the jog keys is selected.

  (>>> 4.10.3 "Axis-specific jogging with the jog keys" page 45)
- No program is selected.
- Operating mode T1

Procedure

1. Select the menu sequence **Setup** > **Master** > **Dial**.

   An option window is opened. 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 the zero position.
7. Switch to incremental jogging in the right-hand status key bar.
8. Move the axis from “+” to “-” until the zero position is reached.

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

9. Press the **Master** softkey. The axis that has been mastered is removed from the option 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.

5.3.8 Saving the mastering

Procedure

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

Description

Save current data saves all mastering data to the hard drive.

This prevents loss of the mastering data if the robot controller cannot be shut down properly, e.g. due to a defective battery.

5.3.9 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**.
2. Select the axis to be unmastered.
3. Press the **Unmaster** softkey. The mastering data of the axis are deleted.
4. Repeat steps 2 and 3 for all axes to be unmastered.

5.4 Calibration

5.4.1 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.

If the tool is a fixed tool, the type of calibration described here must not be used. A separate type of calibration must be used for fixed tools.
(>>> 5.4.2 "Calibration of an external TCP and fixed tool" page 95)

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

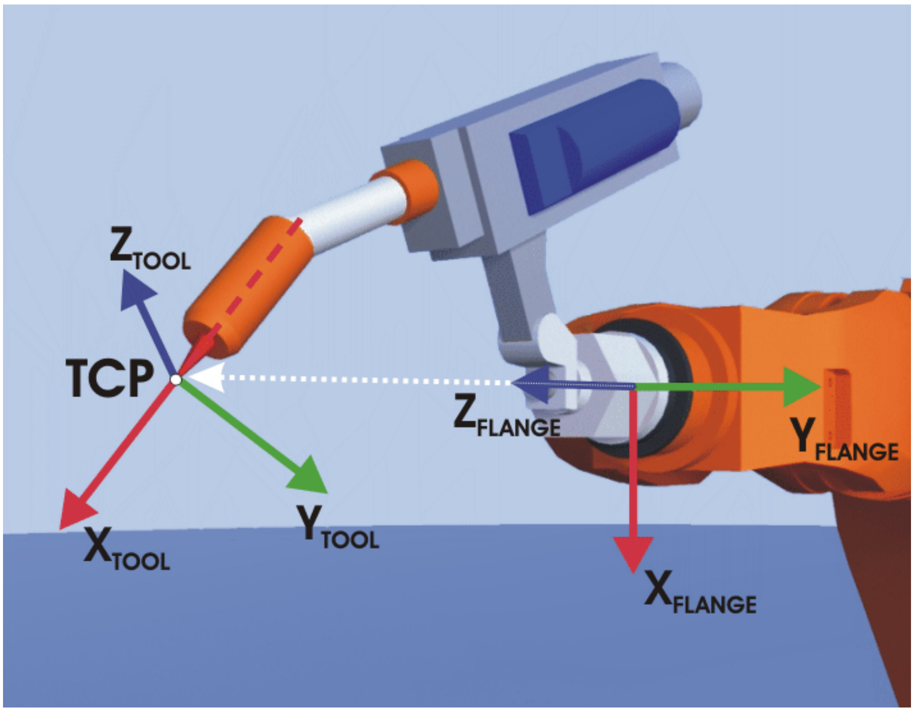


Fig. 5-7: TCP calibration principle

Overview

Tool calibration consists of 2 steps:

Step	Description
1	**Definition of the origin of the TOOL coordinate system** The following methods are available: ■ XYZ 4-Point (>>> 5.4.1.1 "TCP calibration: XYZ 4-Point method" page 90) ■ XYZ Reference (>>> 5.4.1.2 "TCP calibration: XYZ Reference method" page 91)
2	**Definition of the orientation of the TOOL coordinate system** The following methods are available: ■ ABC World (>>> 5.4.1.3 "Defining the orientation: ABC World method" page 92) ■ ABC 2-Point (>>> 5.4.1.4 "Defining the orientation: ABC 2-Point method" page 93)

If the calibration data are already known, they can be entered directly (>>> 5.4.1.5 "Numeric input" page 94).

5.4.1.1 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 sufficiently different from one another.

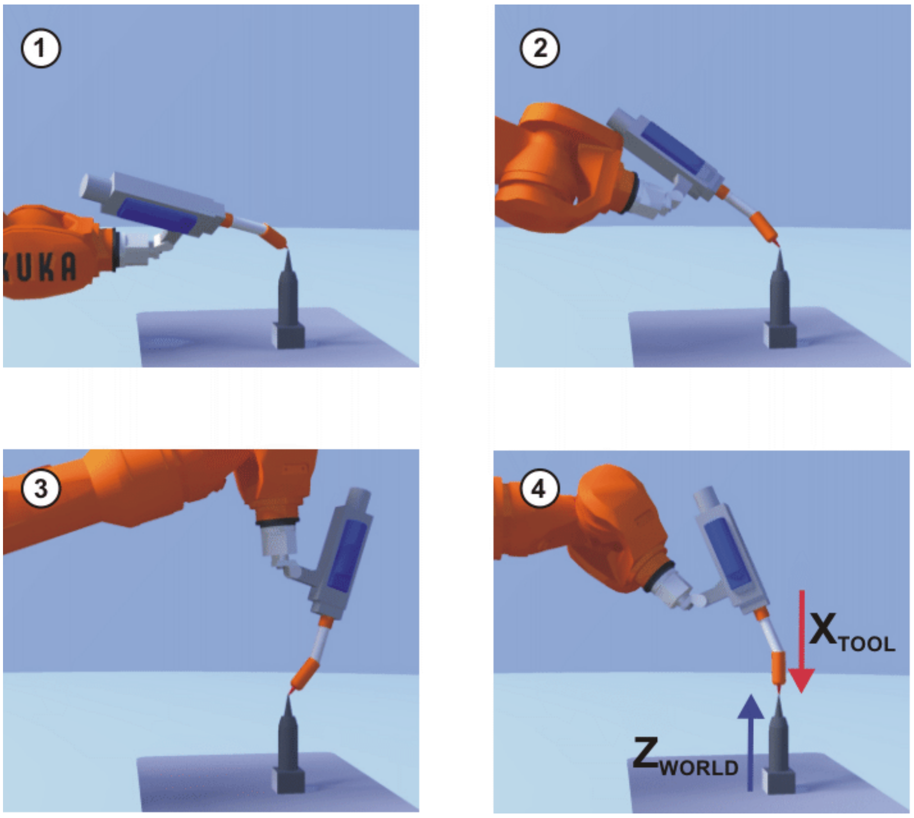


Fig. 5-8: XYZ 4-Point method

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 **OK**.
3. Move the TCP to a reference point. Confirm with **OK**.
4. Move the TCP to the reference point from a different direction. Confirm with **OK**.
5. Repeat step 4 twice.
6. Press **Save**.

5.4.1.2 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.

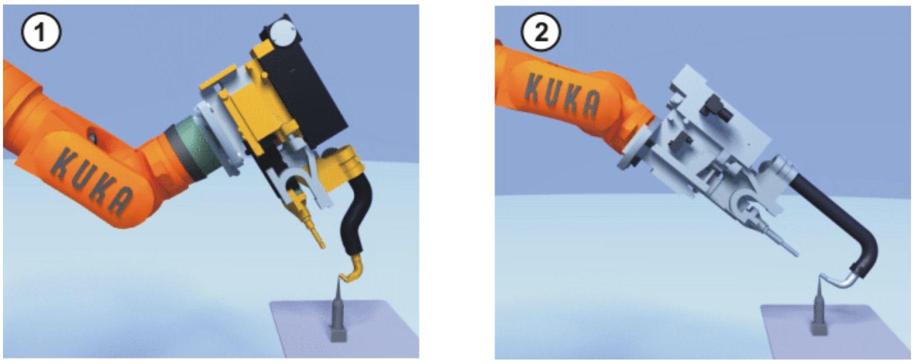


Fig. 5-9: XYZ Reference method

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 **OK**.
3. Enter the TCP data of the calibrated tool. Confirm with **OK**.
4. Move the TCP to a reference point. Confirm with **OK**.
5. Move the tool away and remove it. Mount the new tool.
6. Move the TCP of the new tool to the reference point. Confirm with **OK**.
7. Press **Save**.

5.4.1.3 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

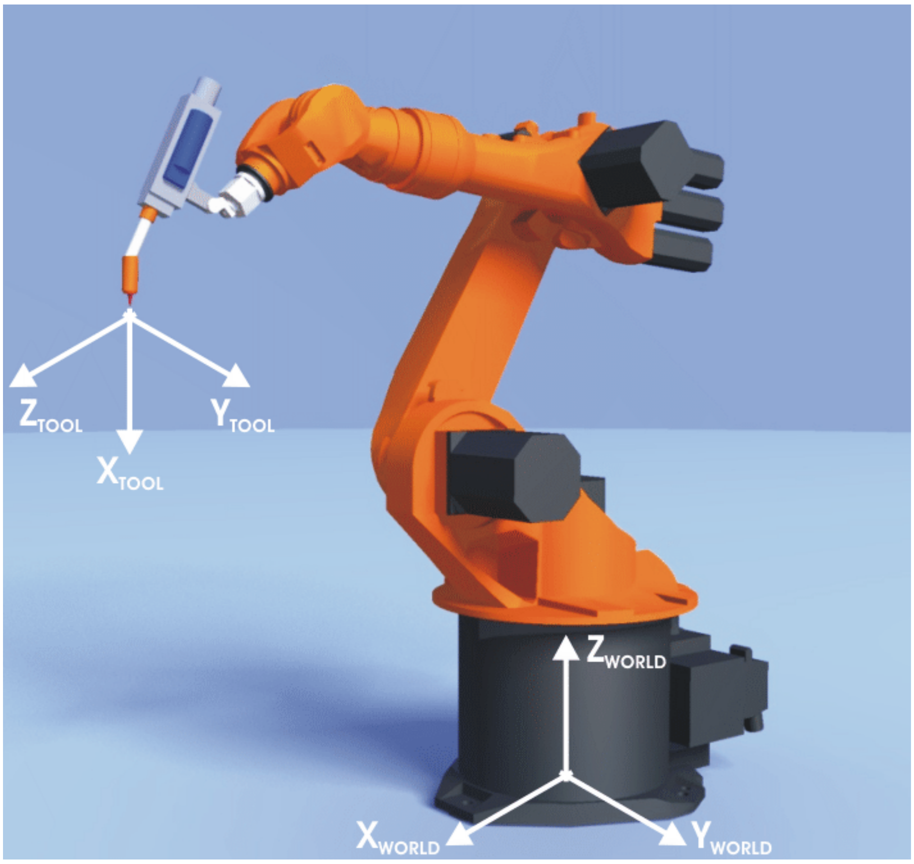


Fig. 5-10: ABC World method

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 **OK**.
3. Select a variant in the box **5D/6D**. Confirm with **OK**.
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. Confirm with **OK**.
6. Press **Save**.

5.4.1.4 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.

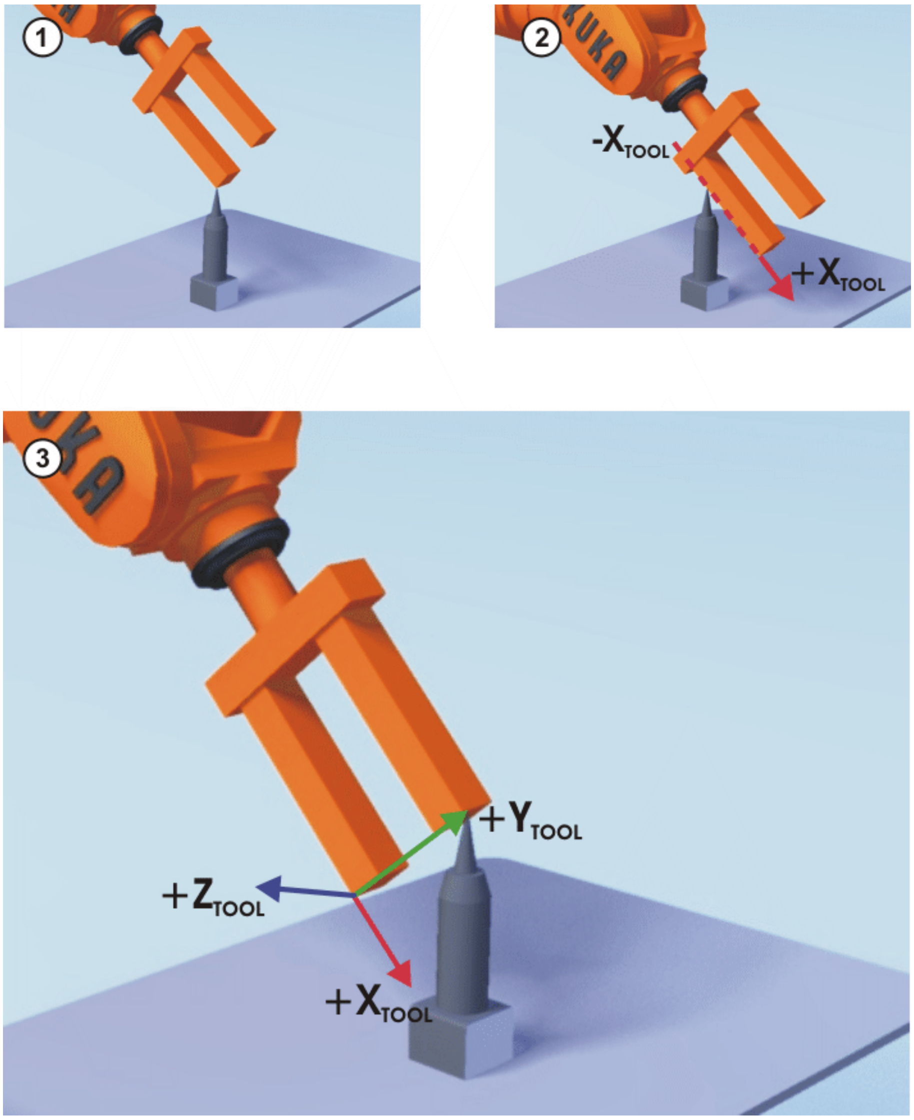


Fig. 5-11: ABC 2-Point method

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 **OK**.
3. Move the TCP to any reference point. Confirm with **OK**.
4. By default, the tool direction is the X axis. Move the tool so that the reference point on the X axis has a negative X value (i.e. move against the tool direction). Confirm with **OK**.
5. Move the tool so that the reference point in the XY plane has a negative Y value. Confirm with **OK**.
6. Press **Save**.

5.4.1.5 Numeric input

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 **OK**.
3. Enter data. Confirm with **OK**.
4. Press **Save**.

5.4.2 Calibration of an external TCP and fixed tool

Overview

Calibration of a fixed tool consists of 2 steps:

Step	Description
1	**Calibration of the TCP of the fixed tool** (>>> 5.4.2.1 "Calibration of an external TCP" page 95) If the calibration data are already known, they can be entered directly. (>>> 5.4.2.2 "Entering the external TCP numerically" page 97)
2	**Calibration of the workpiece** The following methods are available: ■ Direct method (>>> 5.4.2.3 "Workpiece calibration: direct method" page 97) ■ Indirect method (>>> 5.4.2.4 "Workpiece calibration: indirect method" page 98)

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.

5.4.2.1 Calibration of 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.

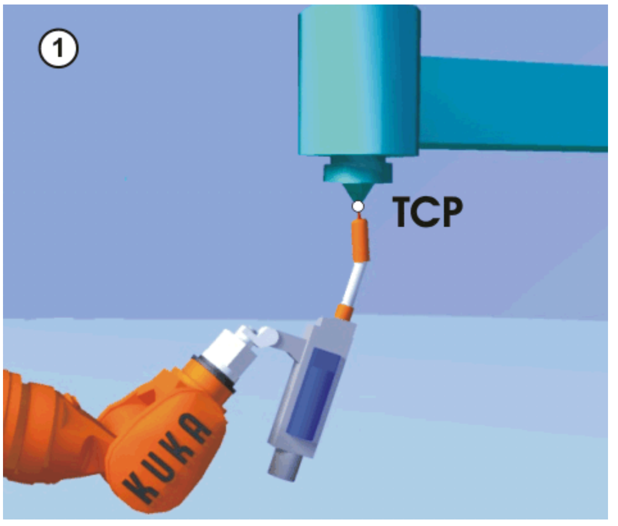


Fig. 5-12: Moving to the external TCP

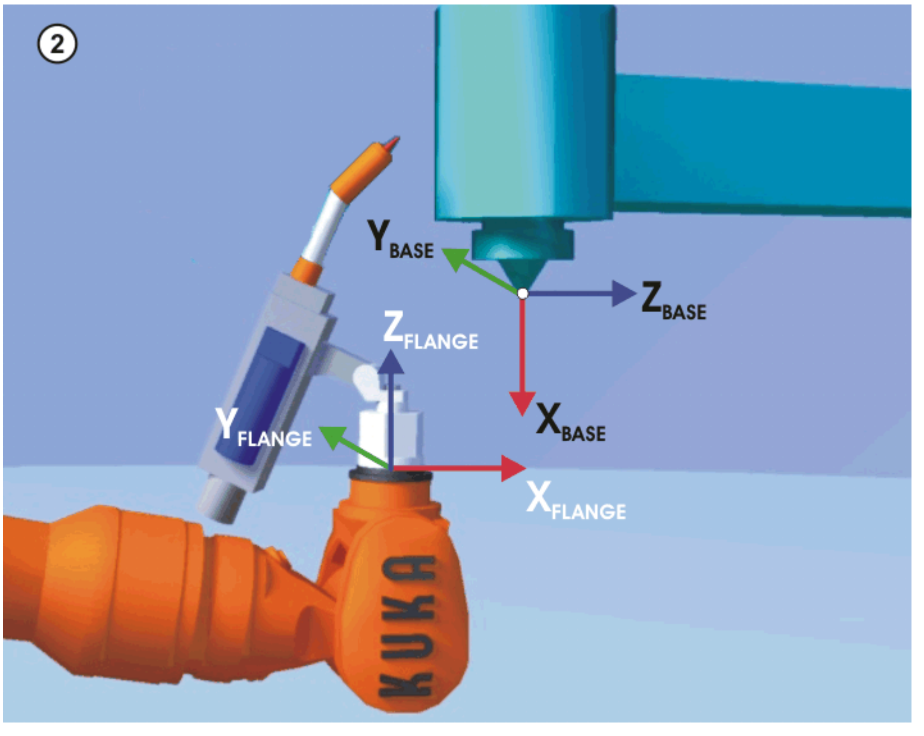


Fig. 5-13: Aligning the coordinate systems parallel to one another

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 **OK**.
3. Enter the number of the calibrated tool. Confirm with **OK**.
4. Select a variant in the box **5D/6D**. Confirm with **OK**.
5. Move the TCP of the calibrated tool to the TCP of the fixed tool. Confirm with **OK**.
6. If **5D** is selected:

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

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

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}

7. Confirm with **OK**.
8. Press **Save**.

5.4.2.2 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)

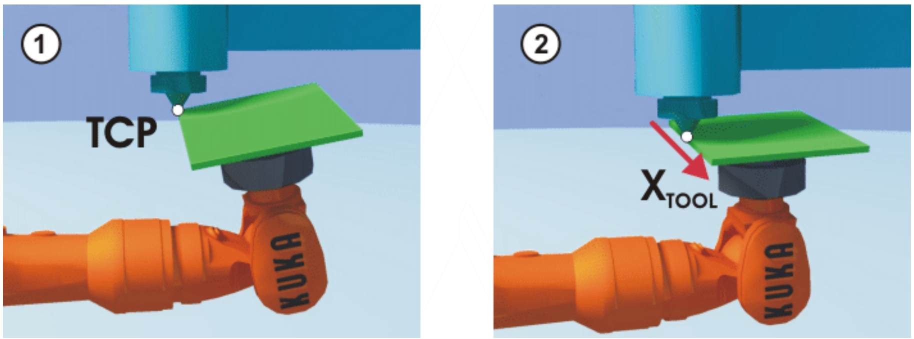
Procedure

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

5.4.2.3 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.



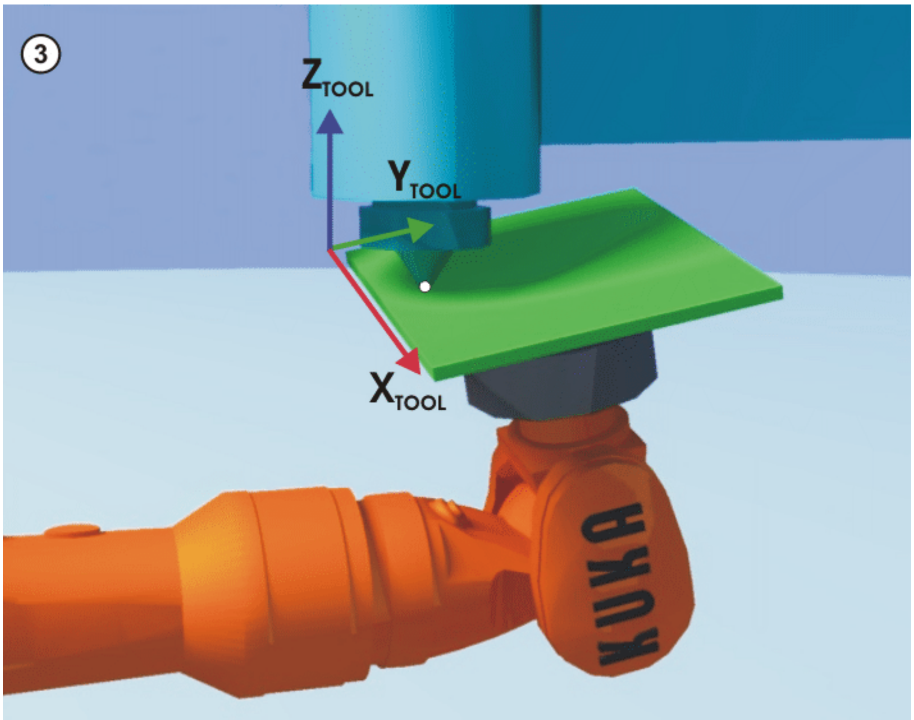


Fig. 5-14: Workpiece calibration: direct method

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 **OK**.
3. Enter the number of the fixed tool. Confirm with **OK**.
4. Move the origin of the workpiece coordinate system to the TCP of the fixed tool.

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

   Confirm with **OK**.
6. Move a point with a positive Y value in the XY plane of the workpiece co-ordinate system to the TCP of the fixed tool.

   Confirm with **OK**.
7. Press **Save**.

5.4.2.4 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 work-piece.

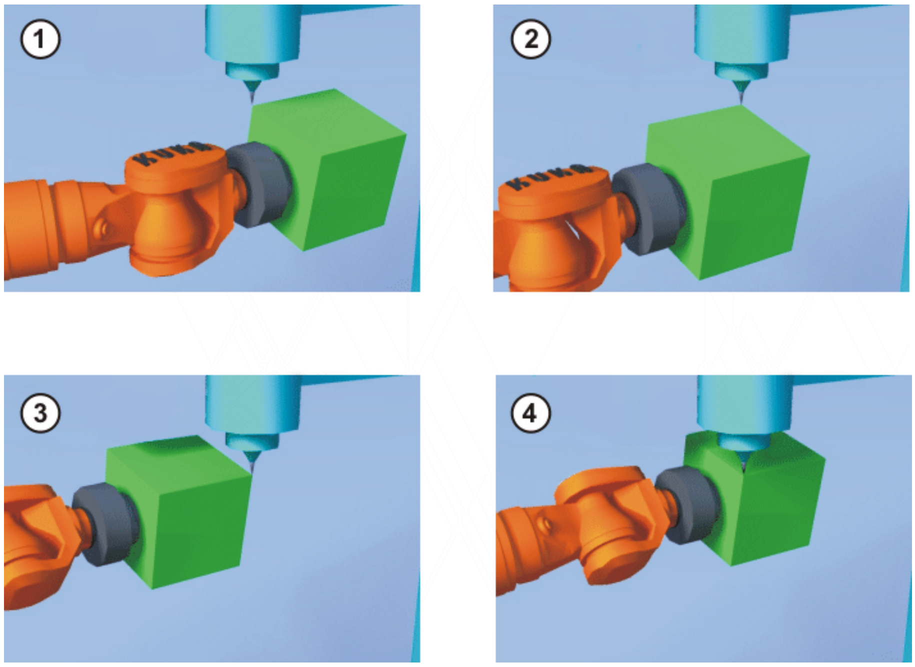


Fig. 5-15: Workpiece calibration: indirect method

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. Confirm with **Continue**.
5. Repeat step 4 three times.
6. Press **Save**.

5.4.3 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. (>>> 5.4.2 "Calibration of an external TCP and fixed tool" page 95)

Advantages of base calibration:

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

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

Overview

There are 2 ways of calibrating a base:

- 3-point method (>>> 5.4.3.1 "3-point method" page 100)
- Indirect method (>>> 5.4.3.2 "Indirect method" page 101)

If the calibration data are already known, they can be entered directly (>>> 5.4.3.3 "Numeric input" page 102).

5.4.3.1 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.

