Training

KUKA Roboter GmbH

Robot Programming 1

KUKA System Software 8.2

Training Documentation, KUKA Roboter GmbH

Issued: 31.05.2011

Version: COL P1KSS8 Roboterprogrammierung 1 V1 en

Translation of the original documentation

KIM-PS5-DOC

Publication:	Pub COLLEGE P1KSS8 Roboterprogrammierung 1 en
Bookstructure:	P1KSS8 Roboterprogrammierung 1 V4.2
Label:	COL P1KSS8 Roboterprogrammierung 1 V1 en

Contents

1 Structure and function of a KUKA robot system

1.1 Introduction to robotics

What is a robot?

The term *robot* comes from the Slavic word *robota*, meaning *hard work*.

According to the official definition of an industrial robot: “A robot is a freely programmable, program-controlled handling device”.

The robot thus also includes the controller and the operator control device, together with the connecting cables and software.

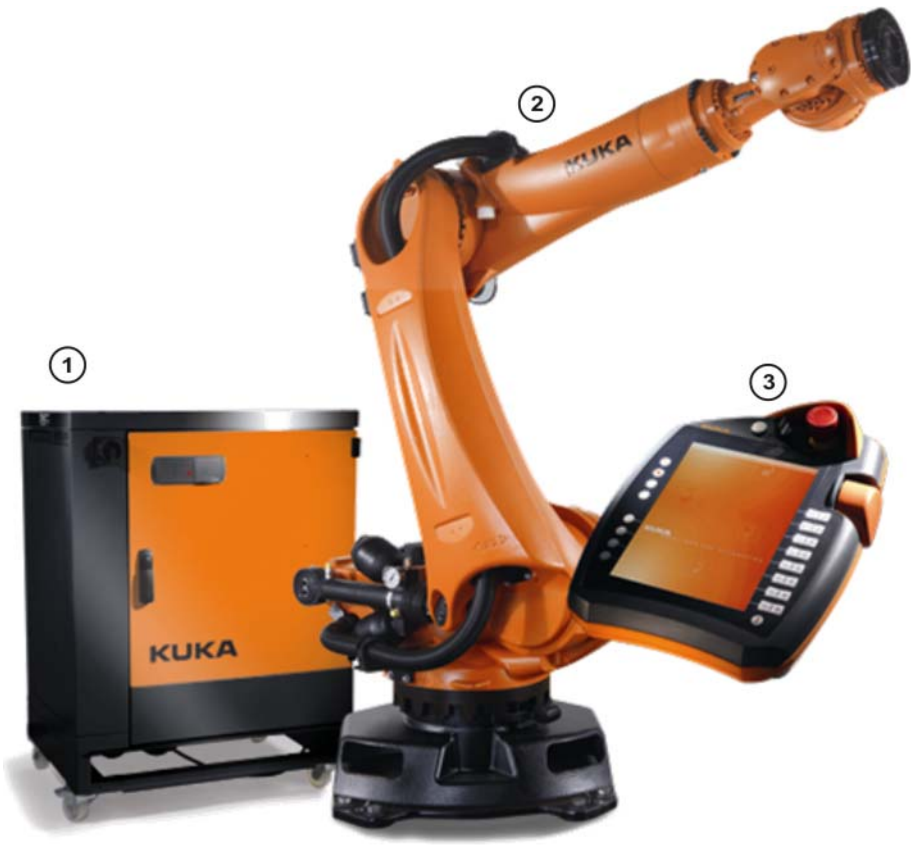


Fig. 1-1: Industrial robot

1 Controller ((V)KR C4 control cabinet)
2 Manipulator (robot arm)
3 Teach pendant (KUKA smartPAD)

Everything outside the system limits of the industrial robot is referred to as the *periphery*:

- Tooling (end effector/tool)
- Safety equipment
- Conveyor belts
- Sensors
- etc.

1.2 Robot arm of a KUKA robot

What is a manipulator?

The manipulator is the actual robot arm. It consists of a number of moving links (axes) that are linked together to form a “kinematic chain”.

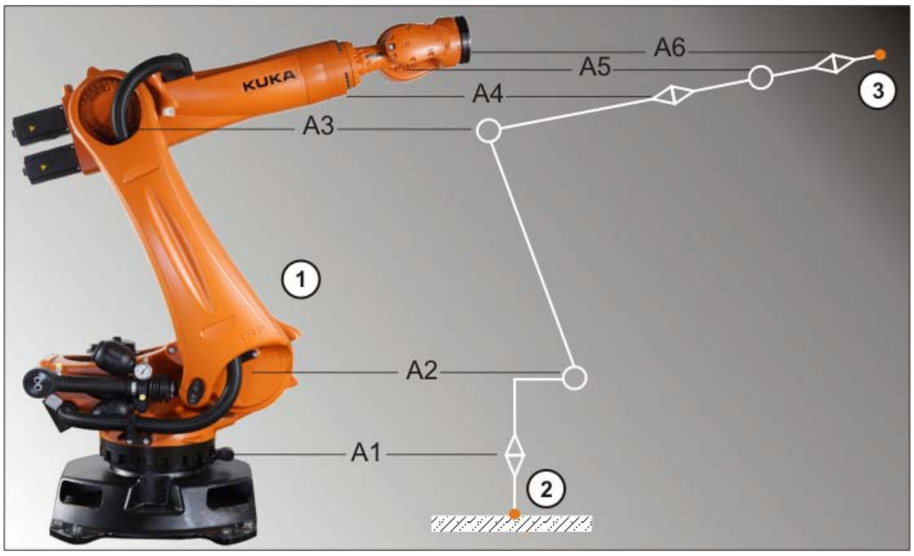


Fig. 1-2: Manipulator

1	Manipulator (robot arm)
2	Start of the kinematic chain: base of the robot (ROBROOT)
3	Free end of the kinematic chain: flange (FLANGE)
A1 ... A6	Robot axes 1 to 6

The individual axes are moved by means of targeted actuation of servomotors. These are linked to the individual components of the manipulator via reduction gears.