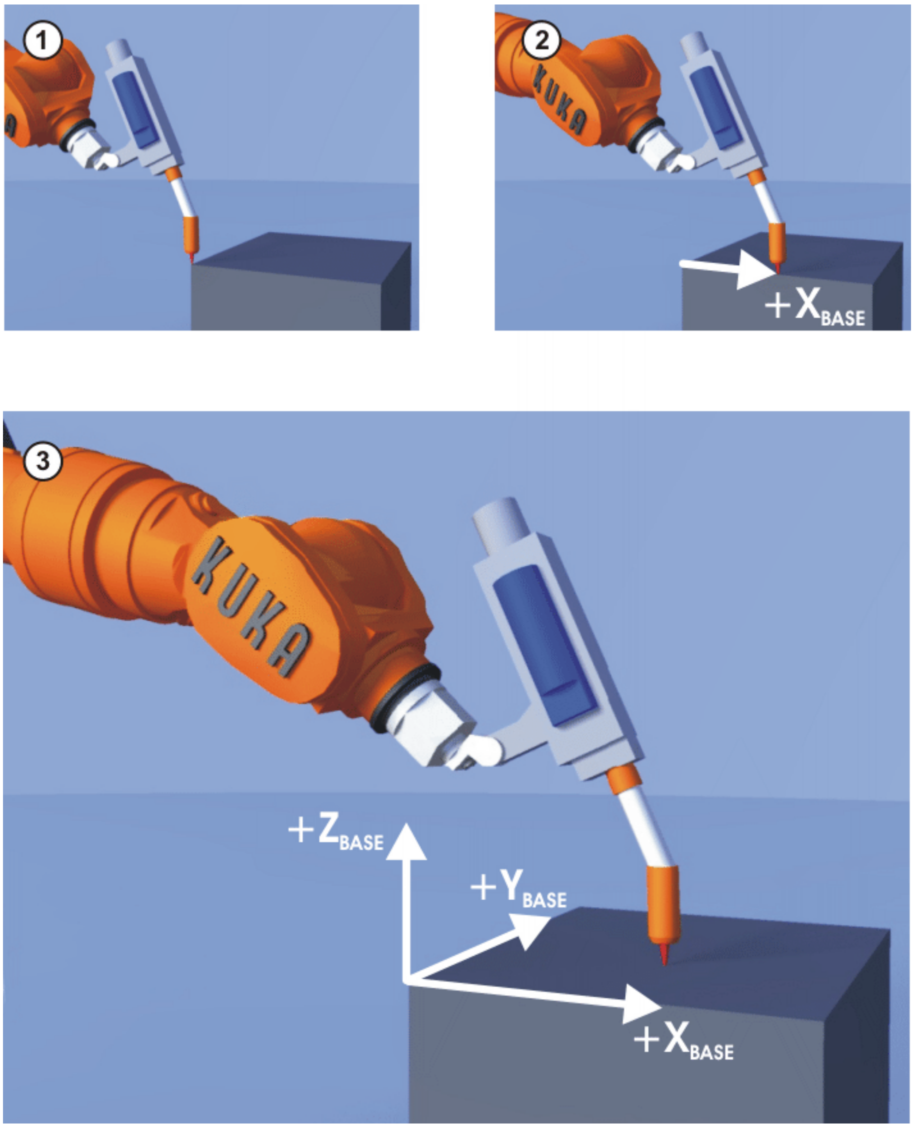


Fig. 5-16: 3-point method

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 **OK**.

3. Enter the number of the mounted tool. Confirm with **OK**.
4. Move the TCP to the origin of the new base. Confirm with **OK**.
5. Move the TCP to a point on the positive X axis of the new base. Confirm with **OK**.
6. Move the TCP to a point in the XY plane with a positive Y value Confirm with **OK**.
7. Press **Save**.

5.4.3.2 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.

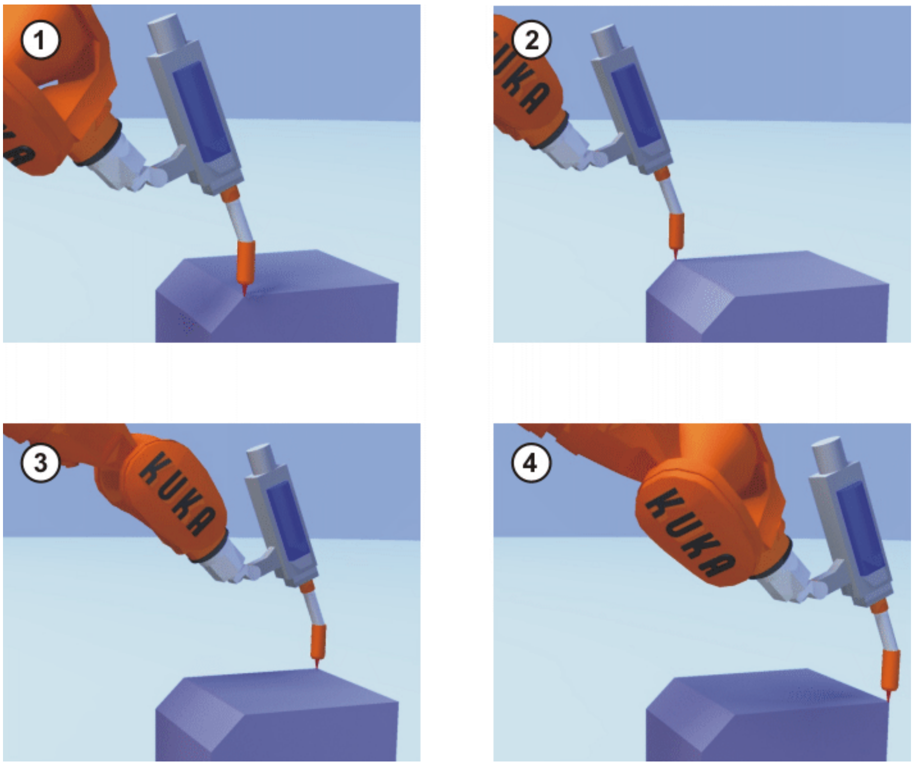


Fig. 5-17: Indirect method

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 **OK**.
3. Enter the number of the mounted tool. Confirm with **OK**.
4. Enter the coordinates of a known point in the new base and move the TCP to this point. Confirm with **OK**.
5. Repeat step 4 three times.
6. Press **Save**.

5.4.3.3 Numeric input

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 **OK**.
3. Enter data. Confirm with **OK**.
4. Press **Save**.

5.5 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 to the robot system.

5.5.1 Loads on the robot

Description

Various loads can be mounted on the robot:

- Payload on the flange
- Supplementary load on axis 3
- Supplementary load on axis 2
- Supplementary load on axis 1

All loads added together give the overall load.

There is a payload diagram for every robot. It can be used to check quickly whether the payload could be suitable for the robot. The diagram is not, however, a substitute for checking the payload with KUKA.Load.

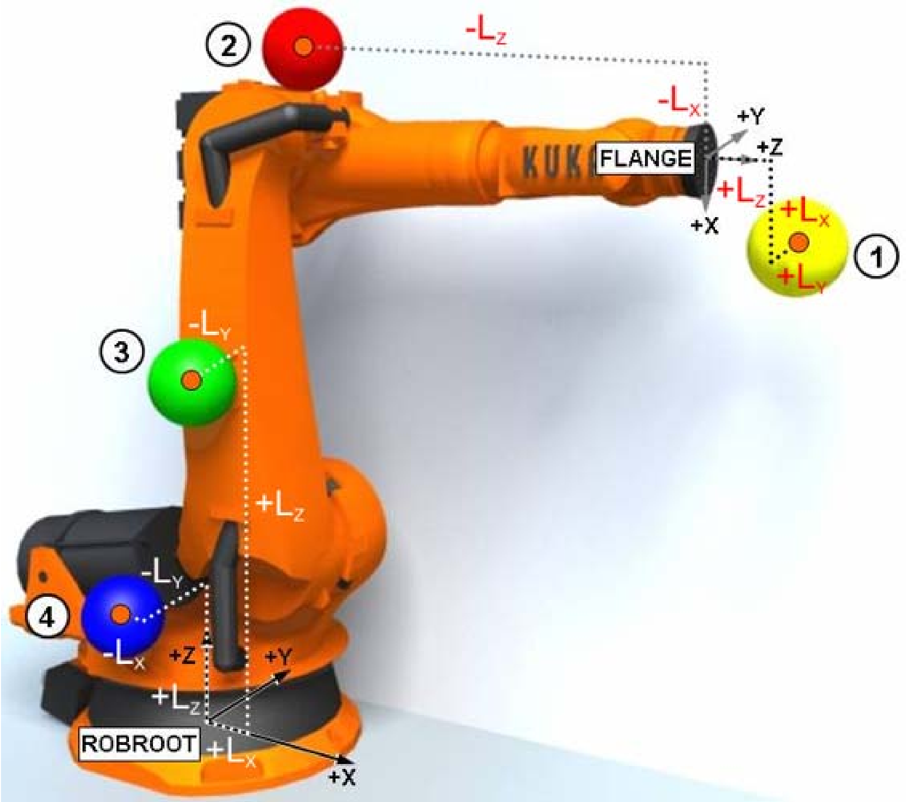


Fig. 5-18: Loads on the robot

1	Payload	3	Supplementary load on axis 2
2	Supplementary load on axis 3	4	Supplementary load on axis 1

Parameters

The load data are defined using the following parameters:

Parameter		Unit
Mass	m	kg lb
Distance to the center of gravity	L_x, L_y, L_z	mm in^2
Mass moments of inertia at the center of gravity	I_{xx}, I_{yy}, I_{zz}	kg m^2 in^2

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

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

Sources

Load data can be obtained from the following sources:

- Manufacturer information
- Manual calculation

- Software option KUKA.LoadDetect
- CAD programs

5.5.2 Static overloading of the robot

Description

If the permissible motor braking torques or the motor holding torques under servo control are exceeded while the robot is at a standstill, this is referred to as static overloading of the robot. This overloading can be prevented by means of the following measures:

- Shifting the position of the center of gravity towards the flange center point
- Using a robot with a higher rated payload
- Reducing the mass/weight

The KUKA Robot Group must always be consulted in the case of overloading.

5.5.3 Dynamic overloading of the robot

Description

If the maximum permissible kinetic energy values are exceeded by means of excessive mass moments of inertia, this is referred to as dynamic overloading of the robot. This overloading can be prevented by means of the following measures:

- Reduce the mass moments of inertia by:
  - Using a more geometrically compact load
  - Reducing the mass
  - Using a robot with a higher rated payload

The KUKA Robot Group must always be consulted in the case of overloading.

5.5.4 KUKA.LoadDetect

Description

KUKA.LoadDetect can be used to determine payload data exactly.

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.

KUKA.Load Detect can only be used for payloads over 40% of the rated payload.

5.5.5 Verifying load data

The determined load data must be verified. The following options are available:

- **Payload diagram**
  The payload diagram can be used to check quickly whether the payload could be suitable for the robot. The diagram is not, however, a substitute for checking the payload with KUKA.Load.
  The payload diagram can be found in the robot operating instructions.

- **KUKA.Load** program

  All load data (payload and supplementary loads) must be checked with KUKA.Load.

  More detailed information is contained in the KUKA.Load documentation. KUKA.Load can be downloaded free of charge, complete with the documentation, from the KUKA website www.kuka.com.

If the results of the test with KUKA.Load are negative, it may nonetheless, in certain cases, be possible to use the load on this robot. KUKA Roboter GmbH must be consulted in such cases. (>>> 11.2 "KUKA Customer Support" page 247)

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 to the robot system.

5.5.6 Entering payload data

Precondition

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

  (>>> 5.5.5 "Verifying load data" page 104)

Procedure

1. Select the menu **Setup** > **Measure** > **Tool** > **Payload data**.
2. Enter the number of the tool. Confirm with **OK**.
3. Enter the payload data. Confirm with **OK**.
4. Press **Save**.

5.5.7 Entering supplementary load data

Precondition

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

  (>>> 5.5.5 "Verifying load data" page 104)

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 **OK**.
3. Enter the load data. Confirm with **OK**.
4. Press **Save**.

5.6 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.

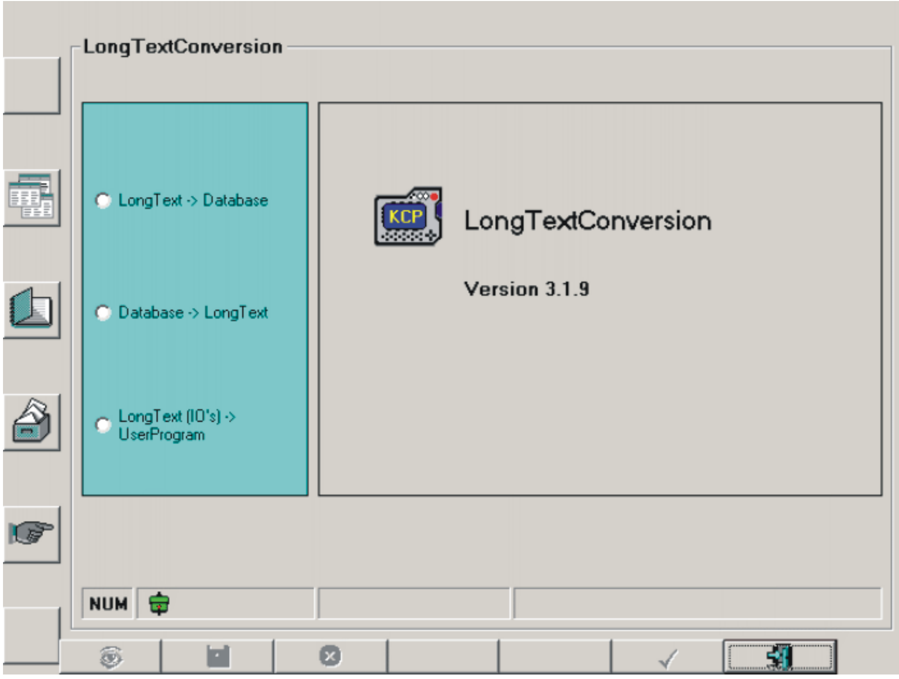


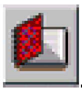
Fig. 5-19: Long text conversion window

5.6.1 Saving long text names

Precondition

- Expert user group

Procedure

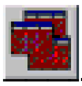
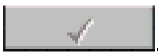
1. Select the menu sequence **Setup** > **Service** > **Long text**.
2. Press the status key .
3. Enter the path and name of the text file to be generated.
4. Press the softkey .
5. To close the window, press the  softkey.

5.6.2 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.

5.6.3 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**.

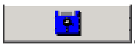
- Active: All files in the selected directory are updated.
- Inactive: Individual files can be selected for updating in the selected directory.

5.6.4 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

Keyword	Number	Meaning
NoticeText	1 … 32	Cyclical flag
FlagText	1 … 999	Flag
TimerText	1 … 20	Timer
CounterText	1 … 20	Counter
ANIN_	1 … 32	Analog input
ANOUT_	1 … 32	Analog output

6 Configuration

6.1 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

Precondition

- The technology package whose status keys are to be displayed must be installed.

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. If required, repeat steps 2 to 4 for an additional tool or base.
6. Save the changes and close the window with **OK**.

6.4 Force cold start

Procedure

- Select the menu sequence **Configure** > **On/Off Options** > **Force cold start**.

  The next time the controller is run up a cold start is carried out.

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** > **Tools** > **Change password**.
2. Select the user group for which the password is to be changed.

   The following softkeys are available:

   - **User**
   - **Expert**
   - **Administrator**
3. Enter the old password. Enter the new password twice. For security reasons, the entries are displayed encrypted.
4. Press the **OK** softkey.

   The new password is valid immediately.

6.7 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.

(>>> 6.7.1 "Configuring Cartesian workspaces" page 110)

(>>> 6.7.2 "Configuring axis-specific workspaces" page 113)

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

6.7.1 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

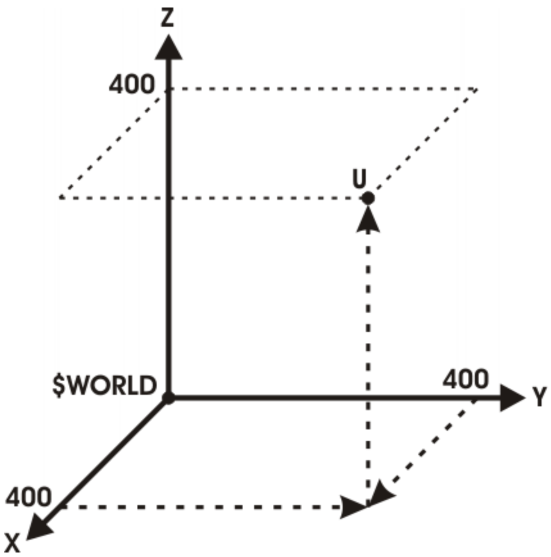


Fig. 6-1: Cartesian workspace, origin U

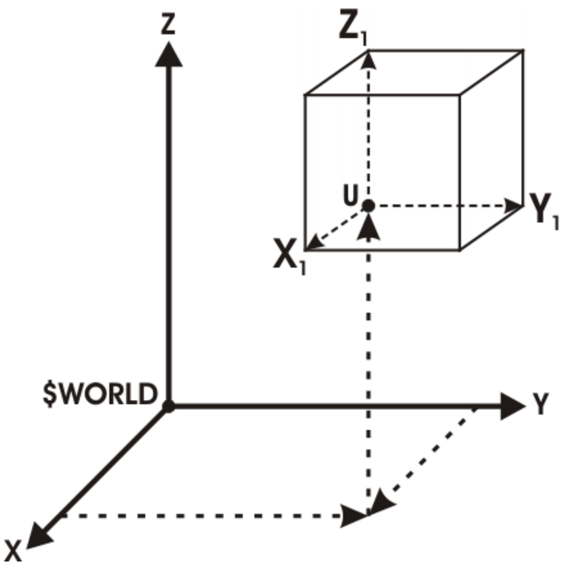


Fig. 6-2: Cartesian workspace, dimensions

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.

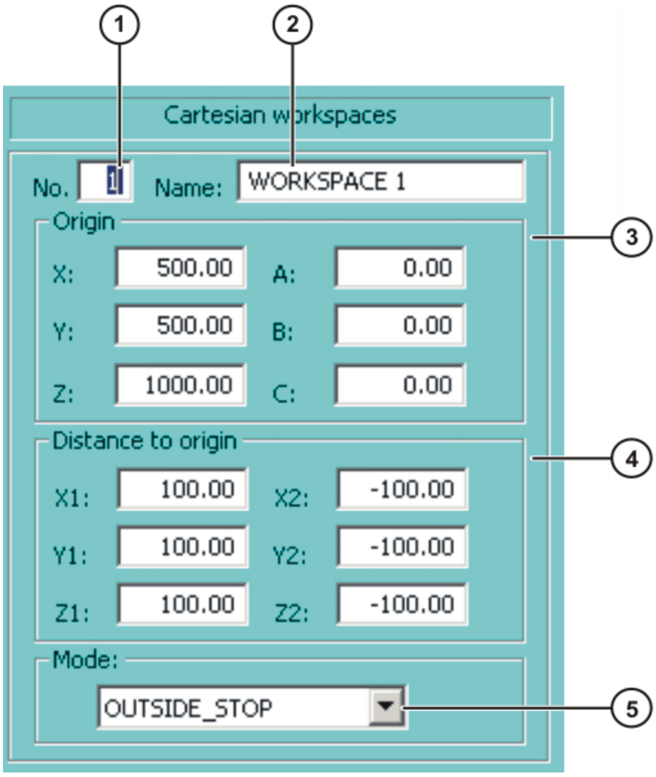


Fig. 6-3: Cartesian workspaces window

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 (>>> 6.7.3 "Mode for workspaces" page 115)

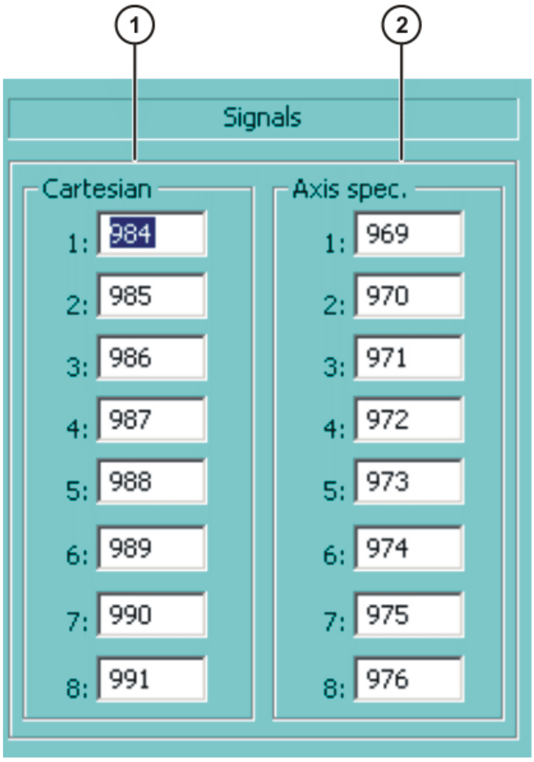


Fig. 6-4: Signals window

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.

6.7.2 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.

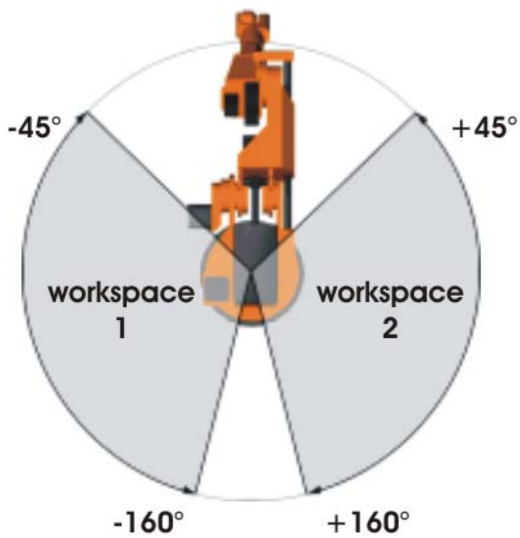


Fig. 6-5: Example of axis-specific workspaces for A1

If an axis-specific workspace is violated during jogging, ramp-down braking is carried out. Otherwise, dynamic braking is carried out.

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.

Description

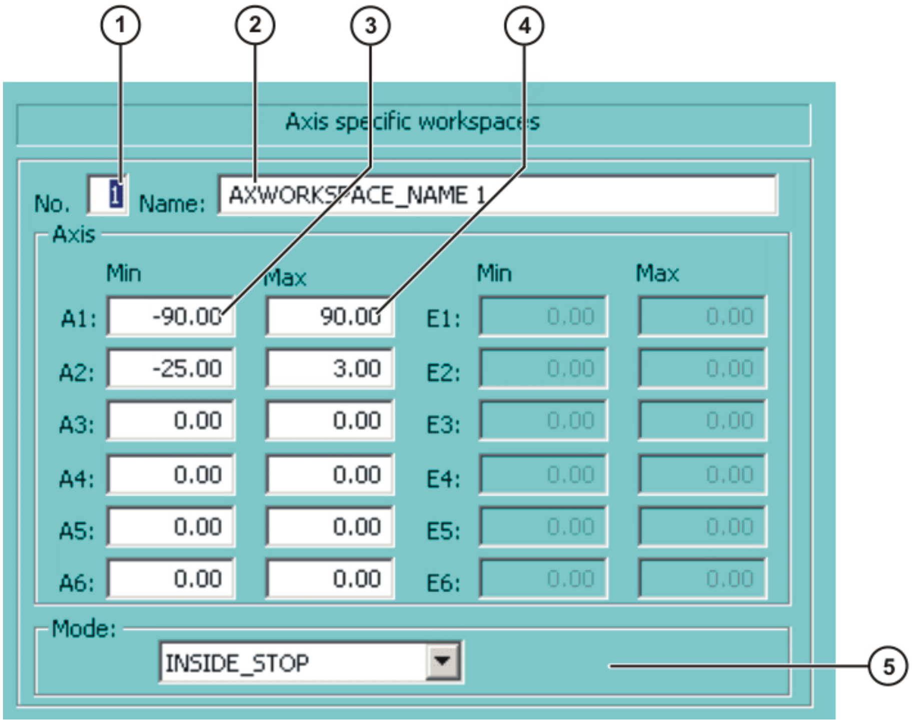


Fig. 6-6: Axis-specific workspaces window

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 (>>> 6.7.3 "Mode for workspaces" page 115)

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

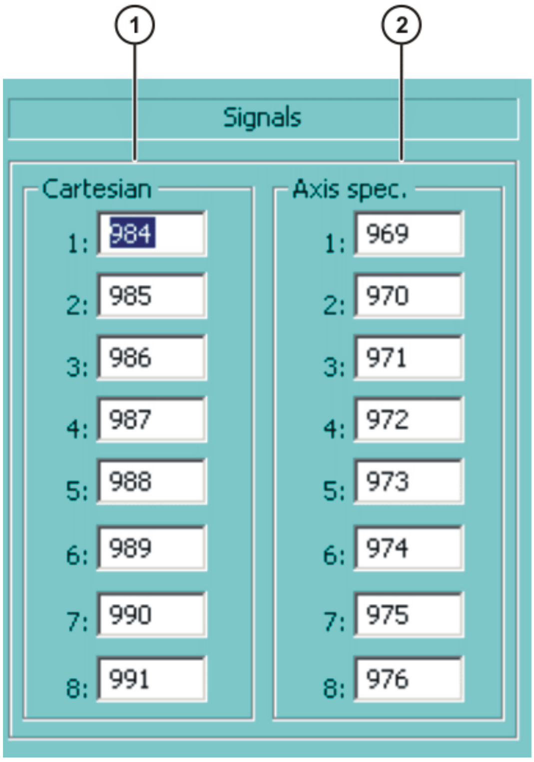


Fig. 6-7: Signals window

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.

6.7.3 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. (>>> 4.11 "Bypassing workspace monitoring" page 50)
#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. (>>> 4.11 "Bypassing workspace monitoring" page 50)

6.8 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**.

6.9 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

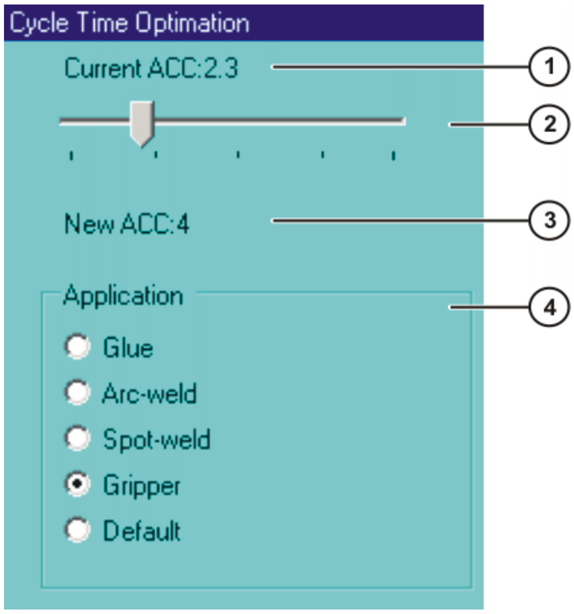


Fig. 6-8: Cycle Time Optimation window

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.

6.10 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

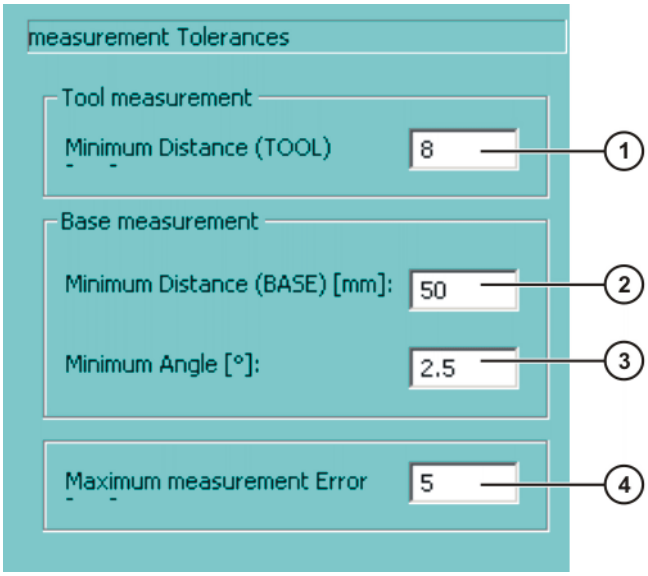


Fig. 6-9: Error tolerances

Item	Description	Range of values
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**.

6.11 Backward motion

There are two methods for executing a program backwards:

- TRACE (>>> 6.11.1 "TRACE method" page 118)
- SCAN (>>> 6.11.2 "SCAN method" page 120)

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.

6.11.1 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** (>>> 6.11.4 "TRACE section" page 121), 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.

6.11.2 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.

6.11.3 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

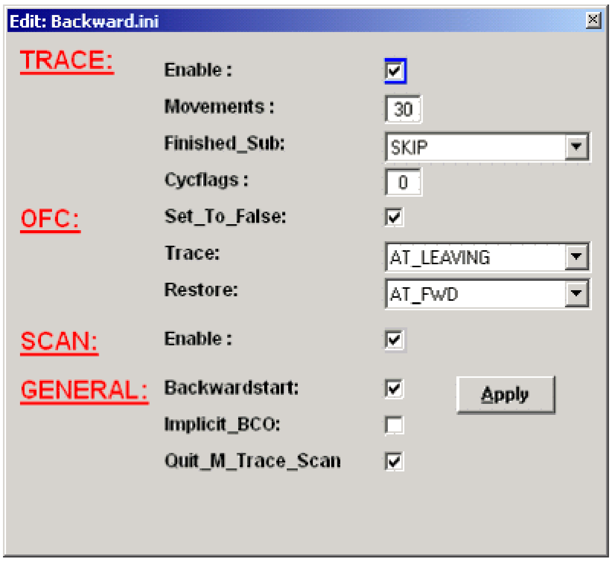


Fig. 6-10: Edit: Backward.ini window

- **TRACE** section (>>> 6.11.4 "TRACE section" page 121)
- **OFC** section (>>> 6.11.5 "OFC section" page 121)
- **SCAN** section (>>> 6.11.6 "SCAN section" page 122)
- **GENERAL** section (>>> 6.11.7 "GENERAL section" page 122)

6.11.4 TRACE section

Parameter	Range of values
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. Range of values: 0 to 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. Range of values: 0 to 26.

6.11.5 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	Range of values
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.

6.11.6 SCAN section

Parameter	Range of values
Enable	■ **TRUE**: SCAN method activated ■ **FALSE**: SCAN method deactivated

6.11.7 GENERAL section

Parameter	Range of values
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.

Parameter	Range of values
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.

6.12 Setting up a new user group and password

Description

In addition to the predefined user groups “User”, “Expert” and “Administrator”, up to 3 more user groups can 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.

> If a menu item or softkey has no identification number, it is available in the user group “User” or higher.

Identification numbers of the predefined user groups:

User group	Identification number
User	10
Expert	20
Administrator	30

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

Overview

Step	Description
1	Set up new user group. (>>> 6.12.1 "Example of setting up a new user group" page 124)
2	Define new user group as default user group (optional). (>>> 6.12.2 "Defining the default user group" page 126)
3	Set up password for new user group. (>>> 6.12.3 "Setting up a password for a new user group" page 126)

6.12.1 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” 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. (>>> 6.12.1.1 "Defining a user group" page 124)
2	Define position of the softkey. (>>> 6.12.1.2 "Defining the position of the softkey" page 125)
3	Enable function. (>>> 6.12.1.3 "Enabling the function" page 125)

6.12.1.1 Defining a user group

Procedure

1. in the file SoftKeyKuka.ini, go to the 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 in the example	Description	Range of values
MYGROUP_UMODE	Internal system designation.	Must end with _UMODE.
MyGroup	Designation of the softkey in the user interface.	Freely selectable
11	Identification number	0 … 30

6.12.1.2 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”.

6.12.1.3 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.

6.12.2 Defining the default user group

Description

When the system is booted, the user group “User” 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 (e.g. 11) or hexadecimal value (e.g. b) and press **OK**.
8. Close the **Registry editor** window.

When the system is next booted, the new default user group will automatically be activated.

6.12.3 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 110)

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

6.13 Configuring Automatic External

Description

If robot processes are to be controlled centrally (by a host computer or 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. (>>> 6.13.1 "Configuring CELL.SRC" page 127)
2	Configuration of the inputs/outputs of the Automatic External interface. (>>> 6.13.2 "Configuring Automatic External inputs/outputs" page 128)
3	Only if error numbers are to be transmitted to the higher-level controller: configuration of the P00.DAT file. (>>> 6.13.3 "Transmitting error numbers to the higher-level controller" page 134)

6.13.1 Configuring CELL.SRC

Description

In Automatic External mode, programs are called using the program CELL.SRC.