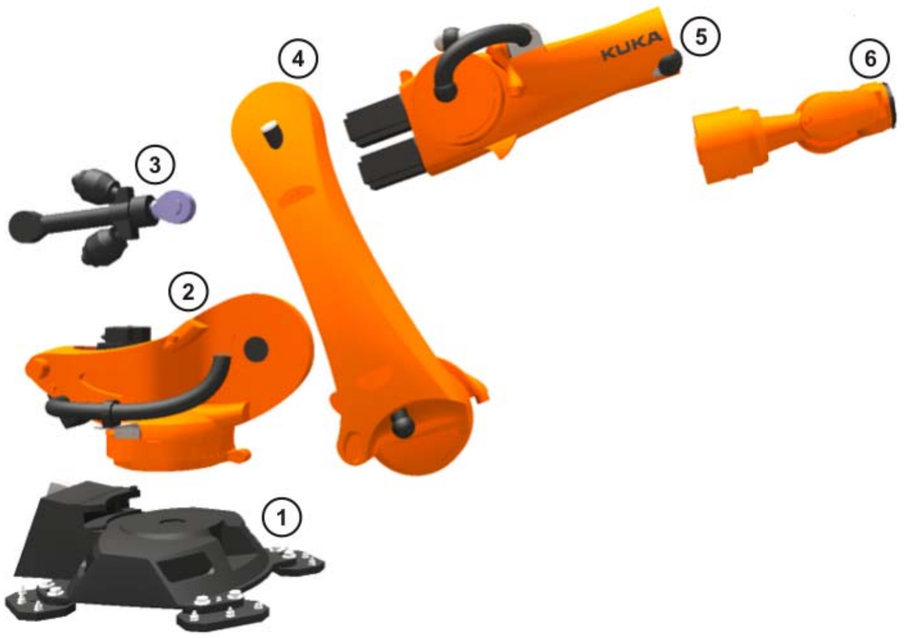


Fig. 1-3: Overview of manipulator components

1	Base frame	4	Link arm
2	Rotating column	5	Arm
3	Counterbalancing system	6	Wrist

The components of a robot arm consist primarily of cast aluminum and steel. In isolated cases, carbon-fiber components are also used.

The individual axes are numbered from bottom (robot base) to top (robot flange):

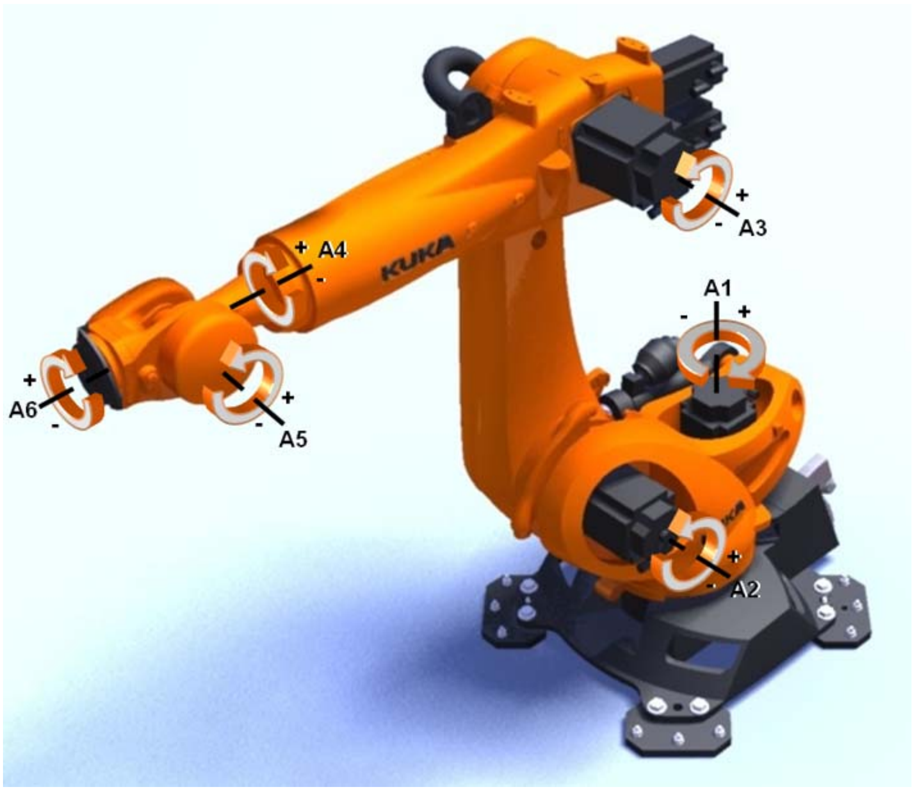
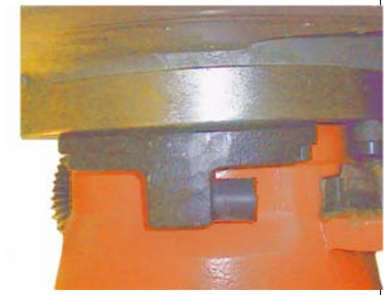


Fig. 1-4: Degrees of freedom of a KUKA robot

Excerpt from the technical data of manipulators from the KUKA product range

- **Number of axes:** 4 (SCARA and parallelogram robots) to 6 (standard vertical jointed-arm robots)
- Reach: from 0.35 m (KR 5 scara) to 3.9 m (KR 120 R3900 ultra K)
- **Weight:** from 20 kg to 4700 kg.
- **Accuracy:** 0.015 mm to 0.2 mm repeatability.

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

Axis 1	Axis 2	Axis 3
		

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

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

1.3 (V)KR C4 robot controller

Who controls motion?

The manipulator is moved by means of servomotors controlled by the (V)KR C4 controller.

Fig. 1-5: (V)KR C4 control cabinet

Properties of the (V)KR C4 controller

- Robot control (path planning): control of six robot axes plus up to two external axes.

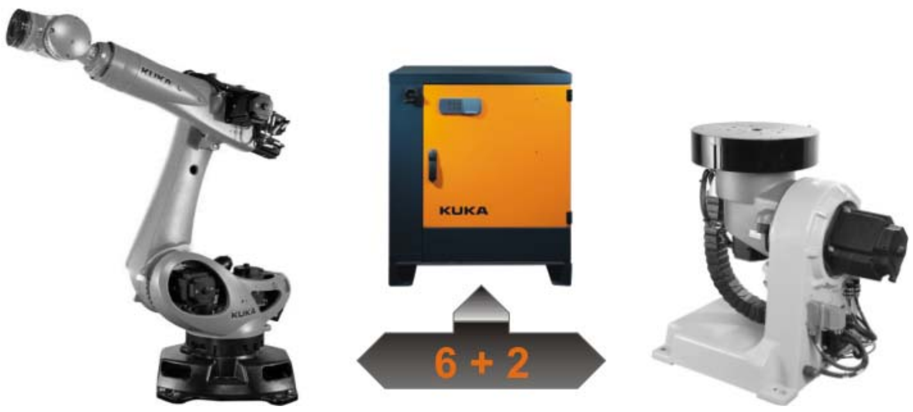


Fig. 1-6: (V)KR C4 axis control

- Sequence control: integrated Soft PLC in accordance with IEC61131
- Safety controller
- Motion control