Overview

```
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	Calling of program numbers from the higher-level controller
15	CASE branch for program number = 1
16	Receipt of program number 1 is communicated to the host computer.
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**.

6.13.2 Configuring Automatic External inputs/outputs

Procedure

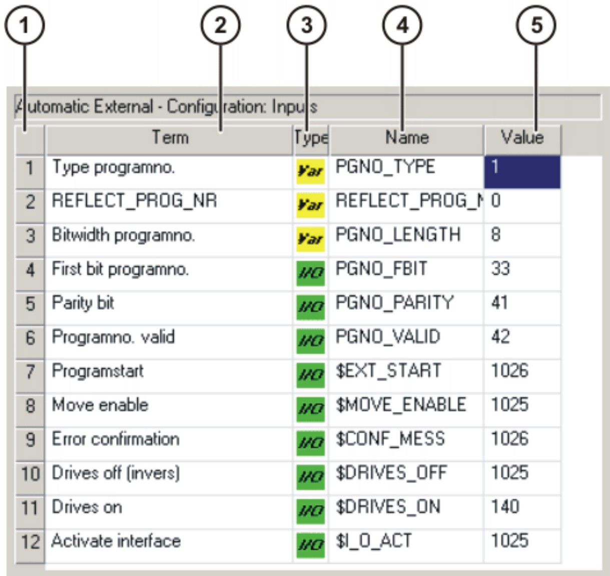
1. Select the menu sequence **Configure** > **I/O** > **Automatic External**.
2. In the **Value** column, select the cell to be edited and press the **Value** softkey.

3. Enter the desired value and save it by pressing the **OK** softkey.
4. Repeat steps 2 and 3 for all values to be edited. Exit the configuration by pressing the **Close** softkey.

Descriptions of the individual inputs and outputs:

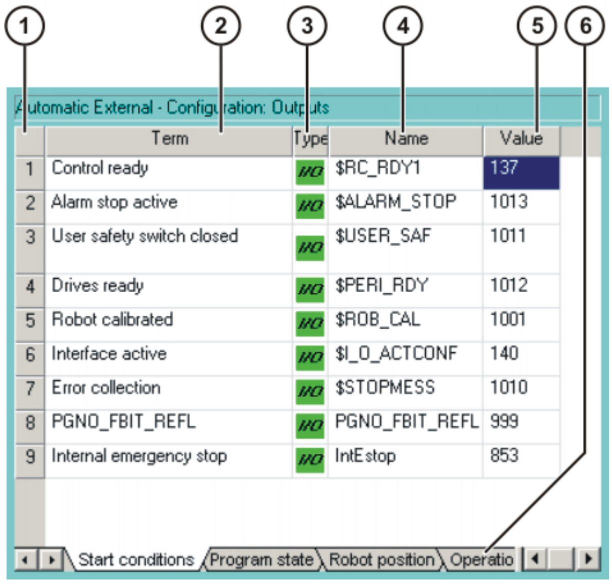
- (>>> 6.13.2.1 "Automatic External inputs" page 130)
- (>>> 6.13.2.2 "Automatic External outputs" page 132)

Description



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

Fig. 6-11: Configuring Automatic External inputs



	Term	Type	Name	Value
1	Control ready	I/O	$RC_RDY1	137
2	Alarm stop active	I/O	$ALARM_STOP	1013
3	User safety switch closed	I/O	$USER_SAF	1011
4	Drives ready	I/O	$PERI_RDY	1012
5	Robot calibrated	I/O	$ROB_CAL	1001
6	Interface active	I/O	$I_O_ACTCONF	140
7	Error collection	I/O	$STOPMESS	1010
8	PGNO_FBIT_REFL	I/O	PGNO_FBIT_REFL	999
9	Internal emergency stop	I/O	IntEstop	853

Fig. 6-12: Configuring Automatic External outputs

Column	Description
1	Number
2	Long text name of the input/output
3	Type ■ Green: Input/output ■ Yellow: Variable or system variable ($...)

Column	Description
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
Display	Switches to the Automatic External display. (>>> 4.12.5 "Displaying inputs/outputs for Automatic External" page 54)
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.

6.13.2.1 Automatic External inputs

PGNO_TYPE

Type: Variable

This variable defines the format in which the program number sent by the host computer 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. (>>> 6.13.2.2 "Automatic External outputs" page 132)

Val-ue	Description
0	Function deactivated.
1	Function activated.

PGNO_LENGTH

Type: Variable

This variable determines the number of bits defining the program number sent by the host computer. 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 host computer.

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 host computer.

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 host computer 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 host computer, 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 host computer 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].

6.13.2.2 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 EMERGENCY STOP

In the case of an EMERGENCY STOP, the nature of the EMERGENCY STOP can be recognized from the states of the outputs **$ALARM_STOP** and **Int. NotAus**.

- Both outputs are FALSE: the EMERGENCY STOP was triggered on the KCP.
- **$ALARM_STOP** is FALSE, **Int. NotAus** 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 host computer 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 host computer 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. (>>> 6.13.2.1 "Automatic External inputs" page 130)

The size of the output area depends on the number of bits defining the program number (PGNO_LENGTH).

If a program selected by the PLC is deselected by the user, the output area starting with PGNO_FBIT_REFL is set to FALSE. In this way, the PLC can prevent a program from being restarted manually.

PGNO_FBIT_REFL is also set to FALSE if the interpreter is situated in the CELL program.

Int. NotAus

This output is set to FALSE if the EMERGENCY STOP button on the KCP is pressed.

In the case of an EMERGENCY STOP, the nature of the EMERGENCY STOP can be recognized from the states of the outputs **$ALARM_STOP** and **Int. NotAus**.

- Both outputs are FALSE: the EMERGENCY STOP was triggered on the KCP.
- **$ALARM_STOP** is FALSE, **Int. NotAus** 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 host computer 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 host computer 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 host computer 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 host computer to determine whether or not the robot is situated within a sphere about the position saved in $POS_RET. The host computer 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, $EXTERN

These outputs are set when the corresponding operating mode is selected.

ERR_TO_PLC

By setting this output, the robot controller communicates to the host computer the fact that a controller or technology error has occurred.

Precondition: PLC_ENABLE must be set to TRUE in the file P00.DAT. (>>> 6.13.3 "Transmitting error numbers to the higher-level controller" page 134)

6.13.3 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 host computer 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.

6.13.4 Signal diagrams

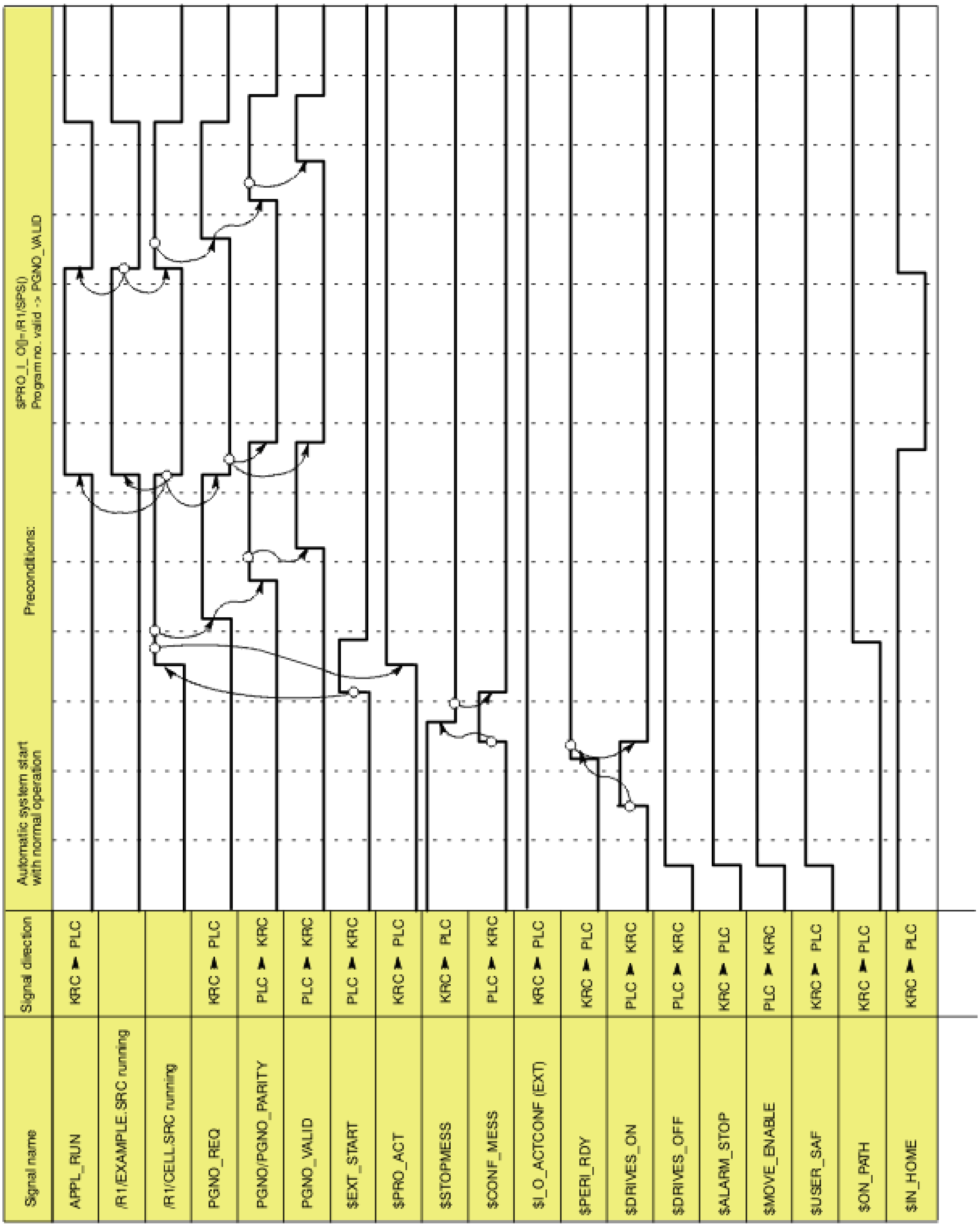


Fig. 6-13: Automatic system start and normal operation with program number acknowledgement by means of PGNO_VALID

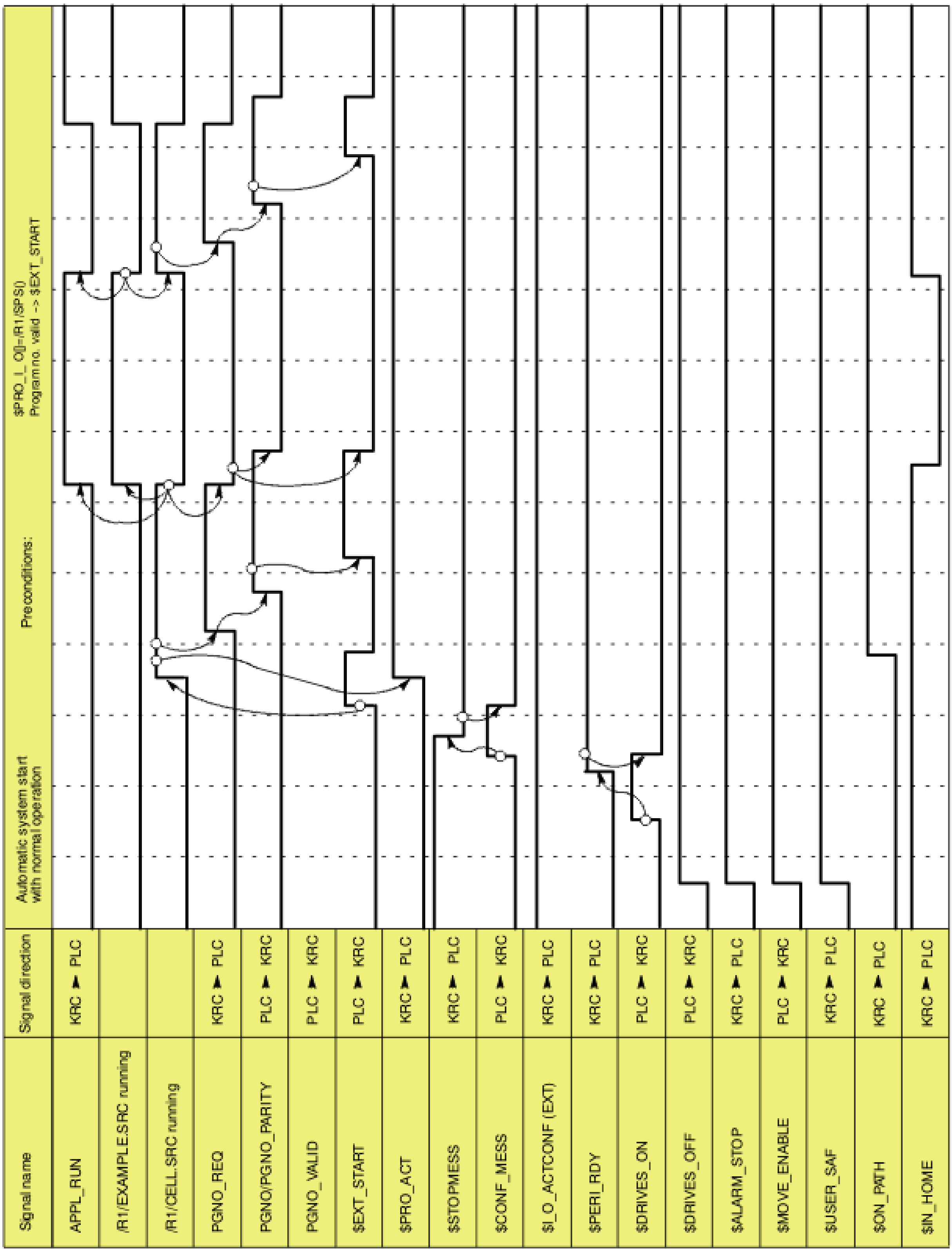


Fig. 6-14: Automatic system start and normal operation with program number acknowledgement by means of $EXT_START

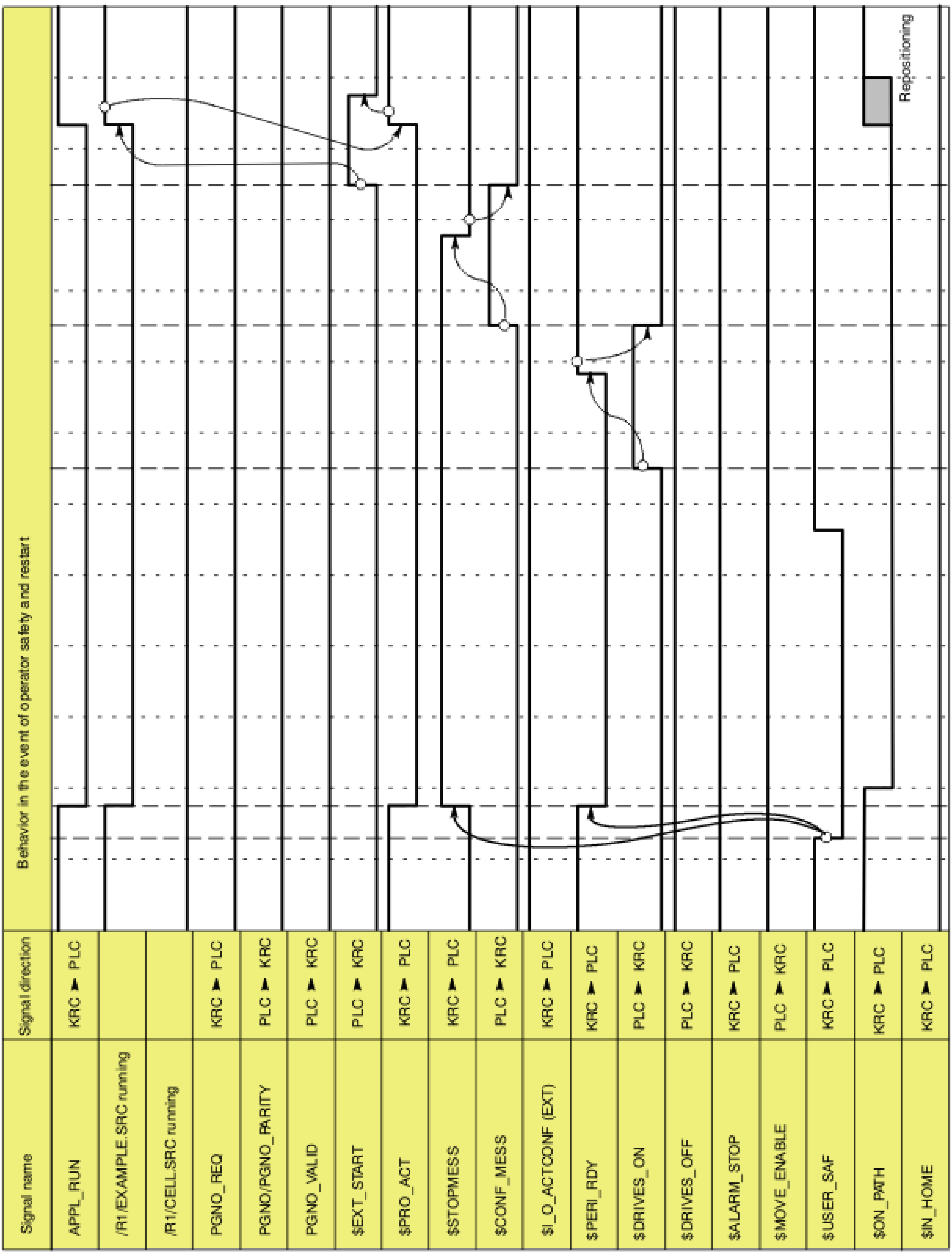


Fig. 6-15: Restart after dynamic braking (operator safety and restart)

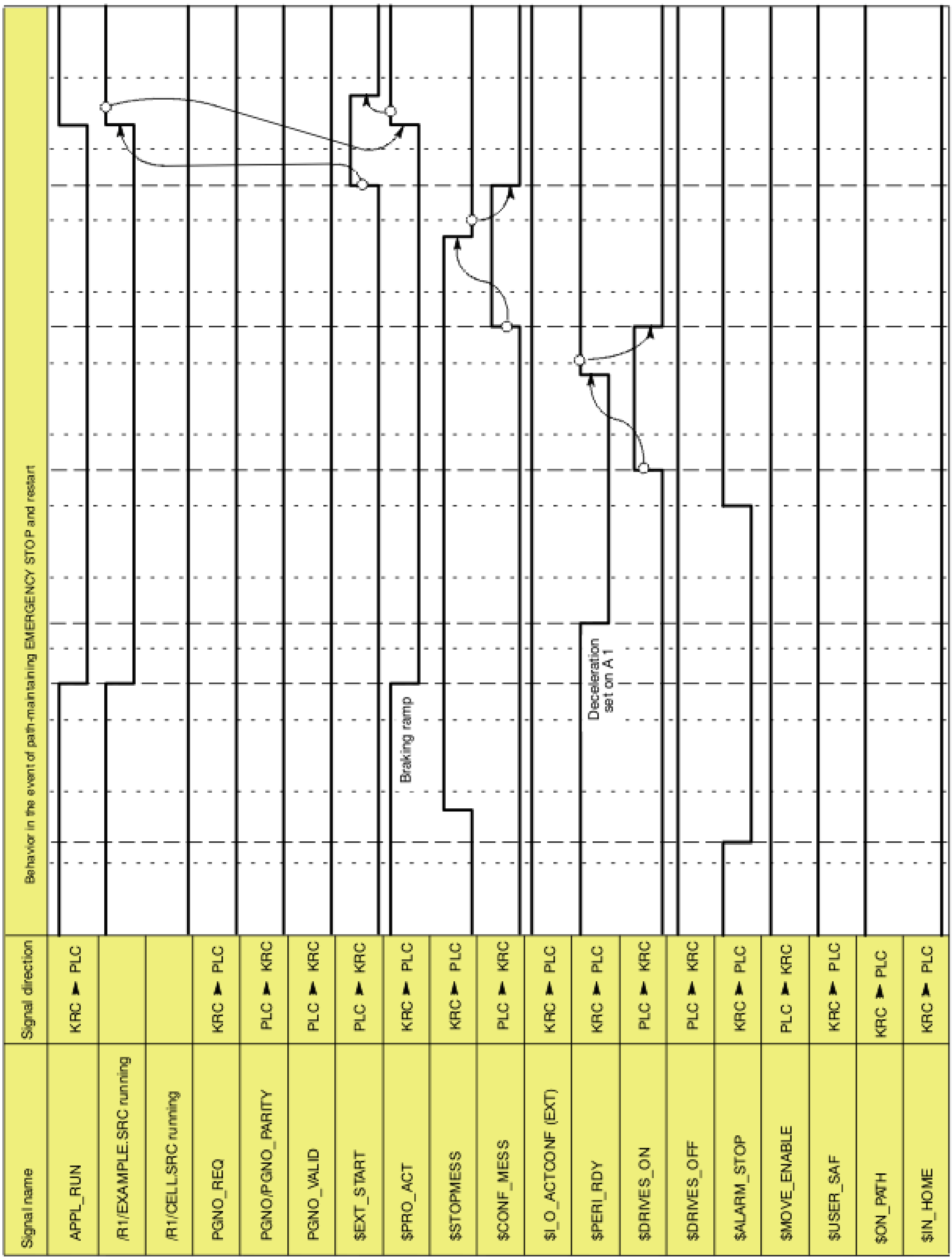


Fig. 6-16: Restart after path-maintaining EMERGENCY STOP

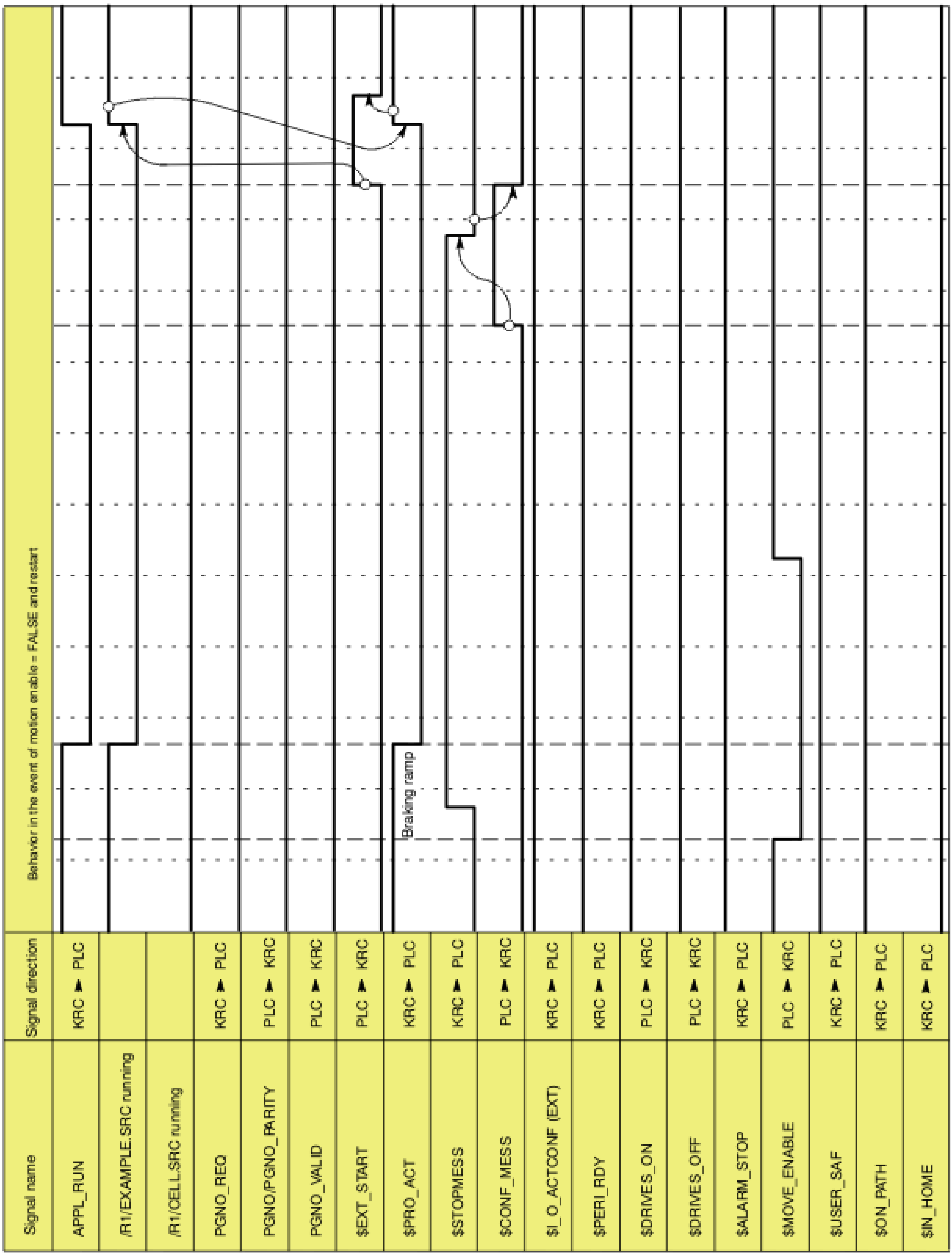


Fig. 6-17: Restart after motion enable

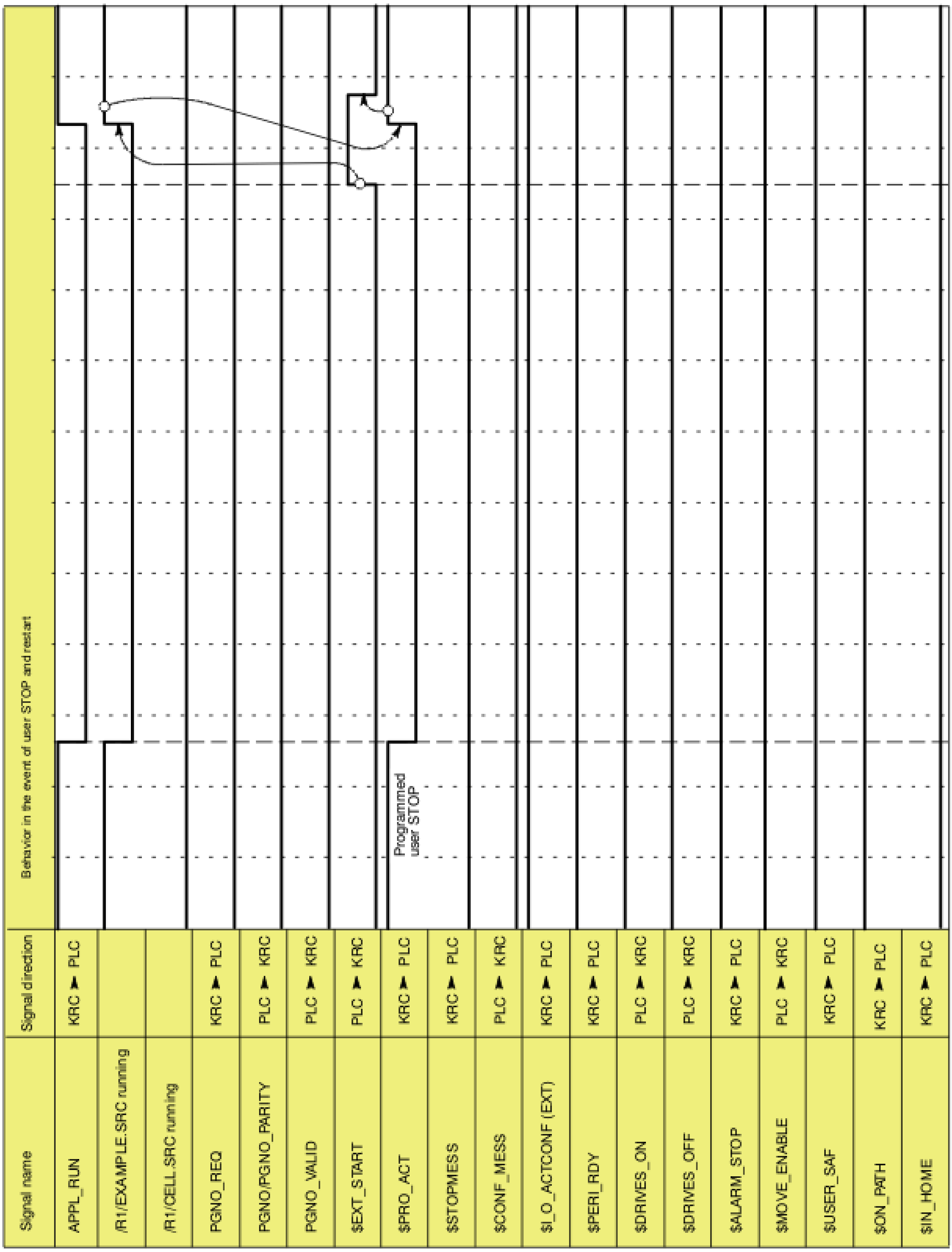


Fig. 6-18: Restart after user STOP

6.14 KRC Configurator

The KRC Configurator can be used to configure the Navigator and other KSS functions. Most functions can be specially configured for each user group.

The settings in the KRC Configurator are not checked for validity or plausibility.

Procedure

Call the KRC Configurator:

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. Repeat step 2 for all other tabs.
4. Exit the KRC Configurator by pressing **Exit**.

6.14.1 Operating the KRC Configurator

The following functions are available in each of the tabs:

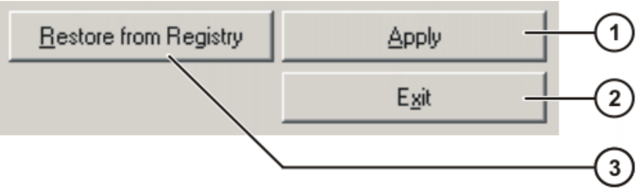
Select tab

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:



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**.

6.14.2 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.

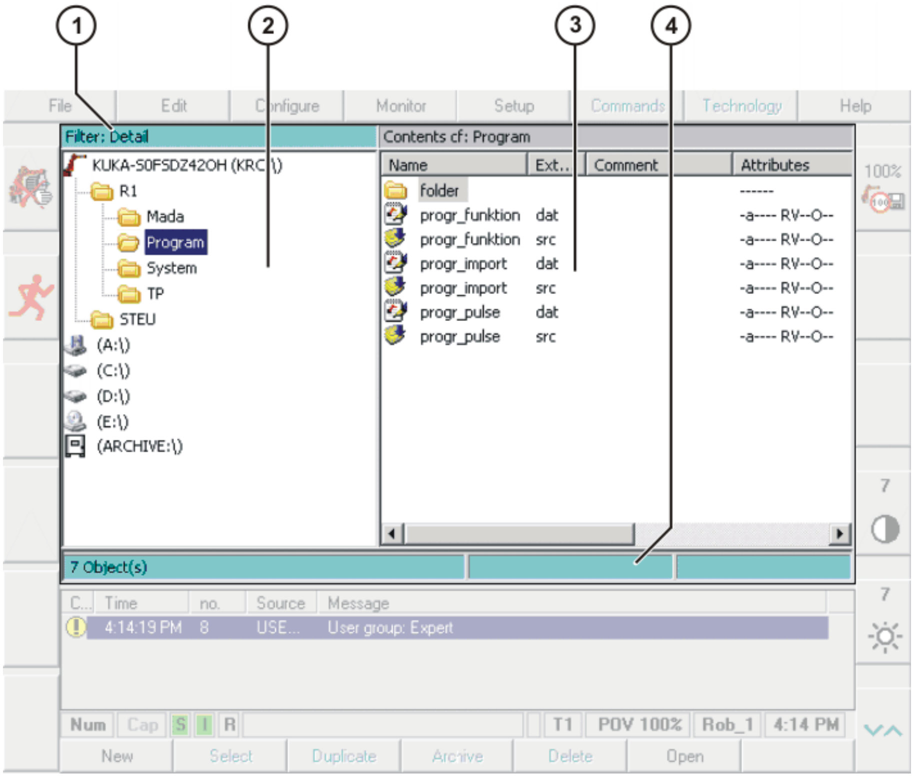


Fig. 6-19: Navigator

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.

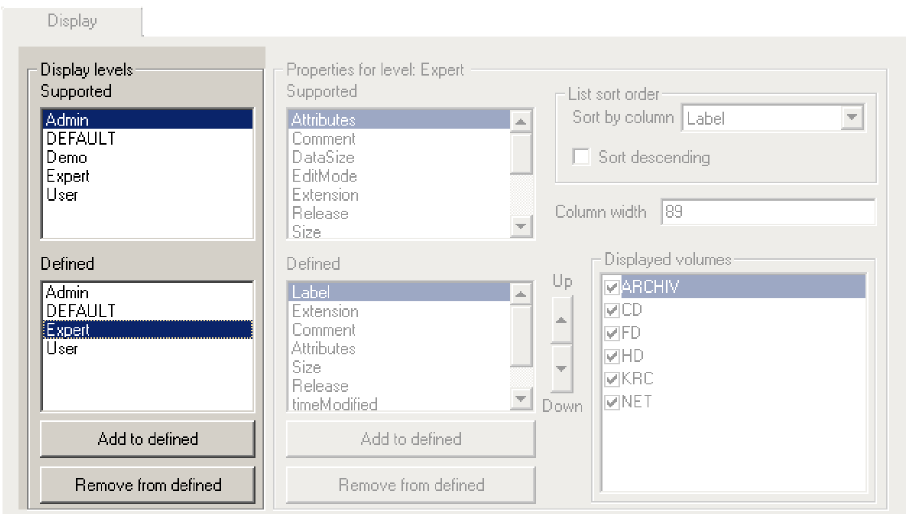


Fig. 6-20: Display tab, Display levels

Element	Description
Supported	Available user groups (cannot be changed)
Defined	User groups set up in the system. The user group for which the Navigator is to be configured must be selected.
Add to defined	Adds the user group selected in **Supported** to **Defined**.
Remove from defined	Removes the user group selected in **Defined**.

Description: Properties for level

The Navigator is configured for a user group here.

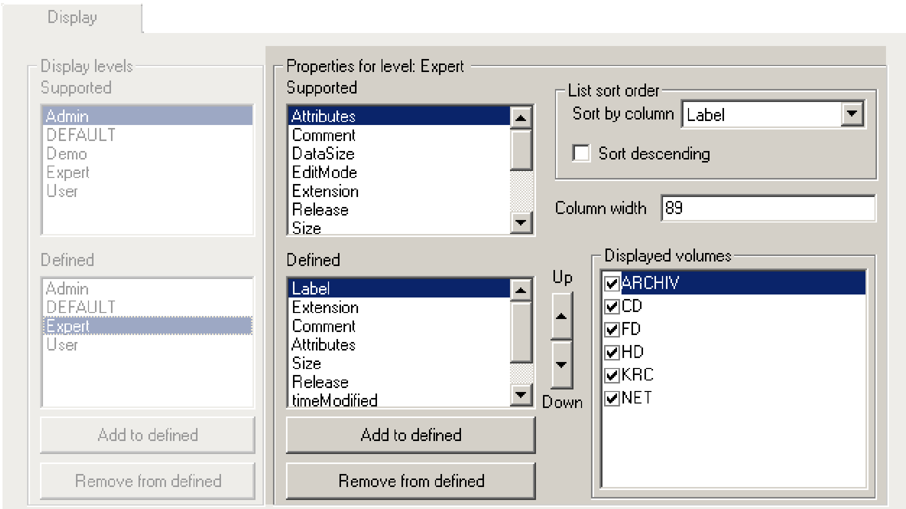


Fig. 6-21: Display tab, Properties for level

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.

6.14.3 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.

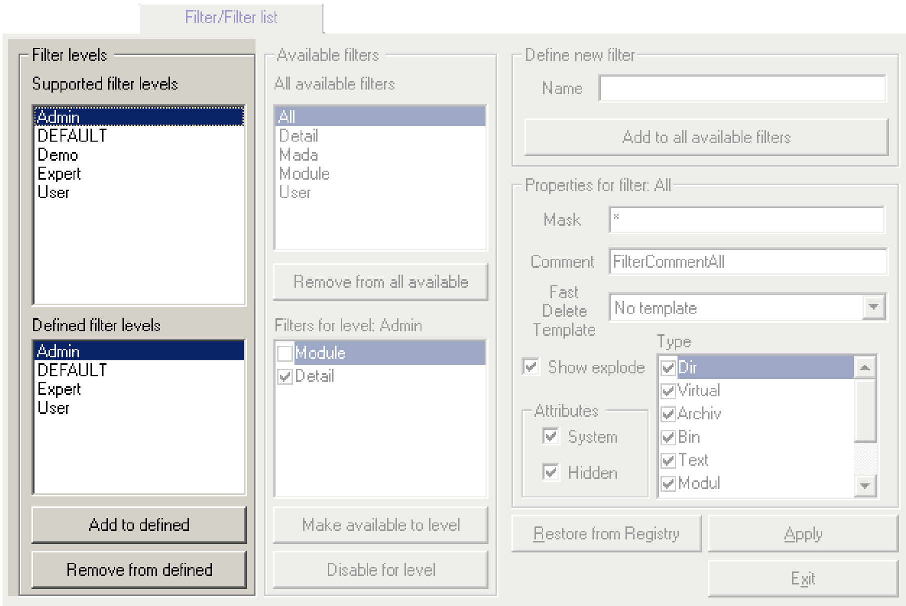


Fig. 6-22: Filter tab, Filter levels

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.

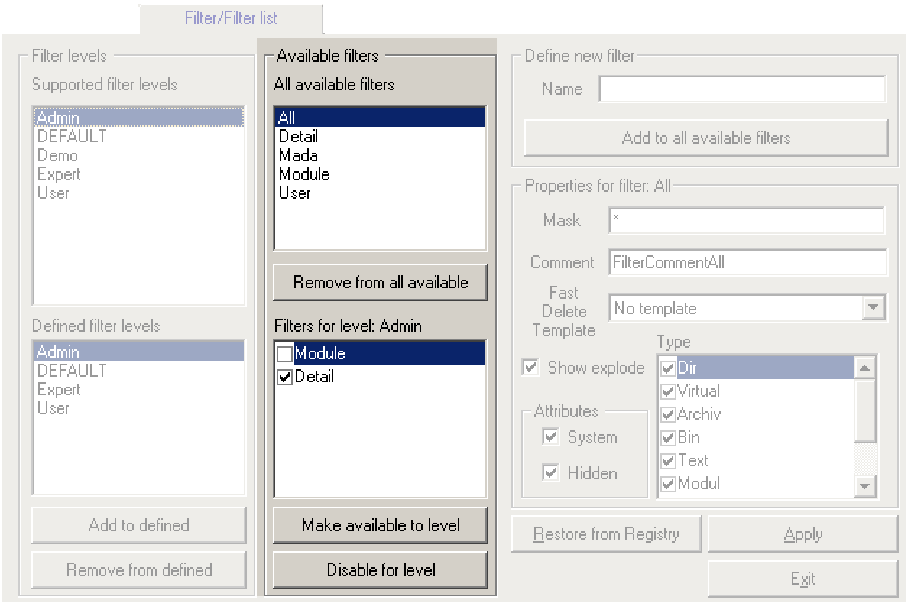


Fig. 6-23: Filter tab, Available filters