- Communication options via bus systems (e.g. ProfiNet, Ethernet IP, Interbus):
  - Programmable logic controllers (PLC)
  - Additional controllers
  - Sensors and actuators
- Communication options via network:
  - Host computer
  - Additional controllers

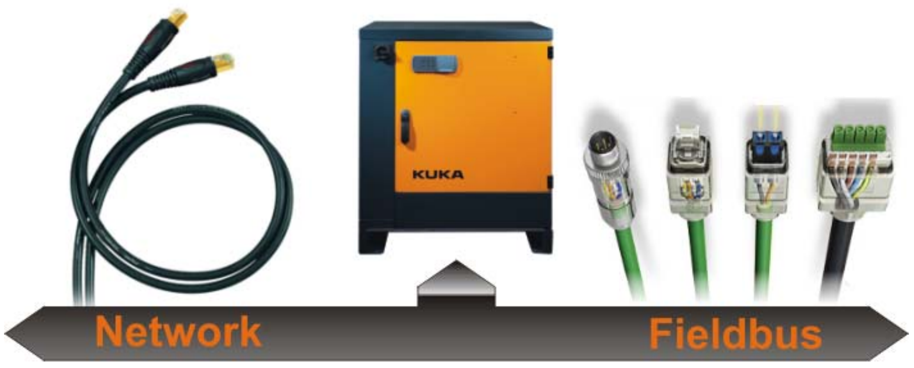


Fig. 1-7: (V)KR C4 communication options

1.4 The KUKA smartPAD

How is a KUKA robot operated?

The KUKA robot is operated by means of the KUKA smartPAD teach pendant.

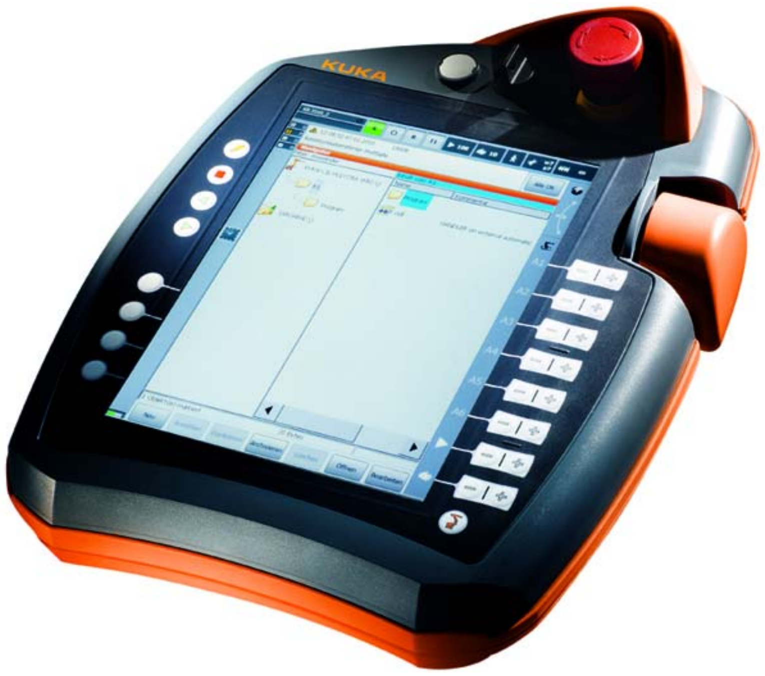


Fig. 1-8

Features of the KUKA smartPAD:

- Touch screen (touch-sensitive user interface) for operation by hand or using the integrated stylus
- Large display in portrait format
- KUKA menu key
- Eight jog keys

- Keys for operator control of the technology packages
- Program execution keys (Stop/Backwards/Forwards)
- Key for displaying the keypad
- Keyswitch for changing the operating mode
- EMERGENCY STOP button
- Space Mouse
- Unpluggable
- USB connection

1.5 Overview of smartPAD

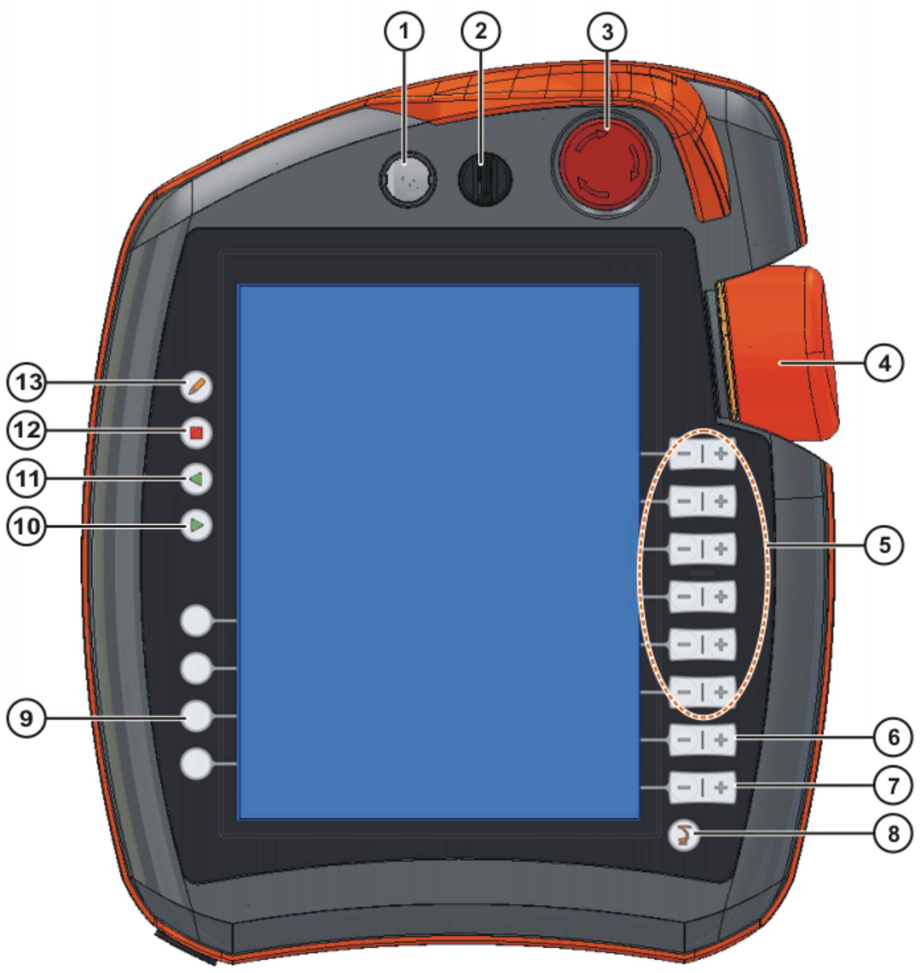


Fig. 1-9

Item	Description
1	Button for disconnecting the smartPAD
2	Keyswitch for calling the connection manager. The switch can only be turned if the key is inserted. The connection manager is used to change the operating mode.
3	EMERGENCY STOP button. Stops the robot in hazardous situations. The EMERGENCY STOP button locks itself in place when it is pressed.