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.

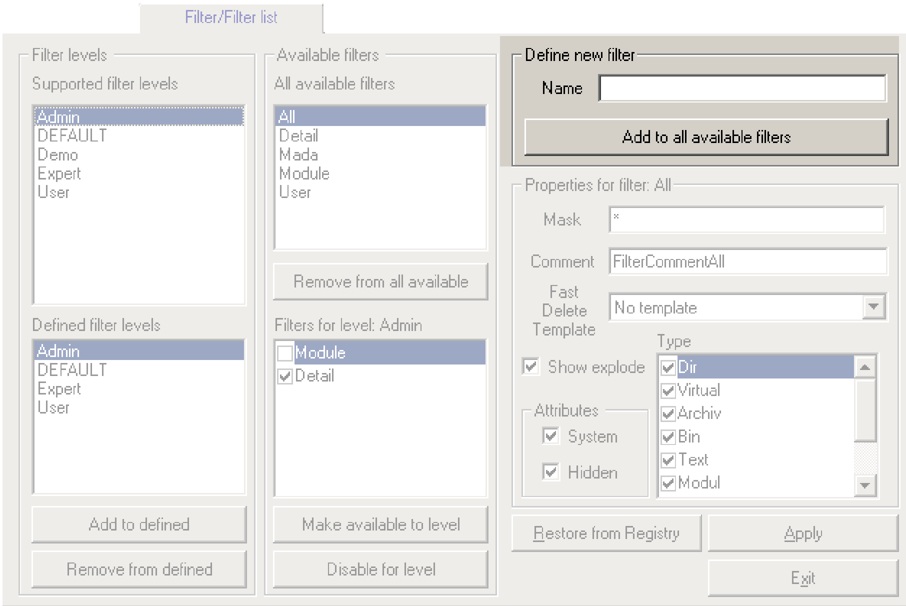


Fig. 6-24: Filter tab, Define new filter

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.

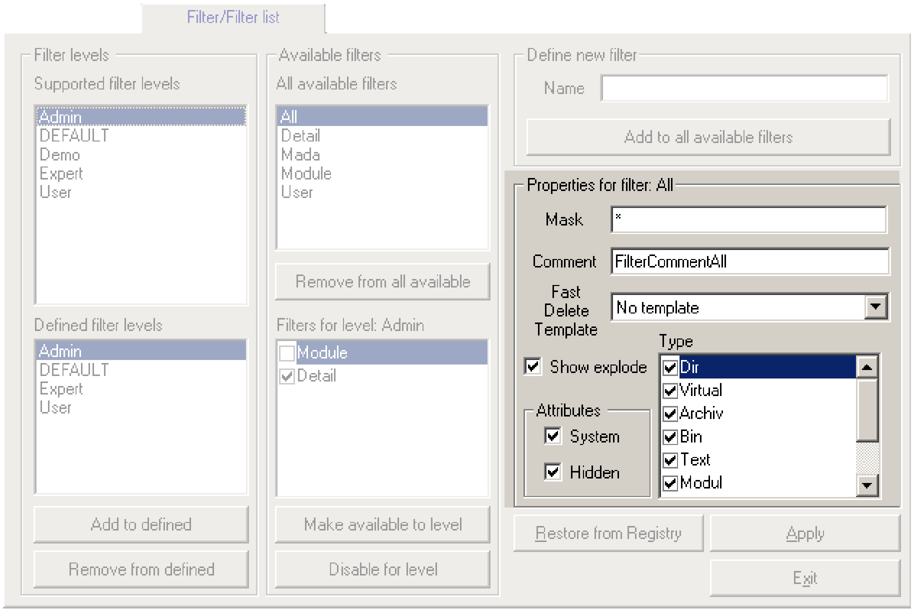


Fig. 6-25: Filter tab, Properties for filter

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

6.14.4 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.

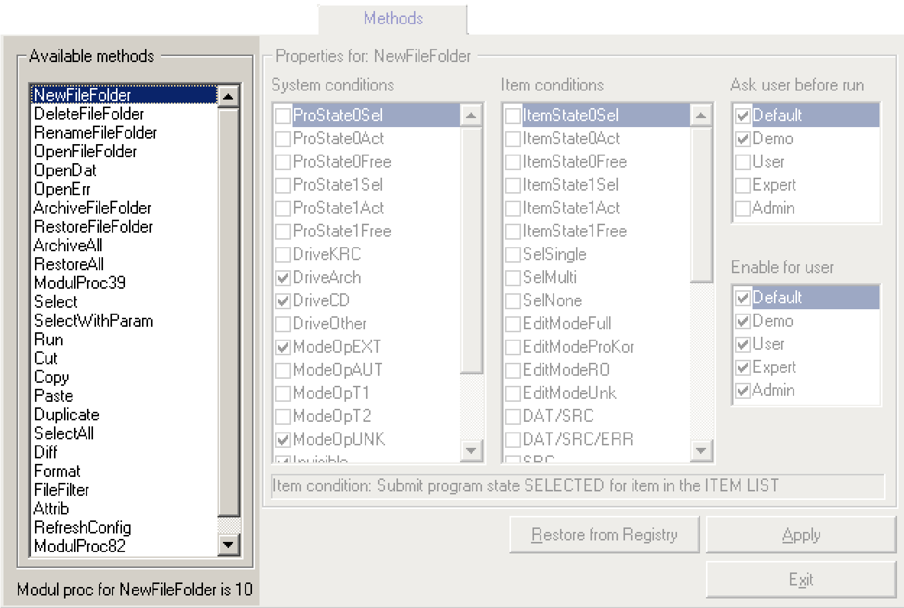


Fig. 6-26: Methods tab, Available methods

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.

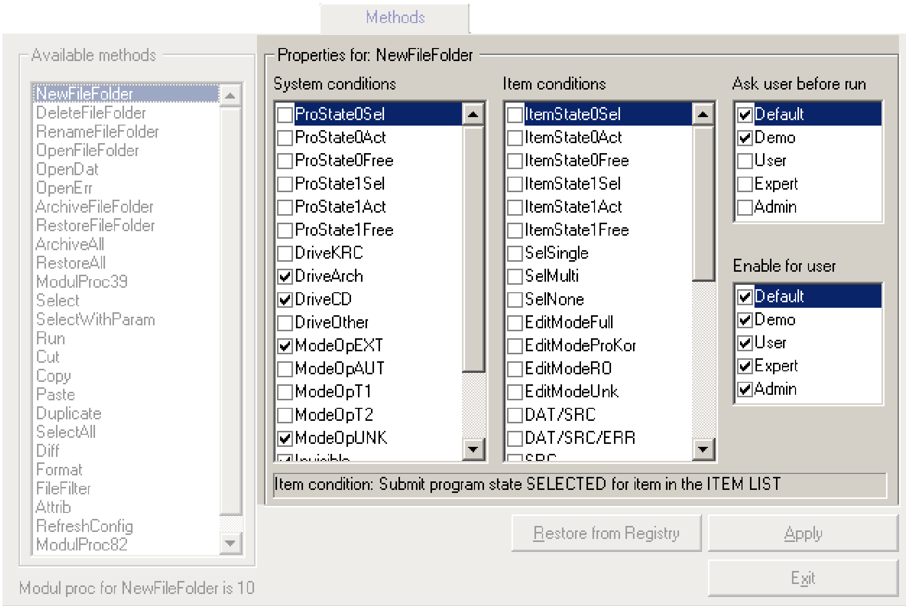


Fig. 6-27: Methods tab, Properties

Element	Description
Properties for:	The method/Navigator command selected in **Available methods** is displayed.
Ask user before run	Check box active: Before the command is executed in this user group, a request for confirmation is displayed.
Enable for user	Check box active: This command is permissible for this user group.

The settings in the KRC Configurator are not checked for validity or plausibility.
It is possible, for example, to activate the object states **SelSingle**, **SelMulti** and **SelNone** for a Navigator command. In this case the Navigator command would be inactive, as at least one of these object states is active at any given time.

System conditions

Check box active: The Navigator command is **not** available if this system state is active.

Example:

- Available method: **NewFileFolder**
- System condition: check box **DriveArch** active

  This means: no directories or files can be created in the Navigator for the ARCHIVE drive.

System conditions	Description
ProState0Sel	Submit program with “SELECTED” state
ProState0Act	Submit program with “ACTIVE” state
ProState0Free	Submit program with “FREE” state
ProState1Sel	Robot program with “SELECTED” state
ProState1Act	Robot program with “ACTIVE” state
ProState1Free	Robot program with “FREE” state
DriveKRC	“KRC:\” drive
DriveArch	“ARCHIVE:\” drive
DriveCD	CD-ROM drive
DriveOther	Hard drive
ModeOpEXT	“Automatic External” mode
ModeOpAUT	“Automatic” mode
ModeOpT1	“T1” mode
ModeOpT2	“T2” mode
ModeOpUNK	“Unknown” mode
Disable	The Navigator is disabled as long as the selected command is active
SelTreeWindow	Focus in the directory structure
SelListWindow	Focus in the file list

Item conditions

Check box active: The Navigator command is **not** available if this object state is active.

Item conditions	Description
ItemState0Sel	Submit program in the file list with “SELECTED” state
ItemState0Act	Submit program in the file list with “ACTIVE” state
ItemState0Free	Submit program in the file list with “FREE” state
ItemState1Sel	Robot program in the file list with “SELECTED” state
ItemState1Act	Robot program in the file list with “ACTIVE” state
ItemState1Free	Robot program in the file list with “FREE” state
SelSingle	Only one object selected
SelMulti	Multiple objects selected
SelNone	No objects selected
EditModeFull	Edit mode “FULL”
EditModeProKor	Edit mode “PROKOR”
EditModeRO	Edit mode “READONLY”
EditModeUNK	Edit mode “UNKNOWN”
DAT / SRC	Object is a DAT or SRC file
DAT / SRC / ERR	Object is a DAT or SRC file containing errors
SRC	Object is an SRC file

Item conditions	Description
SRC / ERR	Object is an SRC file containing errors
DAT / SUB	Object is a DAT or SUB file
DAT / SUB / ERR	Object is a DAT or SUB file containing errors
SUB	Object is a SUB file
SUB / ERR	Object is a SUB file containing errors
DAT / ERR	Object is a DAT file containing errors
DAT	Object is a DAT file
TXT	Object is a text file
BIN	Object is a binary file
Arch	Object is an archive
Virt	Object is a virtual directory
Folder	Object is a directory
IBGN File	Object is an IBGN file
Protected File	Object is an encrypted and/or signed file

6.14.5 User Methods tab

Overview

The layout and function of this tab are largely identical to those of the **Methods** tab (>>> 6.14.4 "Methods tab" page 148).

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.

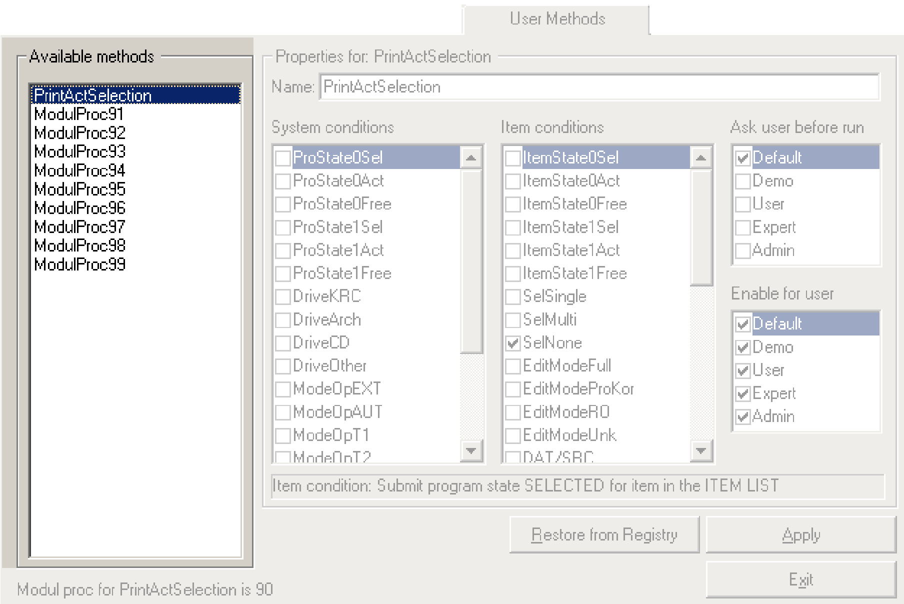


Fig. 6-28: User Methods tab, Available methods

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

Available methods	Description
ModulProc93	Internal identifier
ModulProc94	Internal identifier
ModulProc95	Internal identifier
ModulProc96	Internal identifier
ModulProc97	Internal identifier
ModulProc98	Internal identifier
ModulProc99	Internal identifier

Description: Properties

The properties for the Navigator command are defined here. A name can be assigned to the command.

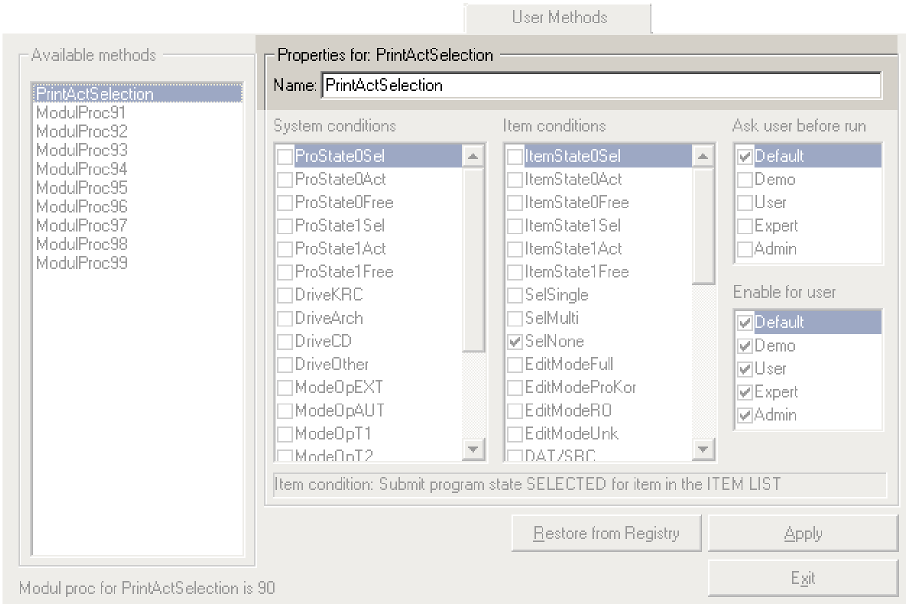


Fig. 6-29: User Methods tab, Properties

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

6.14.6 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.

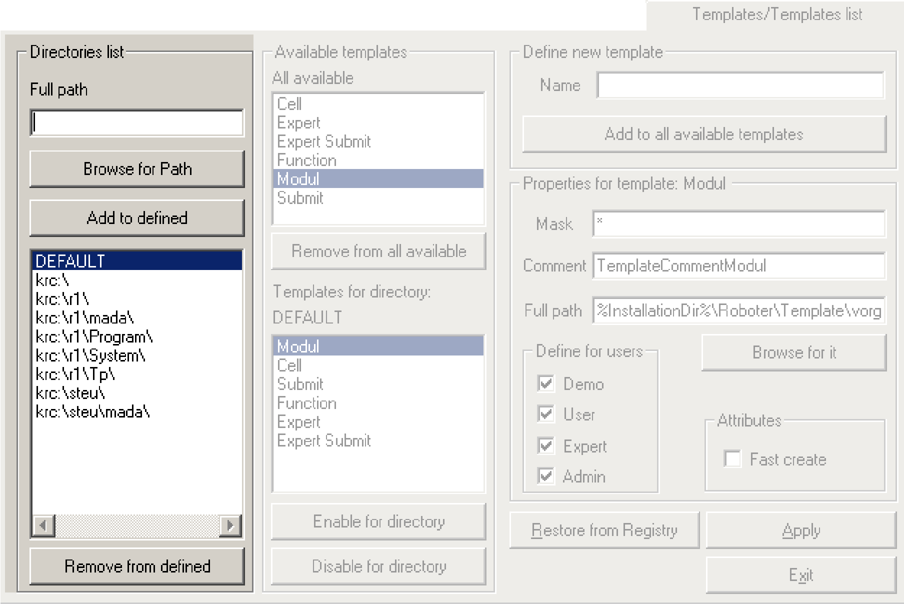


Fig. 6-30: Templates tab, Directories list

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.

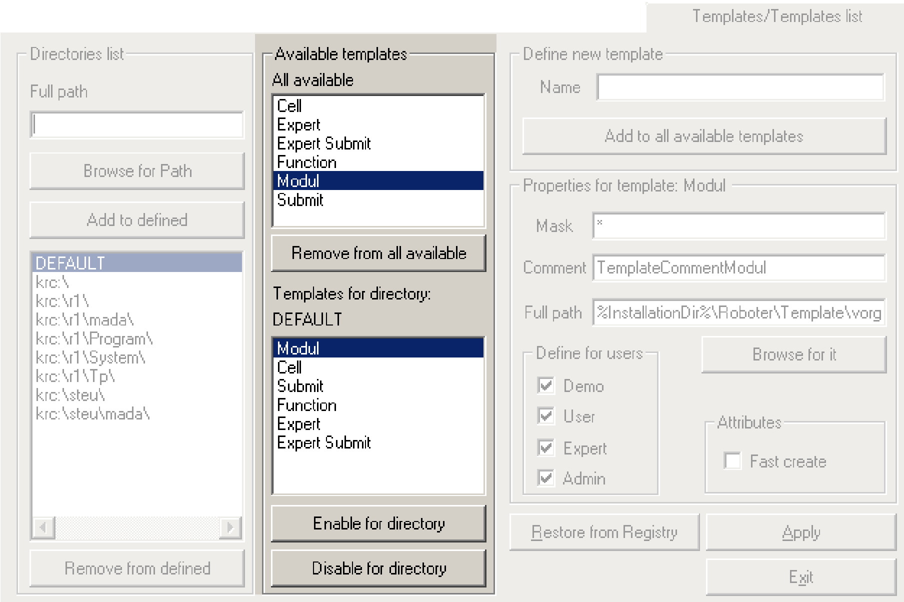


Fig. 6-31: Templates tab, Available templates

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 in the Navigator.
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.

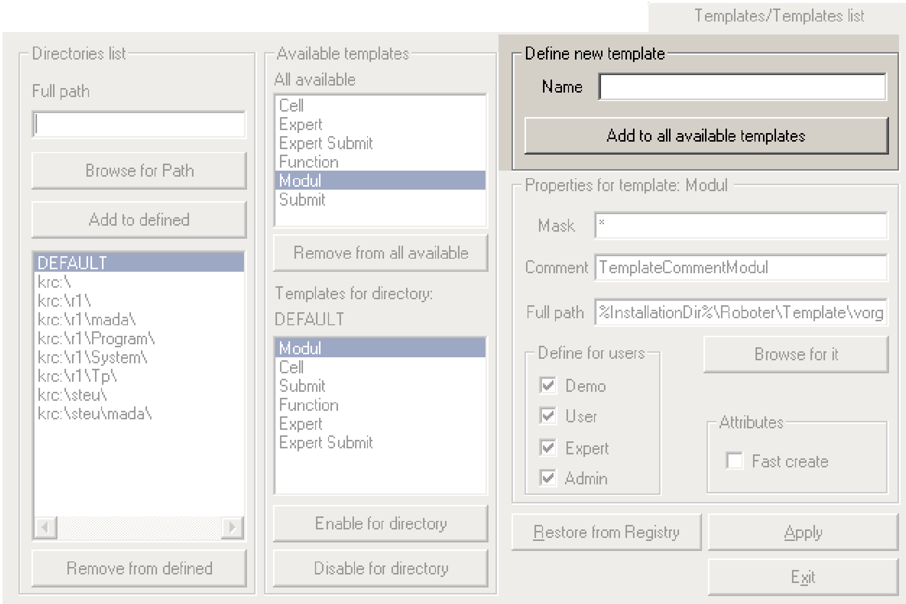


Fig. 6-32: Templates tab, Define new template

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.

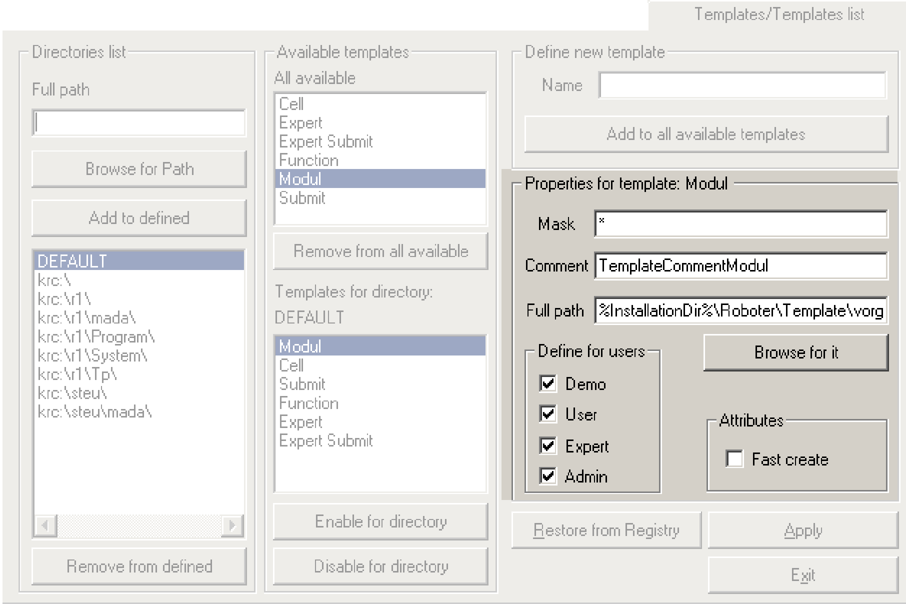


Fig. 6-33: Templates tab, Properties for template

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.

6.14.7 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

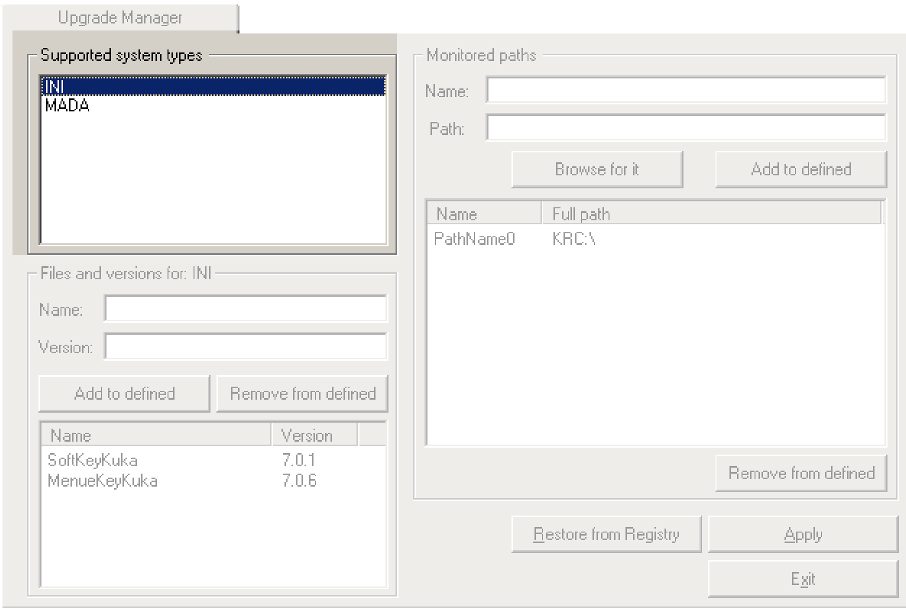


Fig. 6-34: Upgrade Manager tab, Supported system types

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.

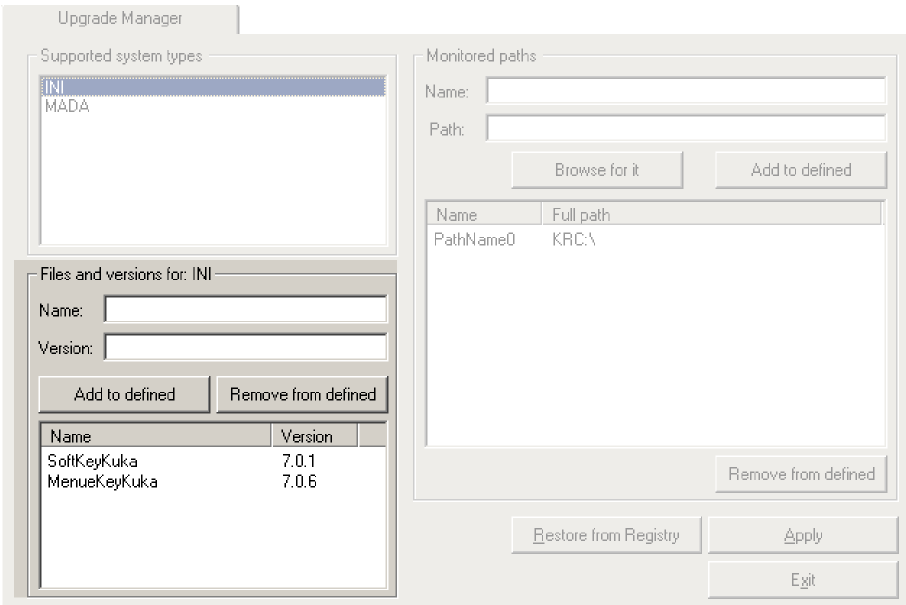


Fig. 6-35: Upgrade Manager tab, Files and versions

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)
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.

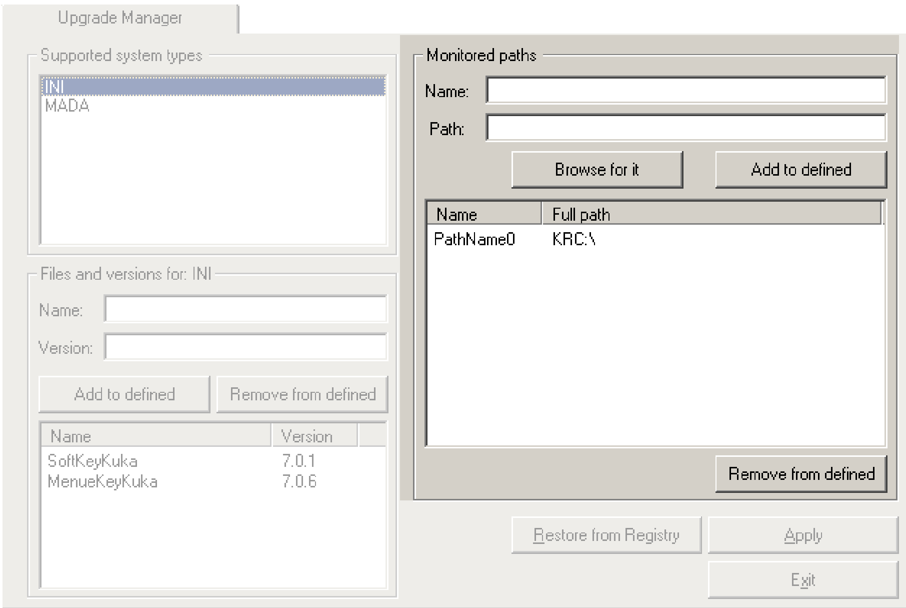


Fig. 6-36: Upgrade Manager tab, Monitored paths

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

6.14.8 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 (>>> 6.14.9 "History Info tab" page 160).

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.

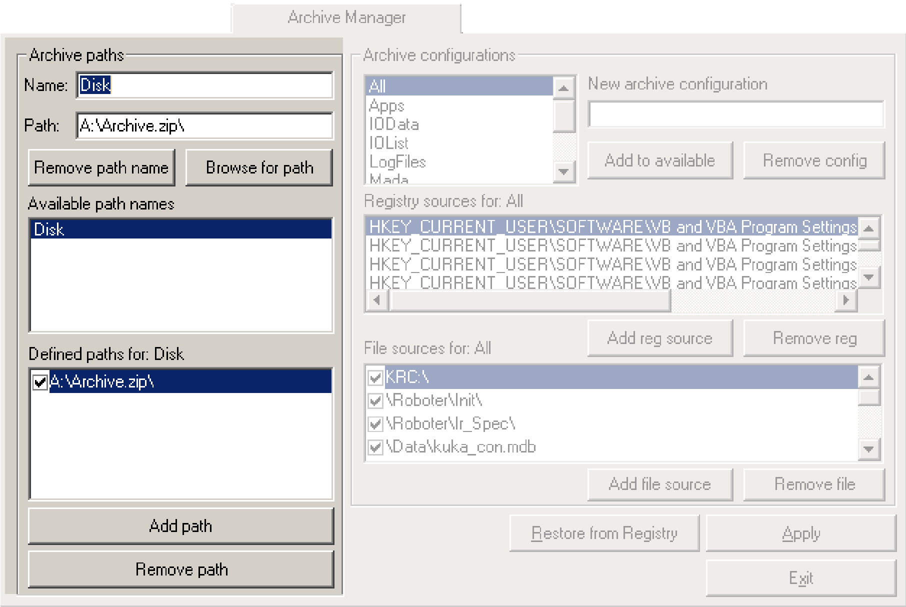


Fig. 6-37: Archive Manager tab, Archive paths

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.

Element	Description
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**.

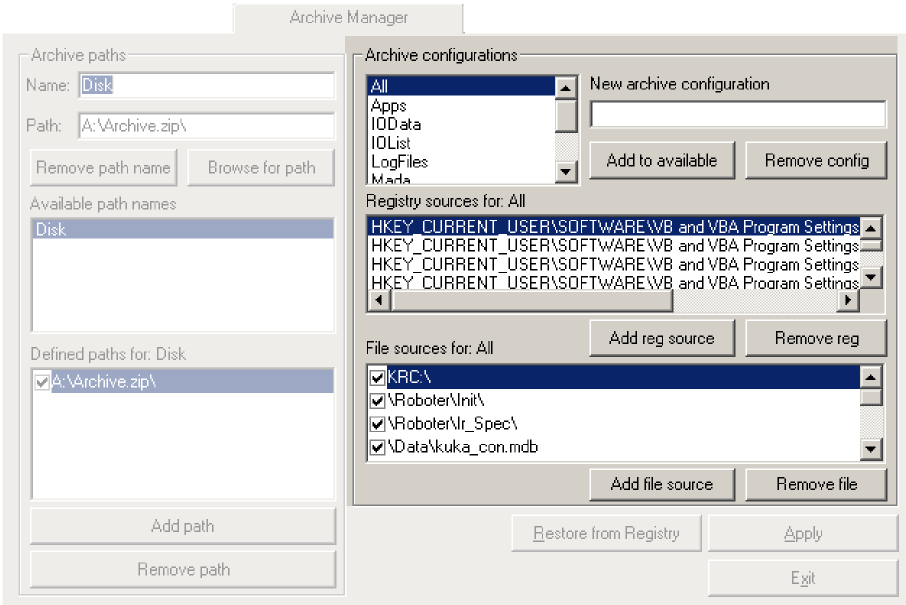


Fig. 6-38: Archive Manager tab, Archive configurations

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.

Element	Description
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
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**.

6.14.9 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

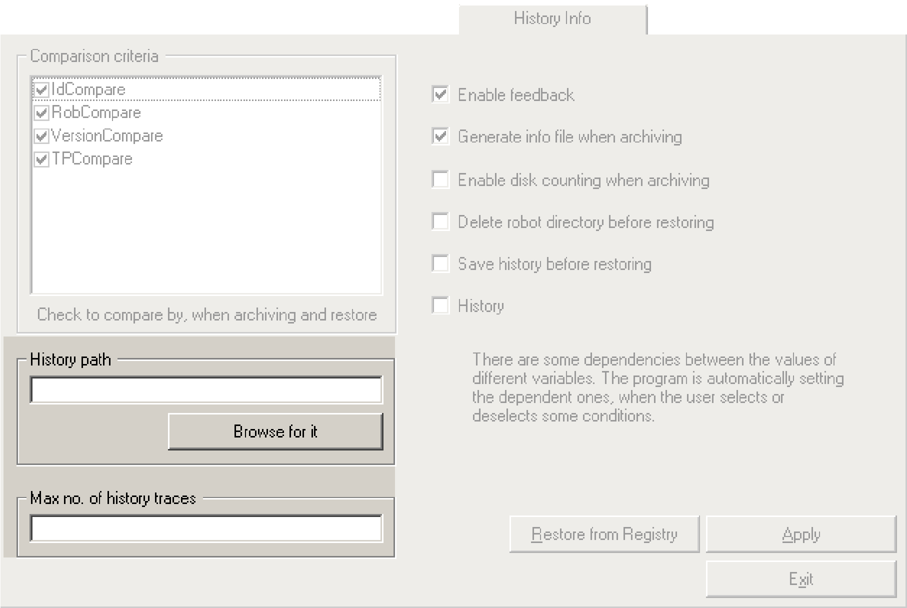


Fig. 6-39: History Info tab, History

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

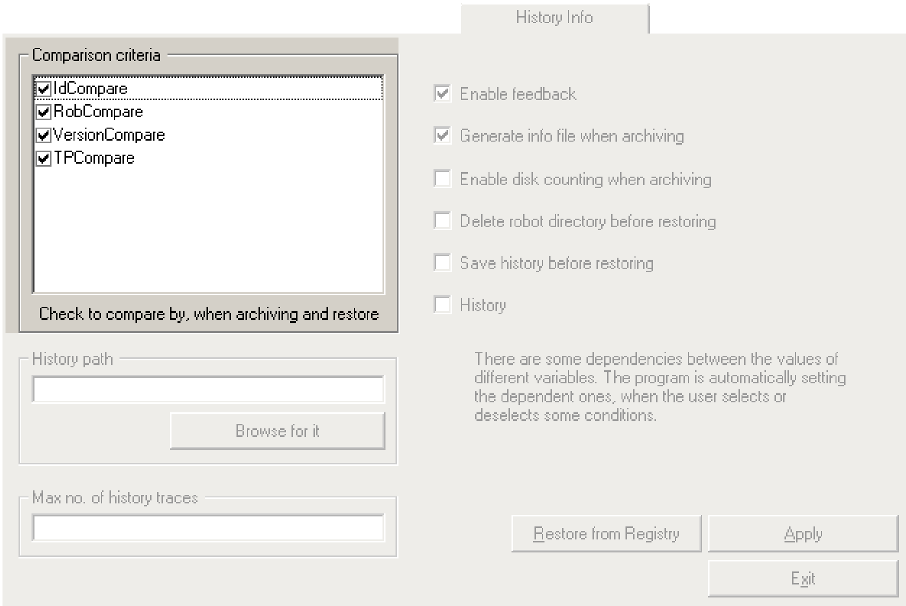


Fig. 6-40: History Info tab, Comparison criteria

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

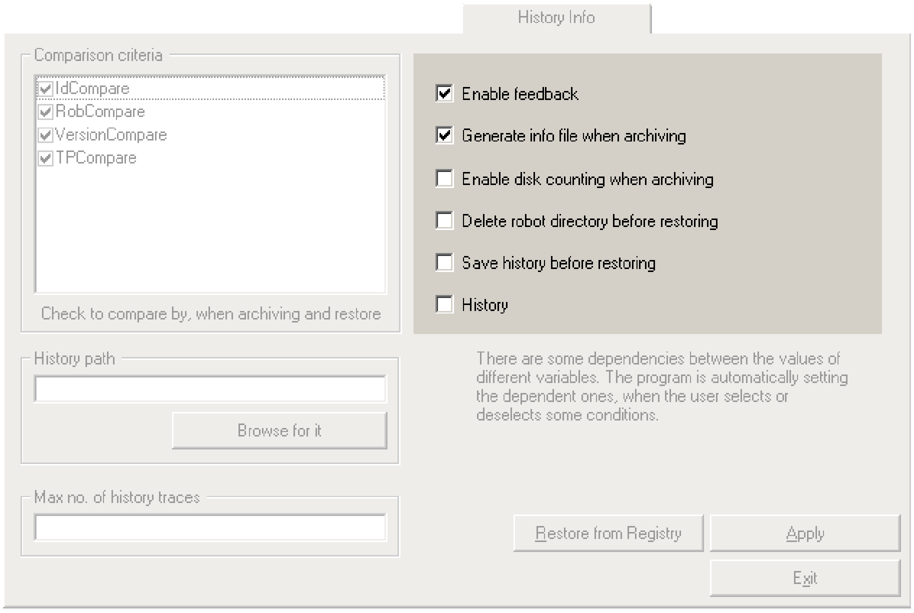


Fig. 6-41: History Info tab, further settings

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.

6.14.10 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

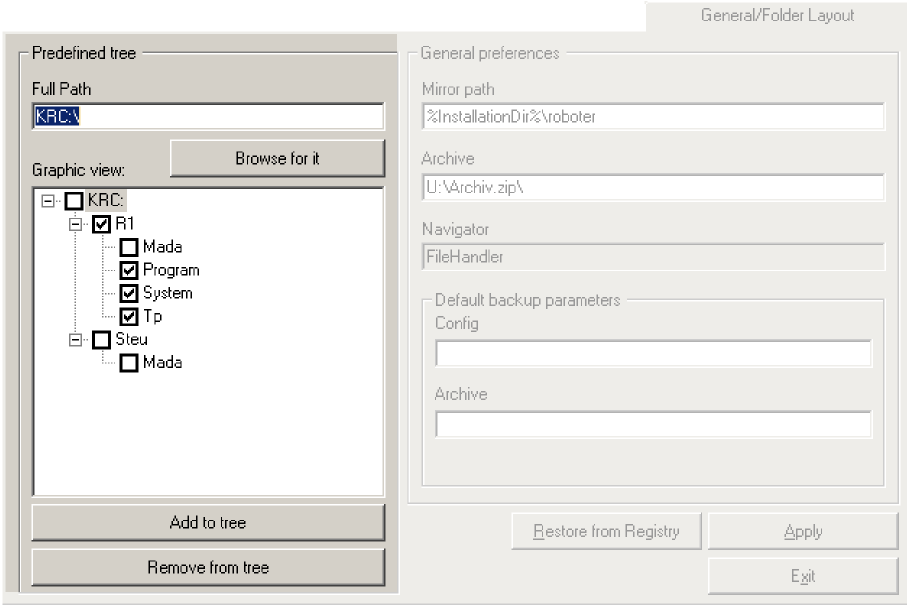


Fig. 6-42: General/Folder Layout tab, Predefined tree

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 %INSTALLATIONDIR%\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.

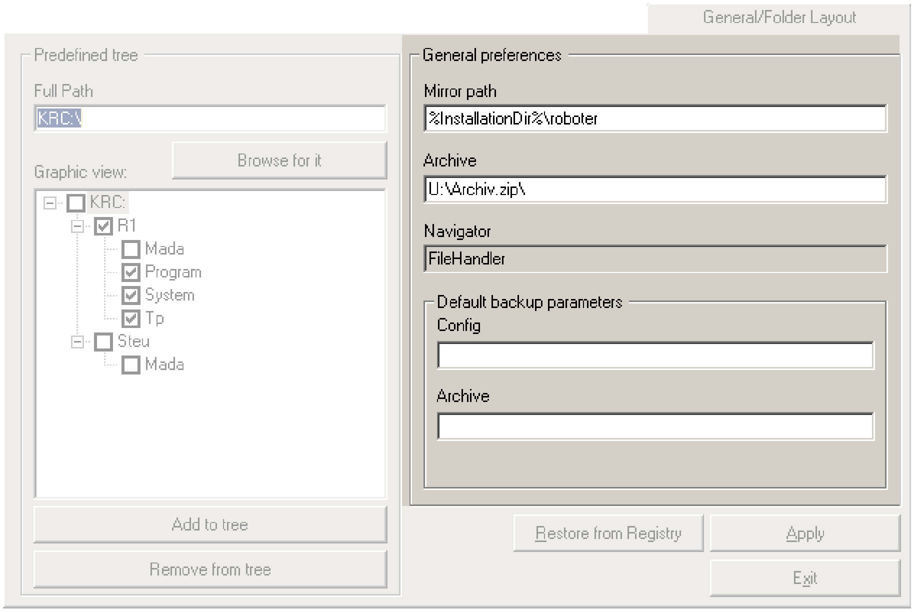


Fig. 6-43: General/Folder Layout tab, General preferences

7 Programming for user group “User” (inline forms)

7.1 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
```