Item	Description
4	Space Mouse. For moving the robot manually.
5	Jog keys. For moving the robot manually.
6	Key for setting the program override
7	Key for setting the jog override
8	Main menu key. Shows the menu items on the smartHMI.
9	Technology keys. The technology keys are used primarily for setting parameters in technology packages. Their exact function depends on the technology packages installed.
10	Start key. The Start key is used to start a program.
11	Start backwards key. The Start backwards key is used to start a program backwards. The program is executed step by step.
12	STOP key. The STOP key is used to stop a program that is running.
13	Keyboard key Displays the keyboard. It is generally not necessary to press this key to display the keyboard, as the smartHMI detects when keyboard input is required and displays the keyboard automatically.

1.6 Robot programming

A robot is programmed so that motion sequences and processes can be executed automatically and repeatedly. For this, the controller requires a large amount of information:

- Robot position = position of the tool in space.
- Type of motion
- Velocity / acceleration
- Signal information for wait conditions, branches, dependencies, etc.

What language does the controller speak?

The programming language is **KRL** - **K**UKA **R**obot **L**anguage

Example program:

```
PTP P1 Vel=100% PDAT1
PTP P2 CONT Vel=100% PDAT2
WAIT FOR IN 10 'Part in Position'
PTP P3 Vel=100% PDAT3
```

How is a KUKA robot programmed?

Various programming methods can be used for programming a KUKA robot:

- **Online programming** with the teaching method.

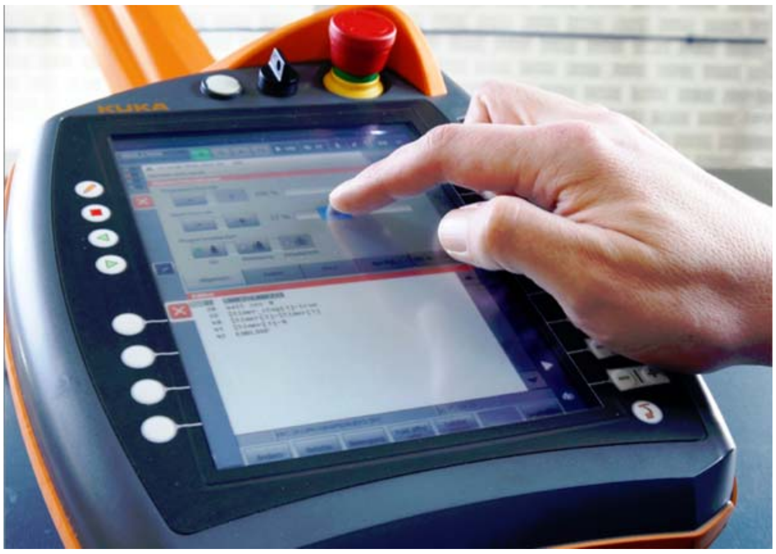


Fig. 1-10: Robot programming with the KUKA smartPAD

- **Offline programming**
  - **Interactive, graphics-based programming:** simulation of the robot process.

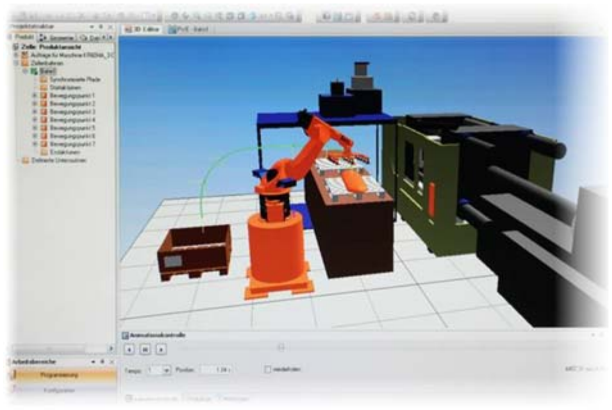


Fig. 1-11: Simulation with KUKA WorkVisual

  - **Text-based programming:** programming with the aid of the smart-PAD user interface display on a higher-level control PC (also for diag-nosis, online adaptation of programs that are already running)

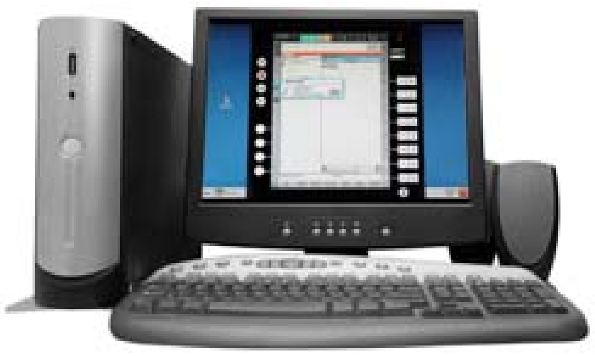


Fig. 1-12: Robot programming with KUKA OfficeLite

1.7 Robot safety

A robot system must always have suitable safety features. These include, for example, physical safeguards (fences, gates, etc.), EMERGENCY STOP buttons, dead-man switches, axis range limitations, etc.

Example: College training cell

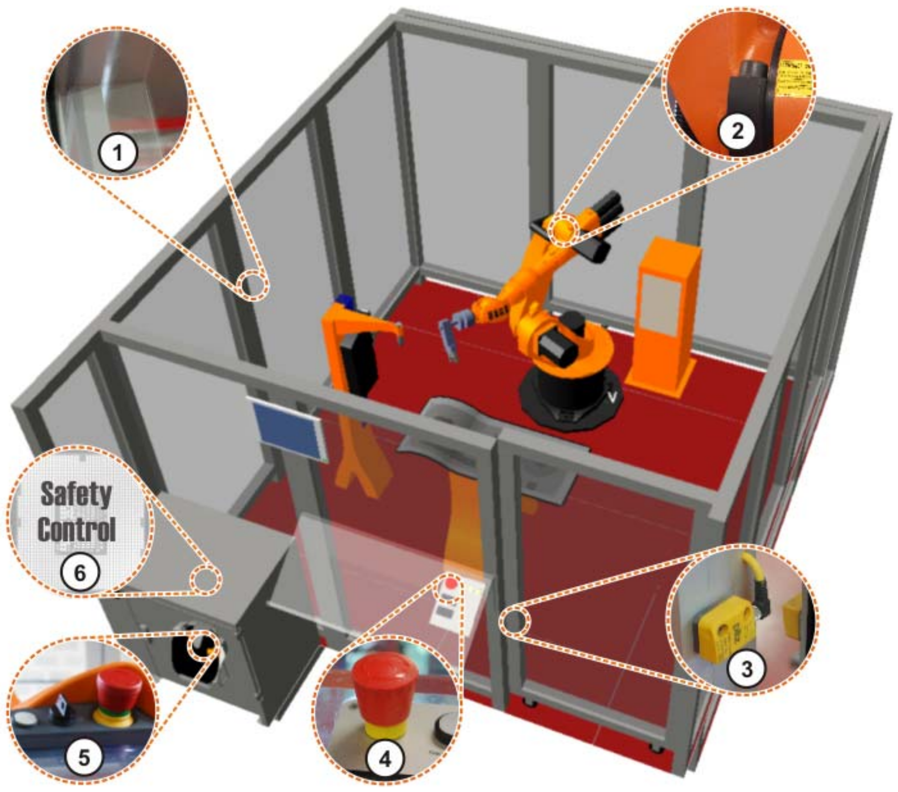


Fig. 1-13: Training cell

1 Safety fence
2 Mechanical end stops or axis range limitation for axes 1, 2 and 3
3 Safety gate with contact for monitoring the closing function
4 EMERGENCY STOP button (external)
5 EMERGENCY STOP button, enabling switch, keyswitch for calling the connection manager
6 Integrated (V)KR C4 safety controller

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

EMERGENCY STOP device

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

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

- The manipulator and any external axes (optional) are stopped with a safety stop 1.

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

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

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

External E-STOP

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

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

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

Operator safety

The operator safety signal is used for interlocking physical safeguards, e.g. safety gates. Automatic operation is not possible without this signal. In the event of a loss of signal during automatic operation (e.g. safety gate is opened), the manipulator stops with a safety stop 1.

Operator safety is not active in the test modes T1 (Manual Reduced Velocity) and T2 (Manual High Velocity).

⚠ WARNING Following a loss of signal, automatic operation must not be resumed merely by closing the safeguard; it must first additionally be acknowledged. It is the responsibility of the system integrator to ensure this. This is to prevent automatic operation from being resumed inadvertently while there are still persons in the danger zone, e.g. due to the safety gate closing accidentally.

- The acknowledgement must be designed in such a way that an actual check of the danger zone can be carried out first. Acknowledgement functions that do not allow this (e.g. because they are automatically triggered by closure of the safeguard) are not permissible.
- Failure to observe this may result in death to persons, severe physical injuries or considerable damage to property.

Safe operational stop

The safe operational stop can be triggered via an input on the customer interface. The state is maintained as long as the external signal is FALSE. If the external signal is TRUE, the manipulator can be moved again. No acknowledgement is required.

External safety stop 1 and external safety stop 2

Safety stop 1 and safety stop 2 can be triggered via an input on the customer interface. The state is maintained as long as the external signal is FALSE. If the external signal is TRUE, the manipulator can be moved again. No acknowledgement is required.

2 Moving the robot

2.1 Reading and interpreting robot controller messages

Overview of messages

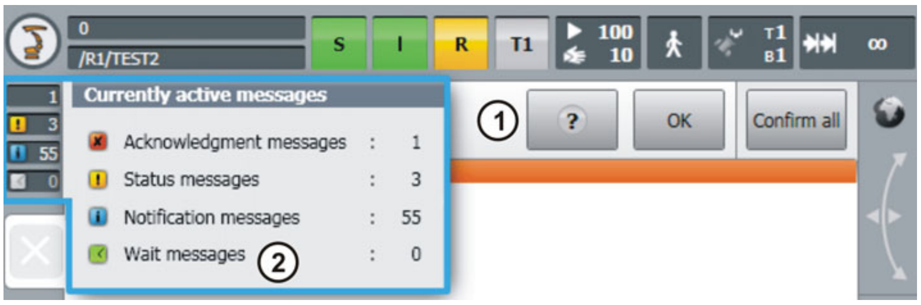


Fig. 2-1: Message window and message counter

1 Message window: the current message is displayed.
2 Message counter: number of messages of each message type.

The controller communicates with the operator via the message window. It has five different message types:

Overview of message types:

Icon	Type
🛑	**Acknowledgement message** ■ Displays states that require confirmation by the operator before program execution is resumed (e.g. “Ackn. EMERGENCY STOP”). ■ An acknowledgement message always causes the robot to stop or not to start.
⚠	**Status message** ■ Status messages signal current controller states (e.g. “EMERGENCY STOP”). ■ Status messages cannot be acknowledged while the status is active.
ℹ	**Notification message** ■ Notification messages provide information for correct operator control of the robot (e.g. “Start key required”). ■ Notification messages can be acknowledged. They do not need to be acknowledged, however, as they do not stop the controller.
🕒	**Wait message** ■ Wait messages indicate the event the controller is waiting for (status, signal or time). ■ Wait messages can be canceled manually by pressing the “Simulate” button.

⚠ WARNING	The command “Simulate” may only be used if there is no risk of a collision or other hazards!

Dialog message

- Dialog messages are used for direct communication with the operator, e.g. to ask the operator for information.
- A message window with buttons appears, offering various possible responses.

An acknowledgeable message can be acknowledged with **OK**. All acknowledgeable messages can be acknowledged at once with **All OK**.

Influence of messages

Messages influence the functionality of the robot. An acknowledgement message always causes the robot to stop or not to start. The message must be acknowledged before the robot can be moved.

The command **OK** (acknowledge) represents a prompt to the operator, forcing a conscious response.

Tips for dealing with messages:

- Read attentively!
- Read older messages first. A newer message could simply be a follow-up to an older one.
- Do not simply press “All OK”.
- Particularly after booting: look through the messages. Display all messages (touching the message window expands the message list).

Dealing with messages

Messages are always displayed with the date and time in order to be able to trace the exact time of the event.

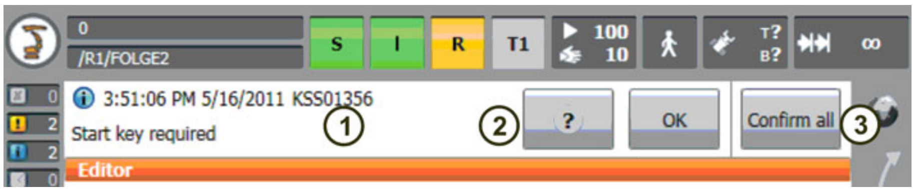

3:51:06 PM 5/16/2011 KSS01356
Start key required

Fig. 2-2: Acknowledging messages

Procedure for viewing and acknowledging messages:

1. Touch the message window (1) to expand the message list.
2. Acknowledge:
   - Acknowledge individual messages with **OK** (2).
   - Alternatively: acknowledge all messages with **All OK** (3).
3. Touching the top message again or an “X” on the left-hand edge of the screen closes the message list.