Program 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. The INI line must not be deleted!
4	HOME position (>>> 7.2 "HOME position" page 165)
8	LIN motion (>>> 7.4.4 "Programming a LIN motion" page 172)
14	PTP motion (>>> 7.4.2 "Programming a PTP motion" page 171)
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 in a KRL program must be a PTP motion with Cartesian coordinates. This does not apply to subprograms.

7.2 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 in the robot controller. Up to 9 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 a HOME position is modified, this affects all programs in which it is used. Physical injuries or damage to property may result.

7.3 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.

7.4 Programming motions (with inline forms)

Inline forms are available in the KSS for frequently used instructions.

Motions can also be programmed without inline forms. Information is contained in the description of the KRL syntax. (>>> 8.5 "Motion programming" page 201)

7.4.1 Basic principles of motion programming

7.4.1.1 Motion types

Overview

The following motion types can be programmed:

- Point-to-point motions (PTP)
  (>>> 7.4.2 "Programming a PTP motion" page 171)
- Linear motions (LIN)
  (>>> 7.4.4 "Programming a LIN motion" page 172)
- Circular motions (CIRC)
  (>>> 7.4.6 "Programming a CIRC motion" page 174)

LIN and CIRC motions are also known as CP (“Continuous Path”) motions.

The start point of a motion is always the end point of the previous motion or, if the robot is stationary, the current robot position.

PTP motion

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.

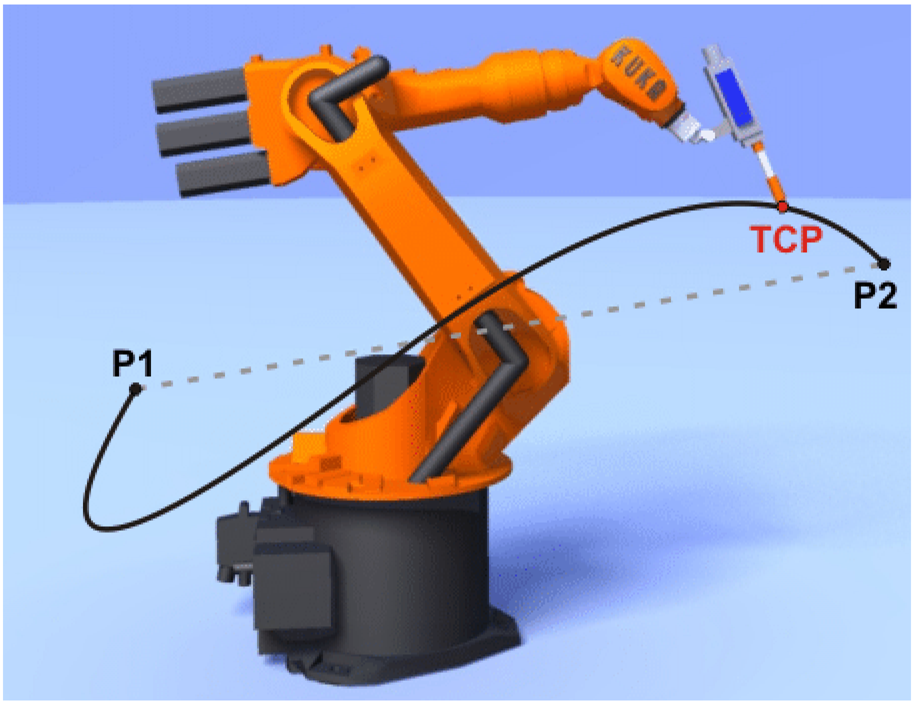


Fig. 7-1: PTP motion

LIN motion

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

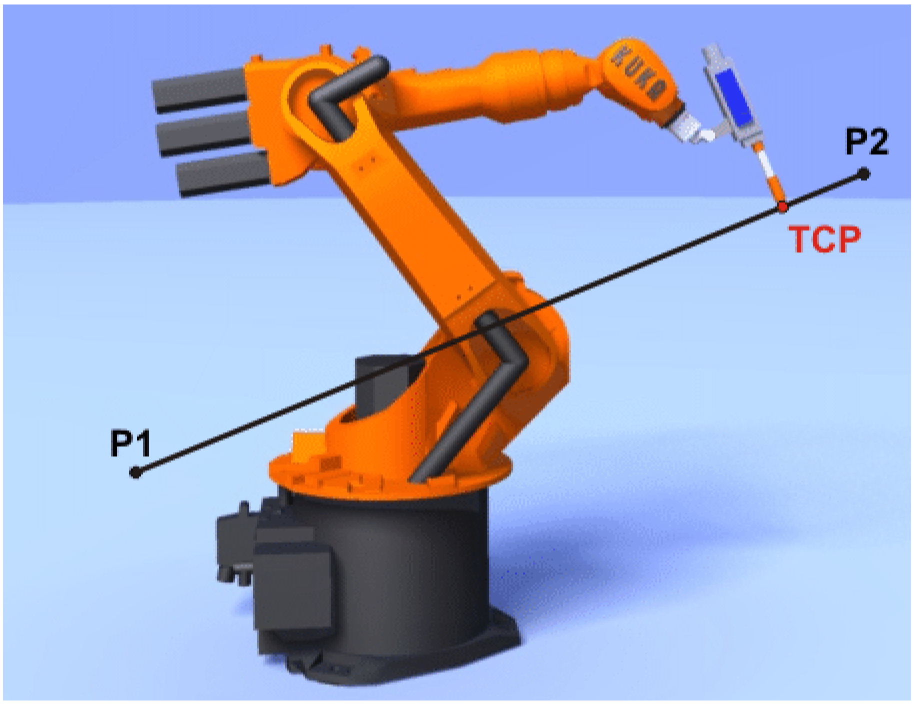


Fig. 7-2: LIN motion

CIRC motion

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.

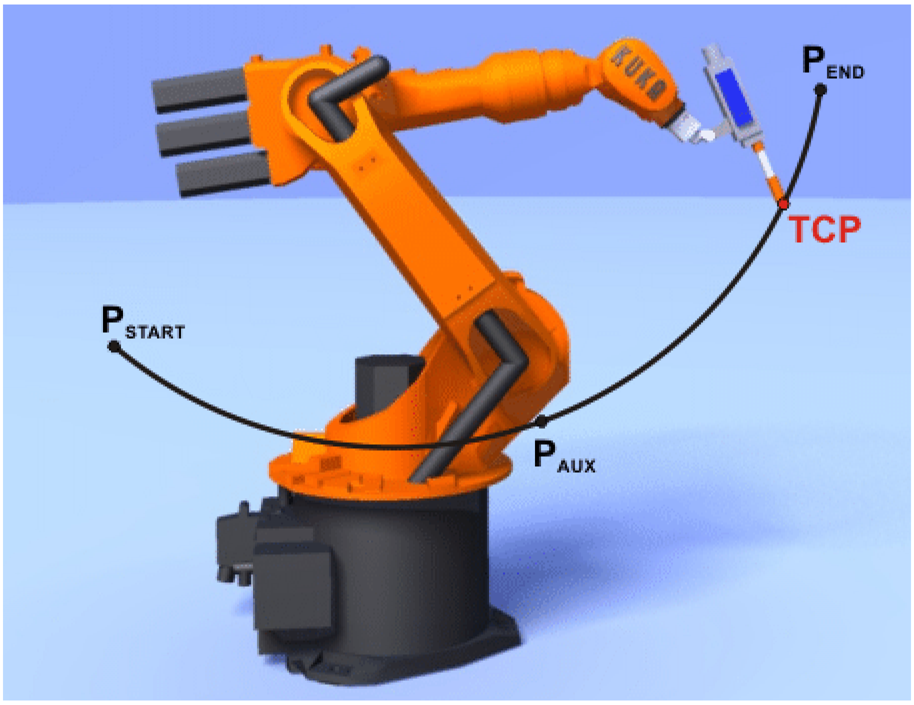


Fig. 7-3: CIRC motion

7.4.1.2 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.

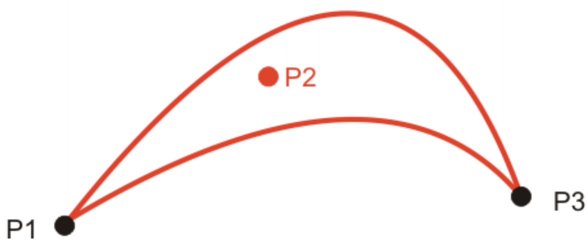


Fig. 7-4: PTP motion, P2 is approximated

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.

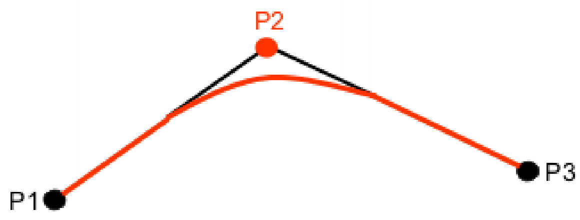


Fig. 7-5: LIN motion, P2 is approximated

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.

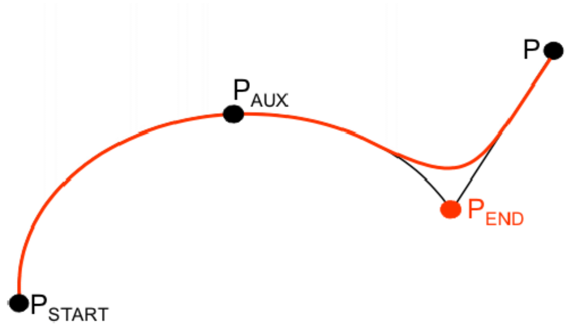


Fig. 7-6: CIRC motion, P_{END} is approximated

7.4.1.3 Orientation control

Overview

The orientation of a tool 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 following options are available:

- Standard
- Wrist PTP
- Orientation control - Constant

For CIRC motions: The orientation at the auxiliary point is not taken into consideration.

Description

Standard

The orientation of the tool changes continuously during the motion.

The orientation is achieved by rotating and pivoting about the TCP.

Wrist PTP

The orientation of the tool changes continuously during the motion.

The CP motion is broken down into several small PTP motions by the robot controller. This excludes the possibility of a singularity occurring in the case of **Wrist PTP**. The robot can deviate slightly from its path, however. **Wrist PTP** is thus not suitable if the robot must follow its path exactly, e.g. in the case of laser welding.

- If, with **Standard**, the robot passes through a singularity, select **Wrist PTP** instead.
- If, with **Standard** selected, a singularity arises, but the path must be followed exactly, reteach the start and/or end point. Select orientations that do not result in a singularity.

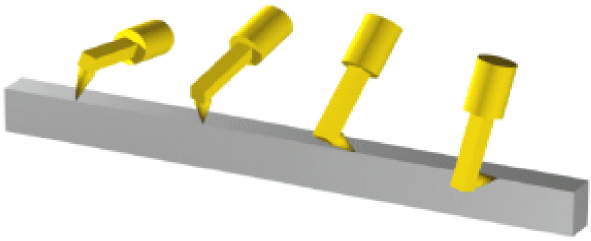


Fig. 7-7: Standard or Wrist PTP

Orientation control - Constant

The orientation of the tool remains constant during the motion. The programmed orientation is disregarded for the end point and that of the start point is retained.

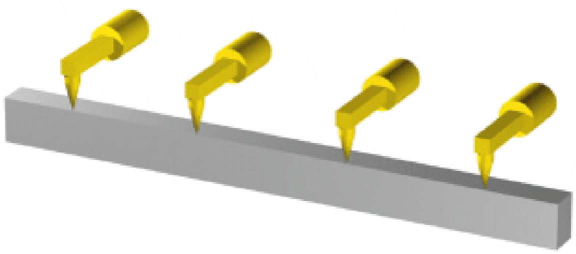


Fig. 7-8: Orientation control - Constant

7.4.1.4 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 (recommended 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 (recommended 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 (recommended 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.

7.4.2 Programming a PTP motion

Description

Programming a PTP motion involves the following steps:

- Saving the coordinates of the end point.
- Setting various parameters (e.g. velocity).

The process of saving the point coordinates is called "teaching".

The start point of a motion is always the end point of the previous motion or, if the robot is stationary, the current robot position.

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.

   (>>> 7.4.3 "Inline form for PTP motions" page 172)
5. Save the instruction by pressing the **Cmd Ok** softkey.

7.4.3 Inline form for PTP motions

Fig. 7-9: Inline form for PTP motions

Item	Description	Range of values
1	Type of motion	PTP, LIN, CIRC
2	name of the end point The system automatically generates a name. The name can be overwritten. Position the cursor in this box to edit the point data. The corresponding option window is opened. (>>> 7.4.8 "Option window "Frames"" page 175)	(>>> 7.3 "Names in inline forms" page 166)
3	■ CONT: end point is approximated ■ [blank]: the motion stops exactly at the end point	CONT, [blank]
4	Velocity	0% … 100%
5	Name for the motion data set The system automatically generates a name. The name can be overwritten. Position the cursor in this box to edit the motion data. The corresponding option window is opened. (>>> 7.4.9 "Option window "Motion parameter" (PTP motion)" page 176)	(>>> 7.3 "Names in inline forms" page 166)

7.4.4 Programming a LIN motion

Description

Programming a LIN motion involves the following steps:

- Saving the coordinates of the end point.
- Setting various parameters (e.g. velocity).

The process of saving the point coordinates is called "teaching".

The start point of a motion is always the end point of the previous motion or, if the robot is stationary, the current robot position.

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.
   (>>> 7.4.5 "Inline form for LIN motions" page 173)
5. Save the instruction by pressing the **Cmd Ok** softkey.

7.4.5 Inline form for LIN motions

Fig. 7-10: Inline form for LIN motions

Item	Description	Range of values
1	Type of motion	PTP, LIN, CIRC
2	name of the end point The system automatically generates a name. The name can be overwritten. Position the cursor in this box to edit the point data. The corresponding option window is opened. (>>> 7.4.8 "Option window "Frames"" page 175)	(>>> 7.3 "Names in inline forms" page 166)
3	■ CONT: end point is approximated ■ [blank]: the motion stops exactly at the end point	CONT, [blank]
4	Velocity	0.001 … 2 m/s
5	Name for the motion data set The system automatically generates a name. The name can be overwritten. Position the cursor in this box to edit the motion data. The corresponding option window is opened. (>>> 7.4.10 "Option window "Motion parameter" (CP motion)" page 176)	(>>> 7.3 "Names in inline forms" page 166)

7.4.6 Programming a CIRC motion

Description

Programming a CIRC motion involves the following steps:

- Saving the coordinates of the auxiliary point.
- Saving the coordinates of the end point.
- Setting various parameters (e.g. velocity).

The process of saving the point coordinates is called "teaching".

The start point of a motion is always the end point of the previous motion or, if the robot is stationary, the current robot position.

Precondition

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

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

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.

   (>>> 7.4.7 "Inline form for CIRC motions" page 174)
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.

7.4.7 Inline form for CIRC motions

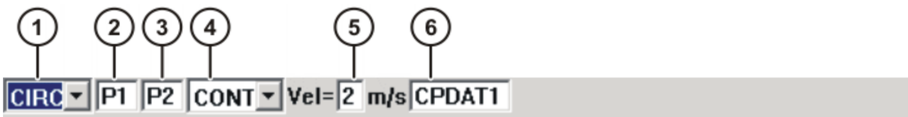


Fig. 7-11: Inline form for CIRC motions

Item	Description	Range of values
1	Type of motion	PTP, LIN, CIRC
2	Name of the auxiliary point The system automatically generates a name. The name can be overwritten.	(>>> 7.3 "Names in inline forms" page 166)
3	name of the end point The system automatically generates a name. The name can be overwritten. Position the cursor in this box to edit the point data. The corresponding option window is opened. (>>> 7.4.8 "Option window "Frames"" page 175)	(>>> 7.3 "Names in inline forms" page 166)
4	■ CONT: end point is approximated ■ [blank]: the motion stops exactly at the end point	CONT, [blank]

Item	Description	Range of values
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. Position the cursor in this box to edit the motion data. The corresponding option window is opened. (>>> 7.4.10 "Option window "Motion parameter" (CP motion)" page 176)	(>>> 7.3 "Names in inline forms" page 166)

7.4.8 Option window "Frames"

This option window is called from the following inline forms:

- PTP (>>> 7.4.3 "Inline form for PTP motions" page 172)
- LIN (>>> 7.4.5 "Inline form for LIN motions" page 173)
- CIRC (>>> 7.4.7 "Inline form for CIRC motions" page 174)

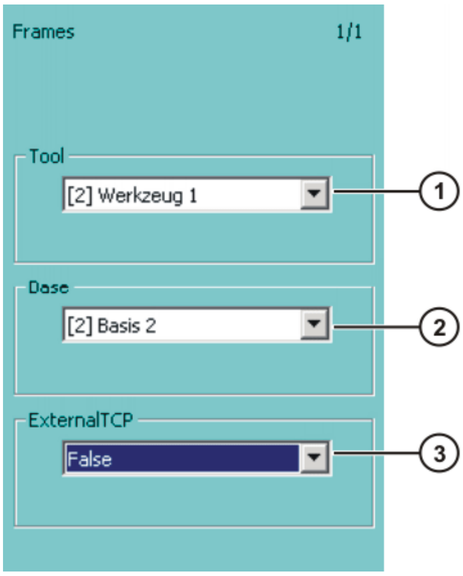


Fig. 7-12: Option window "Frames"

Item	Description	Range of values
1	Tool selection. If **True** in the box **External TCP**: workpiece selection.	[1] … [16]
2	Base selection. If **True** in the box **External TCP**: fixed tool selection.	[1] … [32]
3	■ Tool on mounting flange: **False** ■ Fixed tool: **True**	False, True

7.4.9 Option window "Motion parameter" (PTP motion)

This option window is called from the following inline form:

- PTP (>>> 7.4.3 "Inline form for PTP motions" page 172)

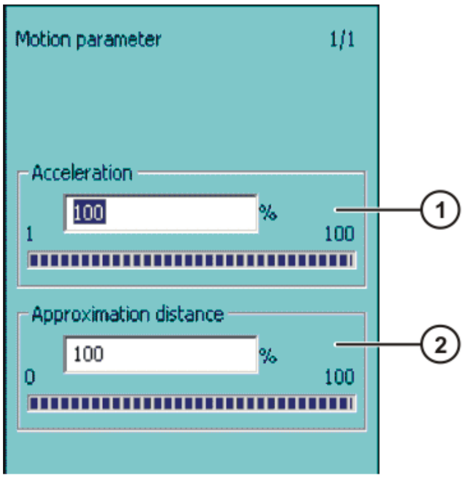


Fig. 7-13: Option window "Motion parameter" (PTP motion)

Item	Description	Range of values
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	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 This box is only displayed if **CONT** has been selected in the inline form.	0% … 100%

7.4.10 Option window "Motion parameter" (CP motion)

This option window is called from the following inline forms:

- LIN (>>> 7.4.5 "Inline form for LIN motions" page 173)
- CIRC (>>> 7.4.7 "Inline form for CIRC motions" page 174)

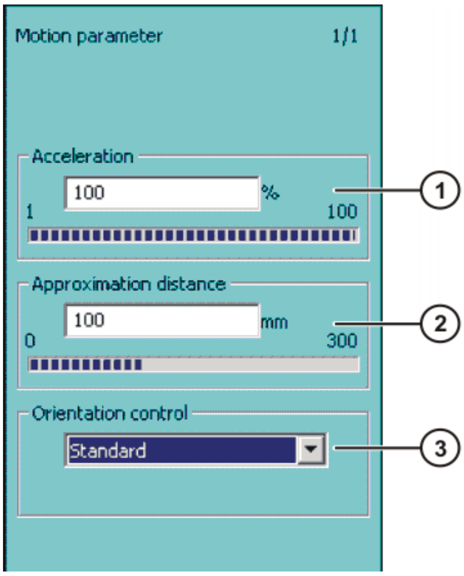


Fig. 7-14: Option window "Motion parameter" (CP motion)

Item	Description	Range of values
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	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** has been selected in the inline form.	0 mm … 300 mm
3	Orientation control selection (>>> 7.4.1.3 "Orientation control" page 169)	■ Standard ■ Wrist PTP ■ Orientation control - Constant

7.4.11 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.