2.2 Selecting and setting the operating mode

Operating modes of a KUKA robot

- T1 (Manual Reduced Velocity)
  - For test operation, programming and teaching
  - Velocity in program mode max. 250 mm/s
  - Velocity in jog mode max. 250 mm/s
- T2 (Manual High Velocity)
  - For test operation
  - Velocity in program mode corresponds to the programmed velocity!
  - Jog mode: not possible.
- AUT (Automatic)

    - For industrial robots without higher-level controllers
    - Velocity in program mode corresponds to the programmed velocity!
    - Jog mode: not possible.
- AUT EXT (Automatic External)
    - For industrial robots with higher-level controllers (PLC)
    - Velocity in program mode corresponds to the programmed velocity!
    - Jog mode: not possible.

Safety instructions – operating modes

Jog mode T1 and T2
Manual mode is the mode for setup work. Setup work is all the tasks that have to be carried out on the robot system to enable automatic operation. These include:

- Teaching/programming
- Executing a program in jog mode (testing/verification)

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

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

- Operator safety (safety gate) is inactive!
- If it can be avoided, there must be no other persons inside the safeguarded area.

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

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

- Operator safety (safety gate) is inactive!
- This mode may only be used if the application requires a test at a velocity higher than Manual Reduced Velocity.
- Teaching is not permissible in this operating mode.
- Before commencing the test, the operator must ensure that the enabling devices are operational.
- The operator must be positioned outside the danger zone.
- There should be no other persons inside the safeguarded area.

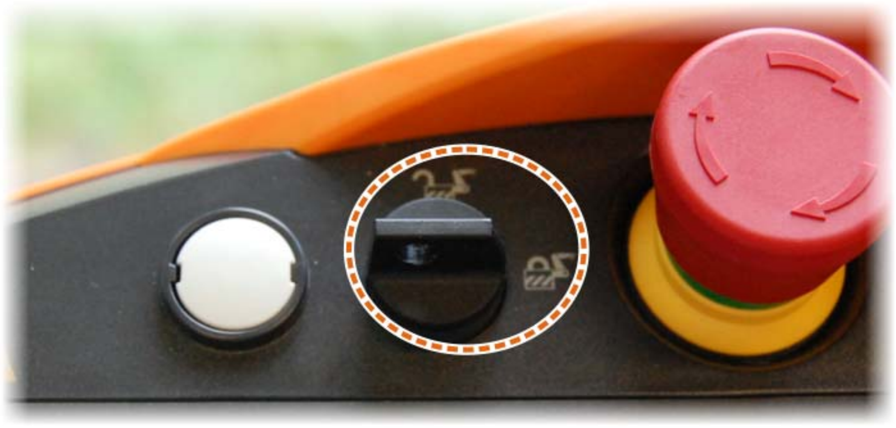
Operating modes **Automatic** and **Automatic External**

- Safety equipment and safeguards must be present and fully operational.
- All persons are outside the safeguarded area.

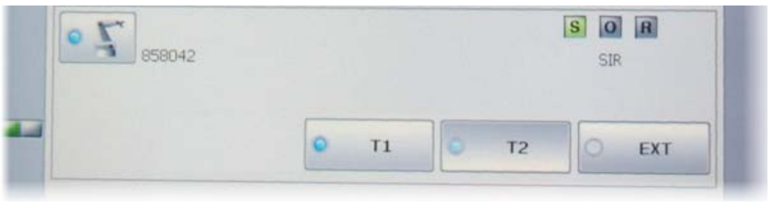
Procedure

If the operating mode is changed during operation, the drives are immediately switched off. The industrial robot stops with a safety stop 2.

1. On the KCP, turn the switch for the connection manager. The connection manager is displayed.

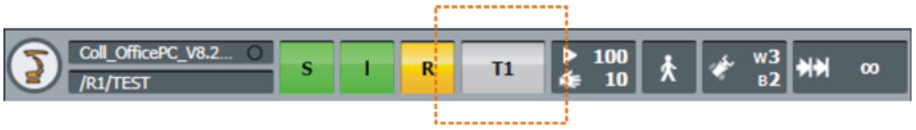


2. Select the operating mode.



3. Return the switch for the connection manager to its original position.

   The selected operating mode is displayed in the status bar of the smartPAD.



2.3 Moving individual robot axes

Description: Axis-specific motion

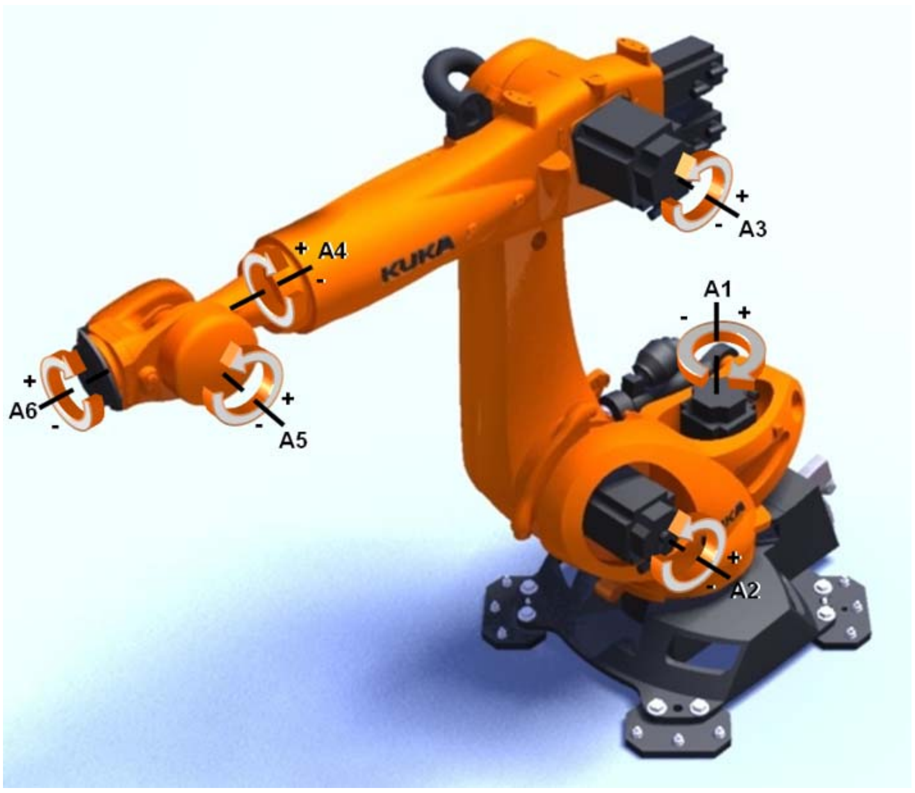


Fig. 2-3: Degrees of freedom of a KUKA robot

Moving robot axes

- Move each axis individually in the plus and minus direction.
- The jog keys or Space Mouse of the KUKA smartPAD are used for this.
- The velocity can be modified (jog override: HOV).
- Jogging is only possible in T1 mode.
- The enabling switch must be pressed.