7.4.12 Modifying 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 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.

7.5 Torque monitoring

Description

If the robot collided with a workpiece, the torques of the axes involved are increased in order to overcome the obstacle. This can result in damage to the robot or the workpiece.

To limit the risk of such damage, it is possible to define the maximum extent to which the actual torque may exceed or fall below the command torque. If this tolerance range is exceeded, the robot stops with a STOP 1.

The tolerance range must be defined such that the torque deviations arising during normal robot motion are within the tolerance range. This torque deviation can be caused, for example, by the difference between motion with a load and motion without a load.

The values for the tolerance range (per axis) are stored in the following system variables in the file C:\KRC\Roboter\KRC\MaDa\$CUSTOM.DAT:

- Program mode

  $TORQMON_DEF[1] … $TORQMON_DEF[6]
- Jog mode

  $TORQMON_COM_DEF[1] … $TORQMON_COM_DEF[6]

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

It is also possible to define a response time for the torque monitoring.

External axes are not monitored.

Signal declarations in the file KRC:\STEU\Mada\$machine.dat:

- $COLL_ENABLE

  This output is set if the value of one of the $TORQMON_DEF[…] variables is less than 200.

- $COLL_ALARM

  This output is set if message 117 “Collision Detection axis <axis number>” is generated. The output remains set as long as $STOPMESS is active.

```
SIGNAL $COLL_ALARM $OUT[151] ;KOLLISIONSALARM
SIGNAL $COLL_ENABLE $OUT[152] ;KOLLISIONSSOFTWARE EIN/AUS
```

Overview

Step	Description
1	Determine suitable values for torque monitoring. (>>> 7.5.1 "Determining values for torque monitoring" page 179)
2	Program torque monitoring. (>>> 7.5.2 "Programming torque monitoring" page 179)

7.5.1 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[...].

7.5.2 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\Robot-er\KRC\R1\MaDa\$ROBCOR.DAT.
- 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.

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

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

7.6 Programming logic instructions

7.6.1 Inputs/outputs

Digital inputs/outputs

The robot controller can manage up to 4096 digital inputs and 4096 digital outputs. The inputs/outputs are implemented in the control PC by means of optional field bus cards. The configuration is customer-specific.

Analog inputs/outputs

The robot controller can manage 32 analog inputs and 32 analog outputs. The inputs/outputs are implemented in the control PC by means of KUKA field bus cards. The configuration is customer-specific.

Permissible range of values for inputs/outputs: -1.0 to +1.0. This corresponds to a voltage range from -10 V to +10 V. If the value is exceeded, the input/output takes the maximum value and a message is displayed until the value is back in the permissible range.

The inputs/outputs are managed via the following system variables:

	Inputs	Outputs
Digital	$IN[1] … $IN[4096]	$OUT[1] … $OUT[4096]
Analog	$ANIN[1] … $ANIN[32]	$ANOUT[1] … $ANOUT[32]

7.6.2 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.

   (>>> 7.6.3 "Inline form "OUT"" page 180)
4. Save the instruction by pressing the **Cmd Ok** softkey.

7.6.3 Inline form "OUT"

The instruction sets a digital output.

Fig. 7-15: Inline form "OUT"

Item	Description	Range of values
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.	Freely selectable
3	State to which the output is switched	TRUE, FALSE
4	■ CONT: Execution in the advance run ■ [blank]: Execution with advance run stop	CONT, [blank]

7.6.4 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.
   (>>> 7.6.5 "Inline form "PULSE"" page 181)
4. Save the instruction by pressing the **Cmd Ok** softkey.

7.6.5 Inline form "PULSE"

The instruction sets a pulse of a defined length.

Fig. 7-16: Inline form "PULSE"

Item	Description	Range of values
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.	Freely selectable
3	State to which the output is switched ■ TRUE: "High" level ■ FALSE: "Low" level	TRUE, FALSE

Item	Description	Range of values
4	■ CONT: Execution in the advance run ■ [blank]: Execution with advance run stop	CONT, [blank]
5	Length of the pulse	0.1 … 3

7.6.6 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.
   (>>> 7.6.7 "Inline form "ANOUT" (static)" page 182)
   (>>> 7.6.8 "Inline form "ANOUT" (dynamic)" page 182)
4. Save the instruction by pressing the **Cmd Ok** softkey.

7.6.7 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. 7-17: Inline form "ANOUT" (static)

Item	Description	Range of values
1	Analog output number	CHANNEL_1 … CHANNEL_32
2	Factor for the voltage	0 … 1 Intervals: 0.01

7.6.8 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.

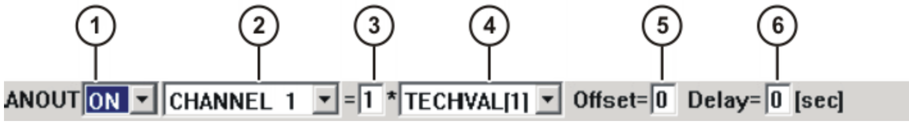


Fig. 7-18: Inline form "ANOUT" (dynamic)

Item	Description	Range of values
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[x]: The voltage is controlled by a function generator.	VEL_ACT, TECHVAL[1] … TECHVAL[6]
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

7.6.9 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.

   (>>> 7.6.10 "Inline form "WAIT"" page 183)
4. Save the instruction by pressing the **Cmd Ok** softkey.

7.6.10 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. 7-19: Inline form "WAIT"

Item	Description	Range of values
1	Wait time	≥ 0 s

7.6.11 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.

   (>>> 7.6.12 "Inline form "WAITFOR"" page 184)
4. Save the instruction by pressing the **Cmd Ok** softkey.

7.6.12 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.

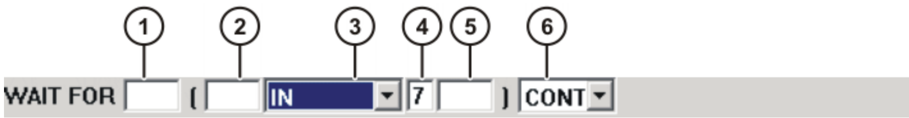


Fig. 7-20: Inline form "WAITFOR"

Item	Description	Range of values
1	■ Add external logic operation. The operator is situated between the bracketed expressions. ■ Add NOT. Enter the desired operator or NOT by means of the softkey.	AND, OR, EXOR, [blank] NOT, [blank]
2	■ Add internal logic operation. The operator is situated inside a bracketed expression. ■ Add NOT. Enter the desired operator or NOT by means of the softkey.	AND, OR, EXOR, [blank] NOT, [blank]
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.	Freely selectable
6	■ CONT: Execution in the advance run ■ [blank]: Execution with advance run stop	CONT, [blank]

7.6.13 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.
   (>>> 7.6.14 "Inline form SYN OUT, option START/END" page 185)
   (>>> 7.6.15 "Inline form SYN OUT, option PATH" page 187)
4. Save the instruction by pressing the **Cmd Ok** softkey.

7.6.14 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.

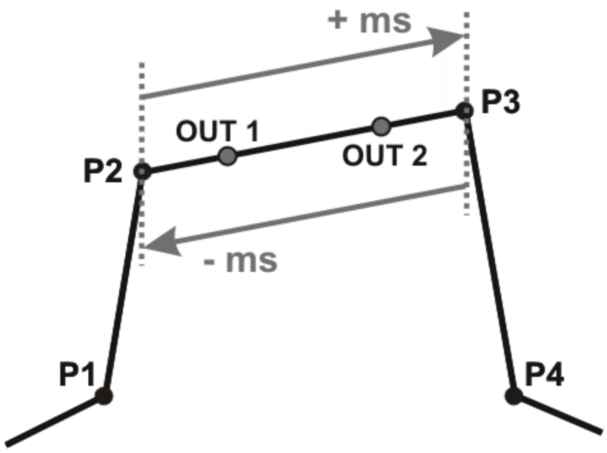
Fig. 7-21: Inline form SYN OUT, option START/END

Item	Description	Range of values
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.	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.	START, END Option PATH: (>>> 7.6.15 "Inline form SYN OUT, option PATH" page 187)
5	Switching action delay **Note:** The time specification is absolute. The switching point thus varies according to the velocity of the robot.	-1,000 … +1,000 ms

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
```



OUT 1 and OUT 2 specify approximate positions at which switching is to occur. The dotted lines indicate the switching limits.

Switching limits:

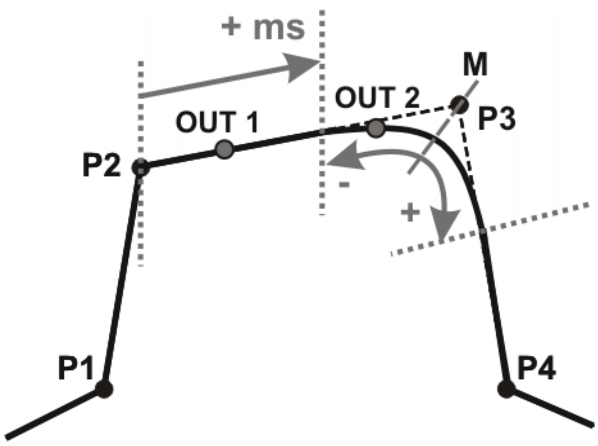
- START: The switching point can be delayed, at most, as far as exact positioning point P3 (+ ms).
- END: The switching point can be brought forward, at most, as far as exact positioning point P2 (- ms).

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
```



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 (-).

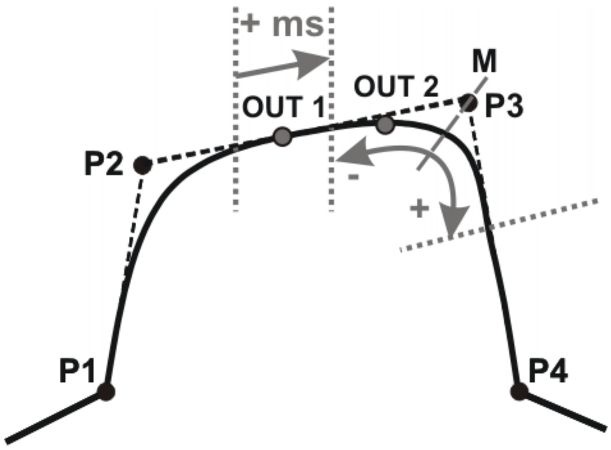
  The switching point can be delayed, at most, as far as the end of the approximate positioning range of P3 (+).

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
```



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.

7.6.15 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.

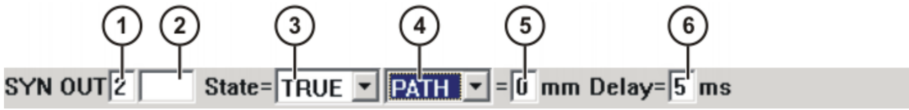


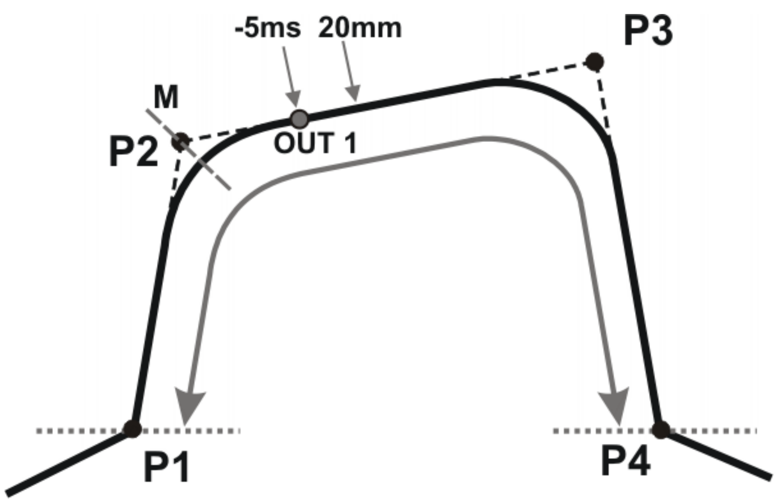
Fig. 7-22: Inline form SYN OUT, option PATH

Item	Description	Range of values
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.	Freely selectable
3	State to which the output is switched	TRUE, FALSE
4	- - -	PATH Option START or END: (>>> 7.6.14 "Inline form SYN OUT, option START/ END" page 185)
5	Distance from the switching point to the end point This box is only displayed if **PATH** has been selected under item 4.	-2,000 … +2,000 mm
6	Switching action delay **Note:** The time specification is absolute. The switching point thus varies according to the velocity of the robot.	-1,000 … +1,000 ms

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
```



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:

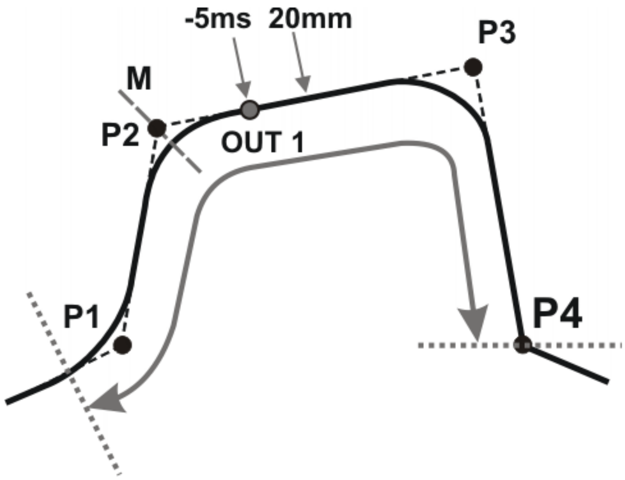
- The switching point can be brought forward, at most, as far as exact positioning point P1.
- The switching point can be delayed, at most, as far as the next exact positioning point P4. If P3 was an exact positioning point, the switching point could be delayed, at most, as far as P3.

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
```



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.

7.6.16 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.
   (>>> 7.6.17 "Inline form "SYN PULSE"" page 190)
4. Save the instruction by pressing the **Cmd Ok** softkey.

7.6.17 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.

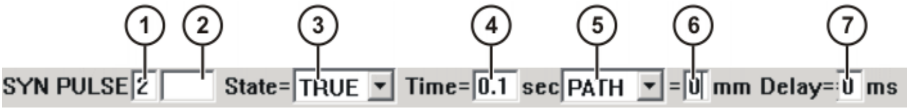


Fig. 7-23: Inline form "SYN PULSE"

Item	Description	Range of values
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.	Freely selectable
3	State to which the output is switched	TRUE, FALSE
4	Duration of the pulse	0.1 … 3 s
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. (>>> 7.6.14 "Inline form SYN OUT, option START/END" page 185) ■ **PATH**: The pulse is triggered at the end point of the motion block. See SYN OUT for examples and switching limits. (>>> 7.6.15 "Inline form SYN OUT, option PATH" page 187)	START, END, PATH
5	Distance from the switching point to the end point This box is only displayed if **PATH** has been selected under item 4.	-2,000 … +2,000 mm
6	Pulse delay. **Note:** The time specification is absolute. The switching point thus varies according to the velocity of the robot.	-1,000 … +1,000 ms

7.6.18 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.

8 Programming for user group “Expert” (KRL syntax)

8.1 Overview of KRL syntax

Variables and declarations	
DECL	(>>> 8.4.1 "DECL" page 196)
ENUM	(>>> 8.4.2 "ENUM" page 198)
IMPORT … IS	(>>> 8.4.3 "IMPORT ... IS" page 199)
STRUC	(>>> 8.4.4 "STRUC" page 199)

Motion programming	
CIRC	(>>> 8.5.1 "CIRC" page 201)
CIRC_REL	(>>> 8.5.2 "CIRC_REL" page 202)
LIN	(>>> 8.5.3 "LIN" page 204)
LIN_REL	(>>> 8.5.4 "LIN_REL" page 204)
PTP	(>>> 8.5.5 "PTP" page 206)
PTP_REL	(>>> 8.5.6 "PTP_REL" page 206)

Program execution control	
CONTINUE	(>>> 8.6.1 "CONTINUE" page 208)
EXIT	(>>> 8.6.2 "EXIT" page 208)
FOR … TO … ENDFOR	(>>> 8.6.3 "FOR ... TO ... ENDFOR" page 208)
GOTO	(>>> 8.6.4 "GOTO" page 209)
HALT	(>>> 8.6.5 "HALT" page 210)
IF … THEN … ENDIF	(>>> 8.6.6 "IF ... THEN ... ENDIF" page 210)
LOOP … ENDLOOP	(>>> 8.6.7 "LOOP ... ENDLOOP" page 211)
REPEAT … UNTIL	(>>> 8.6.8 "REPEAT ... UNTIL" page 211)
SWITCH … CASE … ENDSWITCH	(>>> 8.6.9 "SWITCH ... CASE ... ENDSWITCH" page 212)
WAIT … FOR	(>>> 8.6.10 "WAIT FOR" page 213)
WAIT … SEC	(>>> 8.6.11 "WAIT SEC" page 214)
WHILE … ENDWHILE	(>>> 8.6.12 "WHILE ... ENDWHILE" page 214)

Inputs/outputs	
ANIN	(>>> 8.7.1 "ANIN" page 215)
ANOUT	(>>> 8.7.2 "ANOUT" page 216)
DIGIN	(>>> 8.7.3 "DIGIN" page 217)
PULSE	(>>> 8.7.4 "PULSE" page 218)
SIGNAL	(>>> 8.7.5 "SIGNAL" page 222)

Subprograms and functions	
RETURN	(>>> 8.8.1 "RETURN" page 223)

Interrupt programming	
BRAKE	(>>> 8.9.1 "BRAKE" page 224)
INTERRUPT	(>>> 8.9.2 "INTERRUPT" page 224)
INTERRUPT … DECL … WHEN … DO	(>>> 8.9.3 "INTERRUPT ... DECL ... WHEN ... DO" page 225)
RESUME	(>>> 8.9.4 "RESUME" page 227)

Path-related switching actions (=Trigger)	
TRIGGER WHEN DISTANCE	(>>> 8.10.1 "TRIGGER WHEN DISTANCE" page 228)
TRIGGER WHEN PATH	(>>> 8.10.2 "TRIGGER WHEN PATH" page 232)

Communication
(>>> 8.11 "Communication" page 234)

System functions	
VARSTATE()	(>>> 8.12.1 "VARSTATE()" page 234)

Manipulating string variables
(>>> 8.13 "Manipulating string variables" page 236)

8.2 Symbols and fonts

The following symbols and fonts are used in syntax descriptions:

Description	Example
KRL code: ■ Courier font ■ Upper-case characters	`GLOBAL`; `ANIN ON`; `OFFSET`
Elements that must be replaced by program-specific entries: ■ Upper- and lower-case characters ■ Italics	*Distance*; *Time*; *Format*
Optional elements: ■ In angle brackets	< ... >
Elements that are mutually exclusive: ■ Separated by the "|" symbol	`IN |OUT`

8.3 Important KRL terms

8.3.1 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.

8.3.2 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 (>>> 8.3.5 "Areas of validity" page 195).
- **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.

8.3.3 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 sequences 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. (>>> 8.1 "Overview of KRL syntax" page 191)

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

8.3.4 Data types

There are two 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**

  Important predefined data types are the **simple data types** and the **data types for motion programming**.

The following **simple data types** are predefined in the KSS:

Data type	Keyword	Meaning	Range of values	Example
Integer	`INT`	Integer	$-2^{31}-1 \ldots 2^{31}-1$	32
Real	`REAL`	Floating-point number	+1.1E-38 ... +3.4E+38	1.43
Boolean	`BOOL`	Logic state	TRUE, FALSE	TRUE
Character	`CHAR`	Character	ASCII character	“A”

The following **data types for motion programming** are predefined in the KSS:

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
```

8.3.5 Areas of validity

Local

Data object	Area of validity
Variable (no constant variable)	Valid in the program code between DEF and ENDDEF containing the declaration of the variables.
Constant variable	Valid in the module to which the data list, in which the constant was declared, belongs.
User-defined data type	If the data type has been defined in a DAT file: valid in the SRC file that belongs to the DAT file. If the data type has been defined in an SRC file: valid at, or below, the program level in which it was declared.
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.

The keyword GLOBAL must only be used in data lists.
Precondition: 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 globally valid if they were declared 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 ()*

8.3.6 Constant variables

The value of a constant variable can no longer be modified during program execution after initialization. Constant variables can be used to prevent a value from being changed accidentally during program execution.

Constant variables 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

8.4 Variables and declarations

8.4.1 DECL

Description

Declaration of variables, arrays and constant variables

Syntax

Declaration of variables

Declaration of variables in programs:

`<DECL>` *DataType Name1 <, ..., NameN>*

Declaration of variables in data lists:

`<DECL> <GLOBAL>` *DataType Name1 <, ..., NameN>*

Declaration of variables in data lists with simultaneous initialization:

`<DECL> <GLOBAL>` *DataType 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> *DataType Name1 [Dimension1 <, ..., Dimension3>] <, ..., NameN [DimensionN1 <,..., DimensionN3>] >*

Declaration of arrays in data lists:

<DECL> <GLOBAL> *DataType 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> *DataType 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 *DataType 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 *DataType* is a predefined data type. If *DataType* is a user-defined data type, then DECL is obligatory.
GLOBAL	(>>> 8.3.5 "Areas of validity" page 195)
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
DataType	Specification of the desired data type
Name	Name of the object (variable, array or constant variable) 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 *DataType*, 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"
```

8.4.2 ENUM

Description

Definition of an enumeration type (= ENUM data type)

Syntax

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

Explanation of the syntax

Element	Description
GLOBAL	(>>> 8.3.5 "Areas of validity" page 195)
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

En 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
```

8.4.3 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
DataType	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[,,]*
DataSource	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
```

8.4.4 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	(>>> 8.3.5 "Areas of validity" page 195)
NameStructureType	Name of the new structure type. The names of user-defined data types should end in _TYPE, to distinguish them from variable names.
DataType	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}}
```

8.5 Motion programming

8.5.1 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 The auxiliary point must be specified in Cartesian coordinates. The coordinates refer to the BASE coordinate system. If not all components of the auxiliary point are specified, the controller takes the values of the current 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.
End_Point	Type: POS, E6POS, FRAME The end point must be specified in Cartesian coordinates. The coordinates refer to the BASE coordinate system. If not all components of the end point are specified, the controller takes the values of the current 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.

Element	Description
Circular_Angle	Specifies the overall length of the arc. This makes it possible to extend the arc 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
```

8.5.2 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.
Circular_Angle	Specifies the overall length of the arc. This makes it possible to extend the arc 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
```

8.5.3 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 The end point must be specified in Cartesian coordinates. The coordinates refer to the BASE coordinate system. If not all components of the end point are specified, the controller takes the values of the current position for the missing components. 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.

Example

End point with two components. For the rest of the components, the controller takes the values of the current position.

```
LIN {Z 500,X 123.6}
```

8.5.4 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
```

8.5.5 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 current position for the missing components.
`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 current position.

```
PTP {Z 500,X 123.6}
```

8.5.6 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}
```

8.6 Program execution control

8.6.1 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.

8.6.2 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
```

8.6.3 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
```

8.6.4 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
```

8.6.5 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`

8.6.6 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
```

8.6.7 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
```

8.6.8 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
```

8.6.9 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
```

8.6.10 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
```

8.6.11 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
```

8.6.12 WHILE ... ENDWHILE

Description

Rejecting loop. Loop that is repeated as long as a certain condition is fulfilled.

If the condition is not met, program execution is resumed at the first statement after the ENDWHILE line. The condition is checked before each loop execution. If the condition is not already fulfilled beforehand, the statement block is not executed.

Loops can be nested. In the case of nested loops, the outer loop is executed completely first. The inner loop is then executed completely.

Syntax

`WHILE` *Repetition_Condition*

Statement block

`ENDWHILE`

Explanation of the syntax

Element	Description
Repetition_Condition	Type: BOOL Possible: ■ Variable of type BOOL ■ Function of type BOOL ■ Logic operation, e.g. a comparison, with a result of type BOOL

Example 1

The loop is executed 99 times. After execution of the loop, `W` has the value 100.

```
W=1
WHILE W<100
   W=W+1
ENDWHILE
```

Example 2

The loop is executed as long as `$IN[1]` is true.