Principle

The drives are activated by pressing the enabling switch. As soon as a jog key or the Space Mouse is pressed, servo control of the robot axes starts and the desired motion is executed.

Continuous motion and incremental motion are possible. The size of the increment must be selected in the status bar.

The following messages influence manual operation:

Message	Cause	Remedy
"Active commands inhibited"	A (STOP) message or state is present which inhibits active commands. (e.g. EMERGENCY STOP pressed or drives not ready)	Release EMERGENCY STOP and/or acknowledge messages in the message window. As soon as an enabling switch is pressed, the drives are available immediately.
"Software limit switch – A5"	The robot has moved to the software limit switch of the axis indicated (e.g. A5) in the direction indicated (+ or -).	Move the indicated axis in the opposite direction.

Safety instructions relating to axis-specific jogging

Operating mode
Manual operation of the robot is only permissible in T1 mode (Manual Reduced Velocity). The maximum jog velocity in T1 is 250 mm/s. The operating mode is set via the connection manager.

Enabling switches
In order to be able to jog the robot, an enabling switch must be pressed. There are three enabling switches installed on the smartPAD. The enabling switches have three positions:

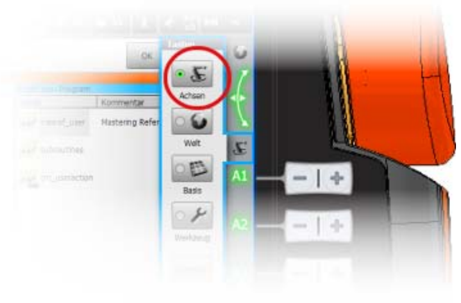
- Not pressed
- Center position
- Panic position

Software limit switches
The motion of the robot is also limited in axis-specific jogging by means of the maximum positive and negative values of the software limit switches.

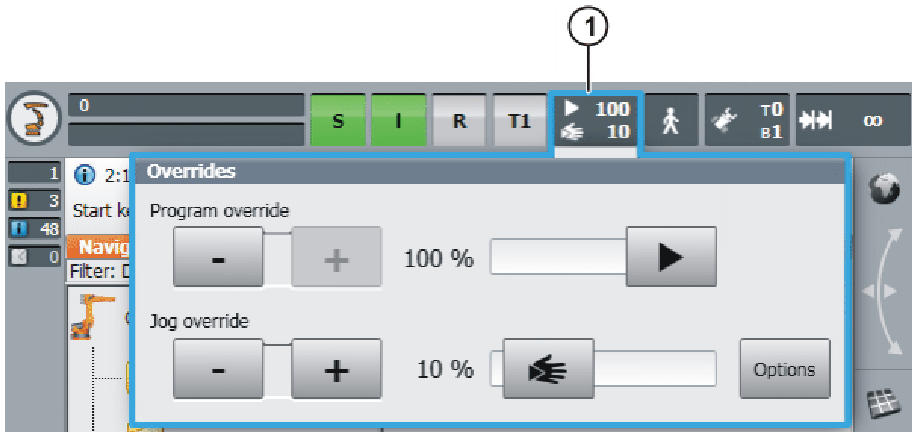
> **⚠ CAUTION** If the message “Perform mastering” appears in the message window, these limits can be exceeded. This can result in damage to the robot system!

Procedure: Executing an axis-specific motion

1. Select **Axis** as the option for the jog keys.



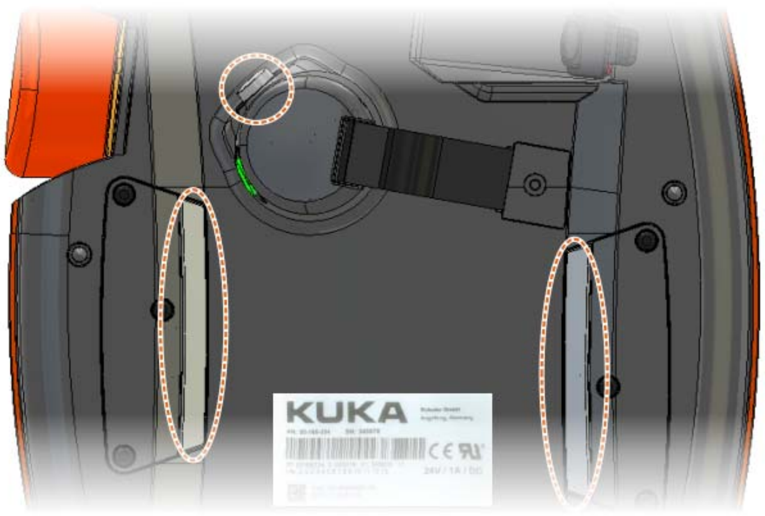
2. Set jog override.



Program override
100 %

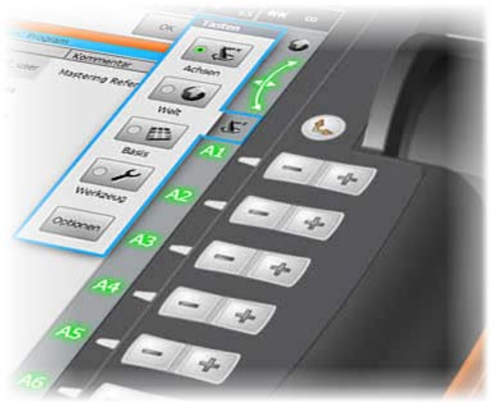
Jog override
10 %

3. Press the enabling switch into the center position and hold it down.



Axes A1 to A6 are displayed next to the jog keys.

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



Moving the robot in emergencies without the controller

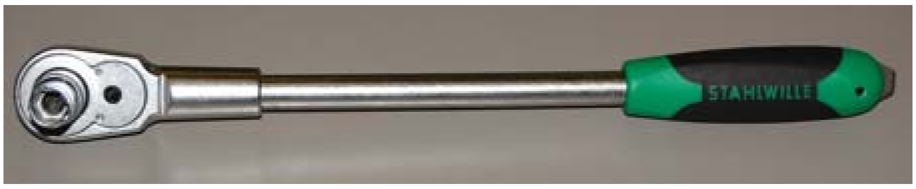


Fig. 2-4: Release device

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.