```
WHILE $IN[1]==TRUE
   W=W+1
ENDWHILE
```

8.7 Inputs/outputs

8.7.1 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.

Element	Description
Signal_Name	Type: REAL Specifies the analog input. *Signal_Name* must first have been declared with `SIGNAL` (>>> 8.7.5 "SIGNAL" page 222). 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 * SIGNAL_1
```

The cyclical scanning of `SIGNAL_1` is ended using the `ANIN OFF` statement.

```
ANIN OFF SIGNAL_1
```

8.7.2 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 (>>> 8.7.5 "SIGNAL" page 222). 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.

Element	Description
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
```

8.7.3 DIGIN

Description

Cyclical reading (every 12 ms) of digital inputs. DIGIN triggers an advance run stop.

If there are array indices in a DIGIN ON statement (e.g. if *Factor* is an array), they are only evaluated once. The expression that is produced after the array indices have been replaced by numeric values is cyclically evaluated.

Syntax

Starting cyclical reading:

`DIGIN ON` *Value* `=` *Factor* `* $DIGINx` <±*Offset*>

Ending cyclical reading:

`DIGIN OFF $DIGINx`

- A maximum of two DIGIN ON statements can be used at the same time.
- A maximum of two DIGIN ON statements can write to the same variable `Value` or access the same digital input.
- All of the variables used in a DIGIN statement must be declared in data lists (locally or in $CONFIG.DAT).
- The robot controller has 6 digital inputs ($DIGIN1 … $DIGIN6).

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.
Offset	Type: REAL It can be a constant, variable or signal name.

Example

The inputs $IN[1020] to $IN[1026] are grouped together as digital input $DIGIN1. The analog input $ANIN[1] is assigned the name FACTOR. The result of the expression to the right of the equals sign is assigned to the variable VAR.

```
SIGNAL $DIGIN1 $IN[1020] TO $IN[1026]
SIGNAL FACTOR $ANIN[1]
DIGIN ON VAR = FACTOR * $DIGIN1 + OFFSET
```

DIGIN OFF deactivates the cyclical reading of the digital input.

```
DIGIN OFF $DIGIN1
```

8.7.4 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 (>>> 8.7.5 "SIGNAL" page 222)
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

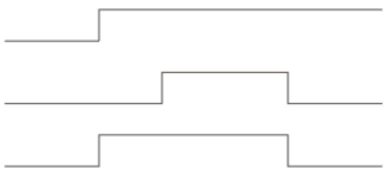


Fig. 8-1: $OUT+PULSE, example 1

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

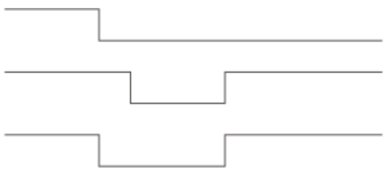


Fig. 8-2: $OUT+PULSE, example 2

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

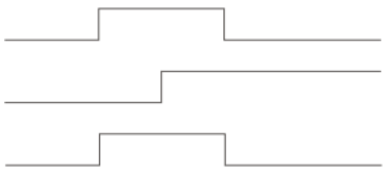


Fig. 8-3: PULSE+$OUT, example 1

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

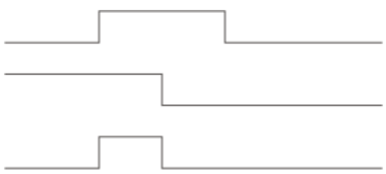


Fig. 8-4: PULSE+$OUT, example 2

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

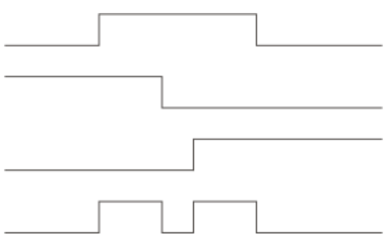


Fig. 8-5: PULSE+$OUT, example 3

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):

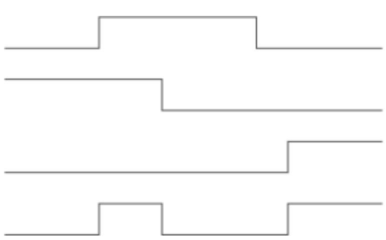


Fig. 8-6: PULSE+$OUT, example 4

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

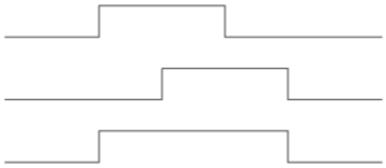


Fig. 8-7: PULSE+PULSE, example 1

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

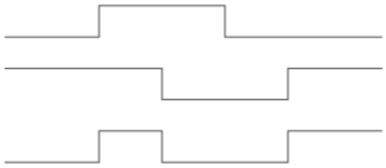


Fig. 8-8: PULSE+PULSE, example 2

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

Actual pulse characteristic at output 50

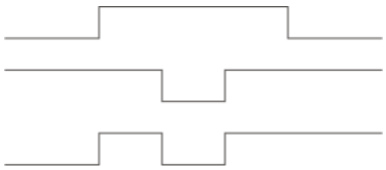


Fig. 8-9: PULSE+PULSE, example 3

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

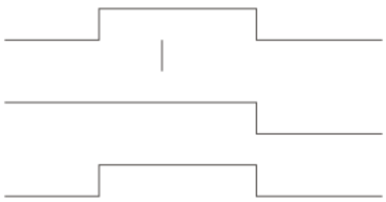


Fig. 8-10: PULSE+END, example

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

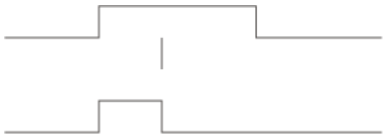


Fig. 8-11: PULSE+RESET, example

8.7.5 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] ■ $DIGINx ■ $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 `SWITCH`. The output $OUT[7] is set.

```
SIGNAL SWITCH $OUT[7]
SWITCH = TRUE
```

Example 2

The outputs $OUT[1] to $OUT[8] are combined to form one digital output under the name OUTWORD. The outputs $OUT[3], $OUT[4], $OUT[5] and $OUT[7] are set.

```
SIGNAL OUTWORT $OUT[1] TO $OUT[8]
OUTWORT = 'B01011100'
```

8.8 Subprograms and functions

8.8.1 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
```

8.9 Interrupt programming

8.9.1 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

(>>> 8.9.2 "INTERRUPT" page 224)

8.9.2 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. (>>> 8.9.3 "INTERRUPT ... DECL ... WHEN ... DO" page 225)

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 for as long as the subprogram is being activated.

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
```

8.9.3 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
(>>> 8.9.4 "RESUME" page 227).

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!
(>>> 8.9.2 "INTERRUPT" page 224)

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 Digital inputs must not be used.
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.

The keyword GLOBAL must only be used in data lists.
Precondition: 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
```

8.9.4 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.

The first motion statement after RESUME should not be a CIRC motion, as the start point is different each time, resulting in different circular motions.

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.

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
```

8.10 Path-related switching actions (=Trigger)

8.10.1 TRIGGER WHEN DISTANCE

Description

Triggers a defined statement. This statement is triggered either at the start point or at the end point of the motion block in which the Trigger is situated in the program. The statement is executed parallel to the robot motion.

It is possible to shift the statement in time so that it is not triggered exactly at the start or end point, but earlier or later.

Syntax

`TRIGGER WHEN DISTANCE=`*Position* `DELAY=`*Time* `DO` *Statement* `<PRIO=`*Priority*`>`

Explanation of the syntax

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Position</td><td>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.</td></tr>
<tr><td>Time</td><td>Type: INT; 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.</td></tr>
</table>

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 for as long as the subprogram is being activated. Up to 16 interrupts may be active at any one time.

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

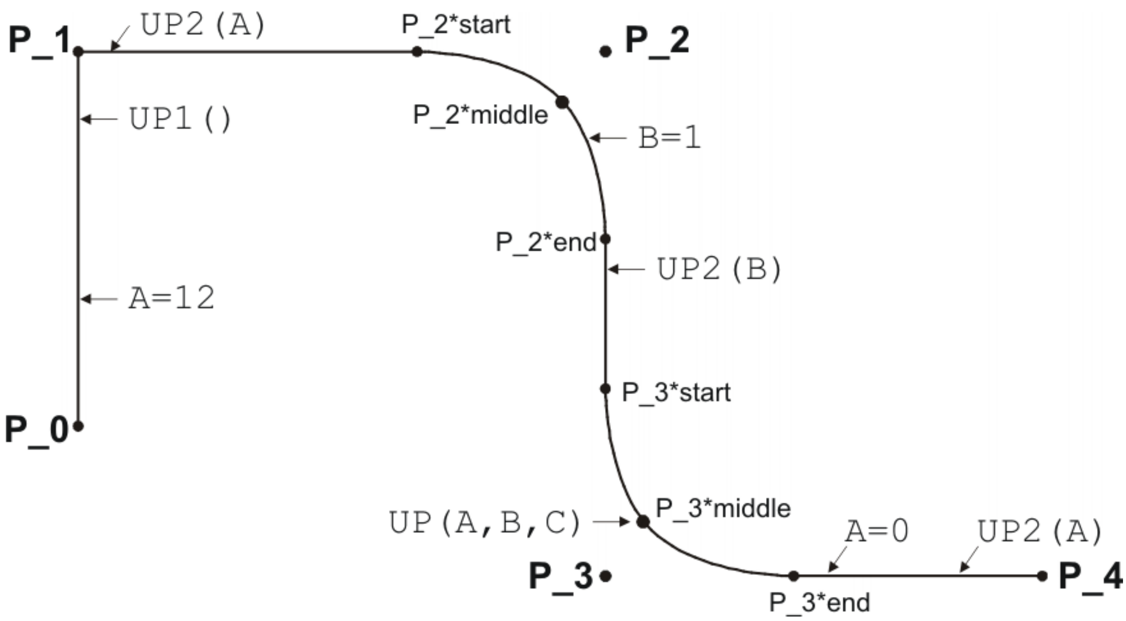


Fig. 8-12: Example of TRIGGER WHEN DISTANCE

8.10.2 TRIGGER WHEN PATH

Description

Triggers a defined statement. The statement is triggered at the end point of the motion block in which the Trigger is situated in the program.

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

It is possible to shift the statement in time and/or space so that it is not triggered exactly at the end point, but before or after it.

The statement is executed parallel to the robot motion.

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: INT; 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 for as long as the subprogram is being activated. Up to 16 interrupts may be active at any one time.

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.

If, in the case of a combination of *Distance* and *Time*, one of the two values exceeds the switching range, this is irrelevant, as long as the overall shift does not exceed the switching range.

Example

```
LIN P_2 C_DIS
TRIGGER WHEN PATH = -20 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.

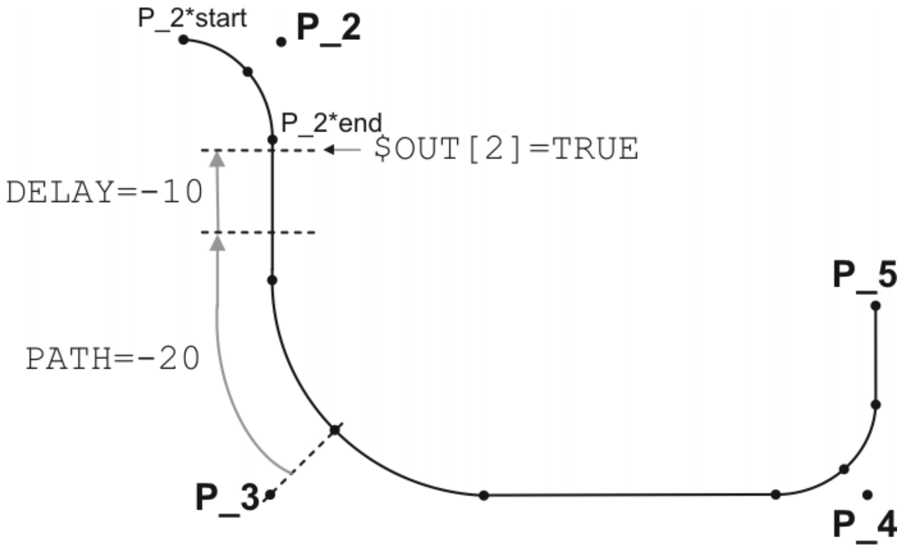


Fig. 8-13: Example of TRIGGER WHEN PATH

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.

8.11 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

8.12 System functions

8.12.1 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
  DEVCONTROL 1
ENDIF
...
IF VARSTATE("MYVAR")==#DECLARED THEN
  DEVCONTROL 2
ENDIF
...
IF VARSTATE("ANYVAR")==#UNKNOWN THEN
  DEVCONTROL 3
ENDIF
...
MYVAR=9
...
IF VARSTATE("MYVAR")==#DECLARED THEN
  DEVCONTROL 4
ENDIF
...
IF VARSTATE("MYVAR")==#INITIALIZED THEN
  DEVCONTROL 5
ENDIF
...
END
```

`DEVCONTROL` *Number* generates a notification message with the text "CONTROL: *Number*". *Number* must be of type INT.

Explanation of the state monitoring:

- The first IF condition is false, as MYVAR has already been declared. The message "CONTROL: 1" is not generated.
- The second IF condition is true, as MYVAR has been declared. The message "CONTROL: 2" is generated.
- The third IF condition is true, on the condition that there is also no variable with the name ANYVAR in $CONFIG.DAT. The message "CONTROL: 3" is generated.
- The third IF condition is false, as MYVAR has not only been declared, but has also already been initialized here. The message "CONTROL: 4" is not generated.
- The fourth IF condition is true, as MYVAR has been initialized. The message "CONTROL: 5" is generated.

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
  DEVCONTROL 100
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.

8.13 Manipulating string variables

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

Function	Description
String variable length in the declaration	(>>> 8.13.1 "String variable length in the declaration" page 236)
String variable length after initialization	(>>> 8.13.2 "String variable length after initialization" page 237)
Deleting the contents of a string variable	(>>> 8.13.3 "Deleting the contents of a string variable" page 237)
Extending a string variable	(>>> 8.13.4 "Extending a string variable" page 238)
Searching a string variable	(>>> 8.13.5 "Searching string variables" page 238)
Comparing the contents of string variables	(>>> 8.13.6 "Comparing the contents of string variables" page 239)
Copying a string variable	(>>> 8.13 "Manipulating string variables" page 236)

8.13.1 String variable length in the declaration

Function

`StrDeclLen(StrVar[])`

Description

This function determines the length of a string variable according to its declaration in the declaration section of a program. The determined length is output as the return value.

Element	Data type	Description
`StrDeclLen`	INT	Length of the string variable as declared in the declaration section
`StrVar[]`	CHAR	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 StrLaenge
   …
3  StrLaenge = StrDeclLen(ProName)
4  StrLaenge = StrDeclLen($Trace.Name[ ])
```

Line	Description
3	StrLaenge = 24
4	StrLaenge = 7

8.13.2 String variable length after initialization

Function

`StrLen(StrVar)`

Description

This function determines the length of the character string of a string variable as defined in the initialization section of the program. The determined length is output as the return value.

Element	Data type	Description
`StrLen`	INT	Number of characters currently assigned to the string variable
`StrVar`	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

8.13.3 Deleting the contents of a string variable

Function

`StrClear(StrVar[])`

Description

This function deletes the contents of a string variable. The return value, once the contents of the variable have been successfully deleted, is TRUE.

Element	Data type	Description
`StrClear`	BOOL	Deletes the character string in the relevant string variable
`StrVar[]`	CHAR	Variable whose character string is to be deleted The string variable StrVar[ ] may only be an array of type CHAR.

Example

```
IF (NOT StrClear($Loop_Msg[]) THEN
HALT
ENDIF
```

8.13.4 Extending a string variable

Function

`StrAdd(StrDest[], StrToAdd[])`

Description

The contents of a string variable can be extended by inserting another string variable. The return value that is generated is the sum of the character strings StrDest[ ] and StrToAdd[ ]. If the sum is longer than the previously defined length of StrDest[ ], the return value 0 is generated. This is also the case if the sum exceeds the permissible limit of 470 characters.

Element	Data type	Description
`StrAdd`	INT	Extends the specified string variable
`StrDest[]`	CHAR	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[]`	CHAR	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

8.13.5 Searching string variables

Function

`StrFind(StartAt, StrVar[], StrFind[], CaseSens)`

Description

It is possible to search the string variable for a particular character string. This search can be case-sensitive. The position of the first character found is output as the return value.

Element	Data type	Description
`StrFind`	INT	Searches the specified string variable.
`StartAt`	INT	Starts the search from this position.
`StrVar[]`	CHAR	The string variable to be searched.
`StrFind[]`	CHAR	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

```
DECL CHAR A[5]
INT B
A[]="ABCDE"
B = StrFind(1, A[], "AC", #CASE_SENS)
B = StrFind(1, A[], "a", #NOT_CASE_SENS)
B = StrFind(1, A[], "BC", #Case_Sens)
B = StrFind(1, A[], "bc", #NOT_CASE_SENS)
```

Line	Description
4	B = 0
5	B = 1
6	B = 2
7	B = 2

8.13.6 Comparing the contents of string variables

Function

`StrComp(StrComp1[], StrComp2[], CaseSens)`

Description

The character strings of two string variables can be compared with one another. Upper/lower case can be taken into consideration. If the character strings match, the return value is TRUE, otherwise FALSE.

Element	Data type	Description
`StrComp`	BOOL	Compares two string variables.
`StrComp1[]`	CHAR	String variable is compared with StrComp2[ ].
`StrComp2[]`	CHAR	String variable 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[], "abcde", #Case_Sens)
7  B = StrComp(A[], "acbde", #NOT_CASE_SENS)
```

Line	Description
4	B = TRUE
5	B = TRUE
6	B = FALSE
7	B = FALSE

8.13.7 Copying a string variable

Function

`StrCopy(StrDest[], StrSource[])`

Description

The contents of a string variable are copied to another string variable. If the copying is successful, the return value is TRUE, otherwise this value is FALSE.

Element	Data type	Description
`StrCopy`	BOOL	Copies a character string to a specified string variable.
`StrDest[]`	INT	The character string is assigned to this string variable. Since StrDest[ ] is an array of type CHAR, individual characters and constants are not permissible.
`StrSource[]`	CHAR	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[ ] is assigned the value “Example” and B is set to the return value TRUE.

9 Diagnosis

9.1 Overview of diagnosis

Topic	Display functions
Logbook	(>>> 9.2.1 "Displaying the log-book" page 241)
Caller stack	(>>> 9.3 "Displaying the caller stack" page 243)
Interrupts	(>>> 9.4 "Displaying interrupts" page 243)

9.2 Logbook

9.2.1 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**.

The following tabs are available:

- Log (>>> 9.2.2 "Log tab" page 241)
- Filter (>>> 9.2.3 "Filter tab" page 242)

9.2.2 Log tab

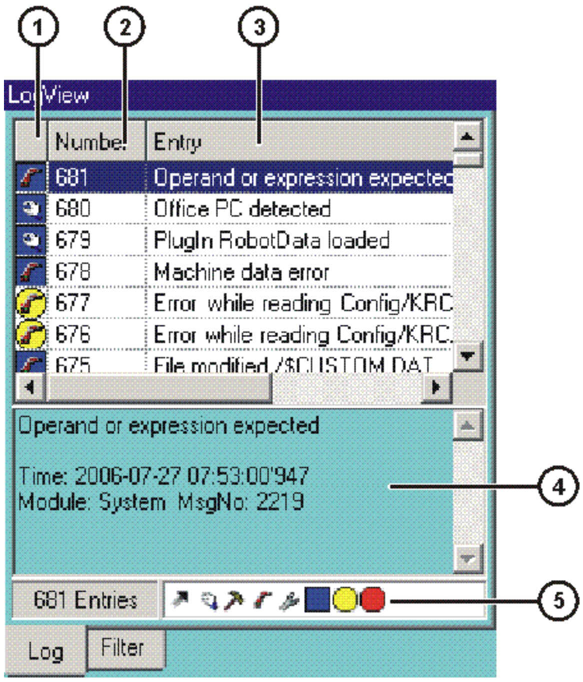


Fig. 9-1: Logbook, Log tab

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.

9.2.3 Filter tab

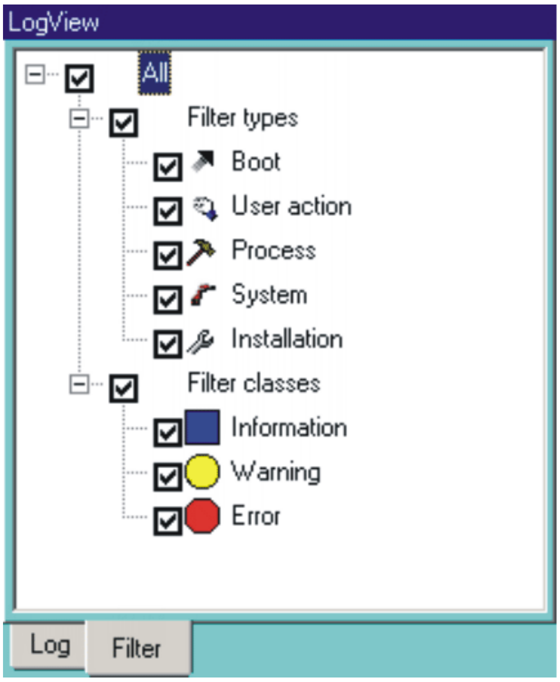


Fig. 9-2: Logbook, Filter tab

The following softkeys are available:

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

9.3 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

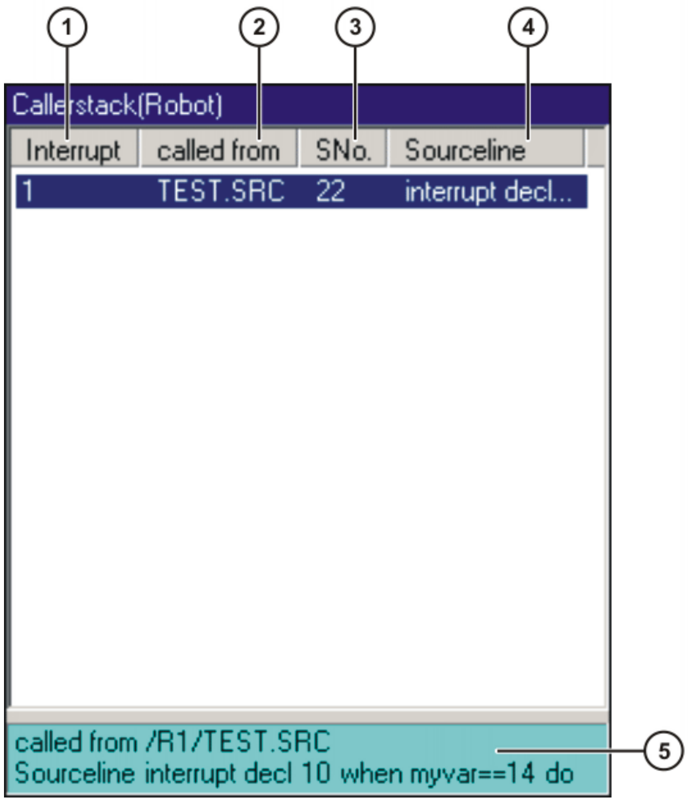


Fig. 9-3: Caller Stack window

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

9.4 Displaying interrupts

Precondition

- User group "Expert"

Procedure

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

Description

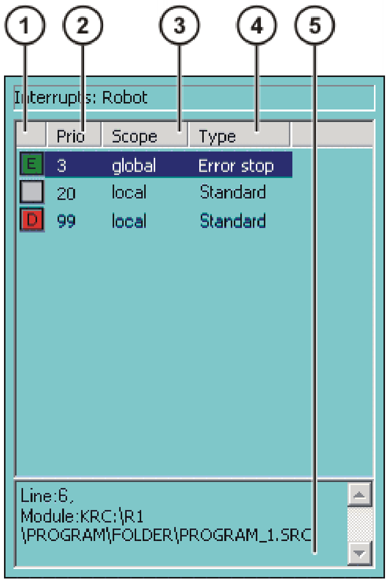


Fig. 9-4: Interrupts

Column	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-induced stop: $STOPMESS ■ E-STOP: $ALARM_STOP ■ Fast Measurement: $SMEAS_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.

10 Messages

10.1 System messages

Details about the system messages can be found in the online help.

(>>> 4.2.5 "Calling online help" page 39)

10.2 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	A program number was sent by the host computer for which no branch for execution has (yet) been created in the CELL.SRC control structure.
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”.

11 KUKA Service

11.1 Requesting support

Introduction

The KUKA Robot Group documentation offers information on operation and provides assistance with troubleshooting. For further assistance, please contact your local KUKA subsidiary.

Faults leading to production downtime are to 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

11.2 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

Austria	KUKA Roboter 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
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

France	KUKA Automatisme + Robotique SAS Techvallée 6 Avenue du Parc 91140 Villebon s/Yvette France Tel. +33 1 6931-6600 Fax +33 1 6931-6601 commercial@kuka.fr www.kuka.fr
Germany	KUKA Roboter GmbH Blücherstr. 144 86165 Augsburg Germany Tel. +49 821 797-4000 Fax +49 821 797-1616 info@kuka-roboter.de www.kuka-roboter.de
Hungary	KUKA Robotics Hungaria Kft. Fö út 140 2335 Taksony Hungary Tel. +36 24 501609 Fax +36 24 477031 info@kuka-robotics.hu
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

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
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	KUKA-VAZ Engineering Jushnoje Chaussee, 36 VAZ, PTO 445633 Togliatti Russia Tel. +7 8482 391249 or 370564 Fax +7 8482 736730 Y.Klychkov@VAZ.RU
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
Spain	KUKA Sistemas de Automatización S.A. Pol. Industrial Torrent de la Pastera Carrer del Bages s/n 08800 Vilanova i la Geltrú (Barcelona) Spain Tel. +34 93 814-2353 Fax +34 93 814-2950 Comercial@kuka-e.com www.kuka-e.com
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

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

UK

KUKA Automation + Robotics
Hereward Rise
Halesowen
B62 8AN
UK
Tel. +44 121 585-0800
Fax +44 121 585-0900
sales@kuka.co.uk

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

Index

X