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

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.

   Labeling of the directions with arrows on the motors can be ordered as an option. It is necessary to overcome the resistance of the mechanical motor brake and any other loads acting on the axis.

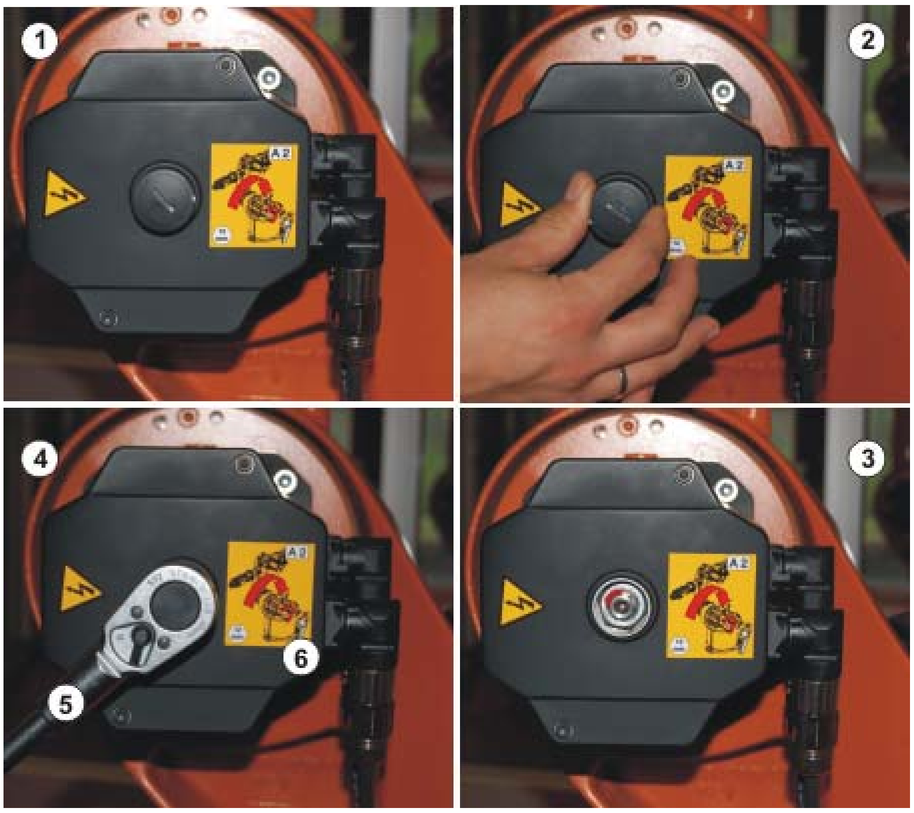


Fig. 2-5: Procedure for using release device

Item	Description
1	Motor A2 with protective cap fitted
2	Removing the protective cap from motor A2
3	Motor A2 with protective cap removed
4	Mounting the release device on motor A2
5	Release device
6	Sign (optional) indicating the direction of rotation

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

Further information is contained in the robot operating instructions.

2.4 Coordinate systems in conjunction with robots

During the operator control, programming and start-up of industrial robots, the coordinate systems are of major significance. The following coordinate systems are defined in the robot controller:

- WORLD | world coordinate system
- ROBROOT | robot base coordinate system
- BASE | base coordinate system
- FLANGE | flange coordinate system
- TOOL | tool coordinate system

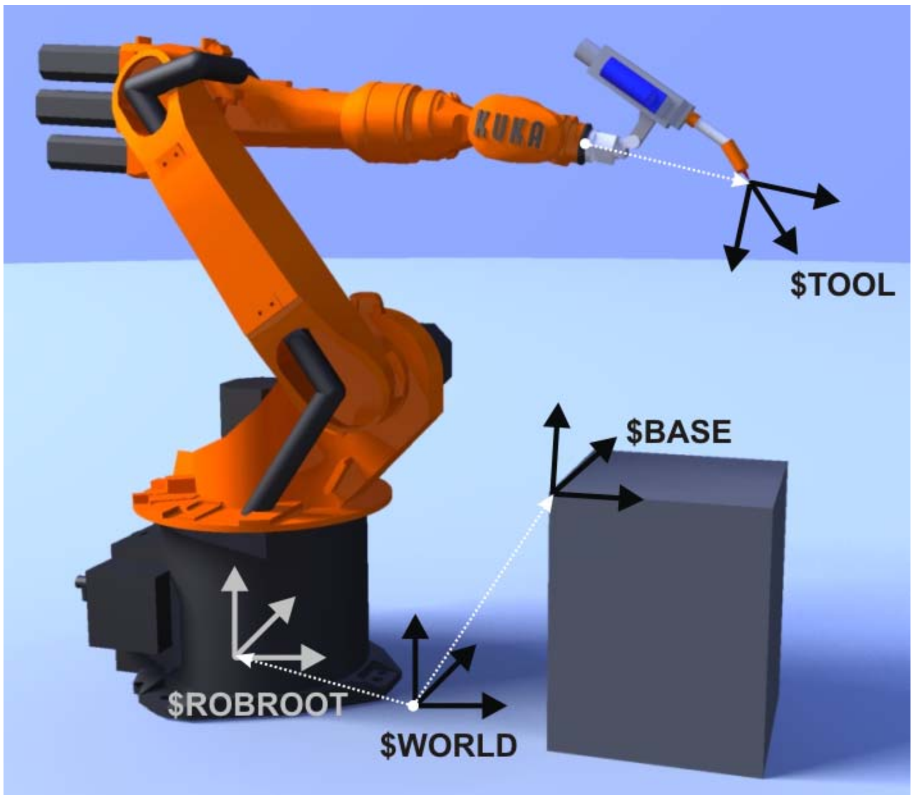


Fig. 2-6: Coordinate systems on the KUKA robot

Name	Location	Use	Special feature:
WORLD	Freely definable	Origin for ROB-ROOT and BASE	Located in the robot base in most cases.
ROB-ROOT	Fixed in the robot base	Origin of the robot	Defines the position of the robot relative to WORLD.
BASE	Freely definable	Tools, fixtures	Defines the position of the base relative to WORLD.
FLANGE	Fixed at the robot flange	Origin for TOOL	Origin is the center of the robot flange.
TOOL	Freely definable	Tools	The origin of the TOOL coordinate system is called the "**TCP**". (*TCP = Tool Center Point*)

2.5 Moving the robot in the world coordinate system

Motion in the world coordinate system

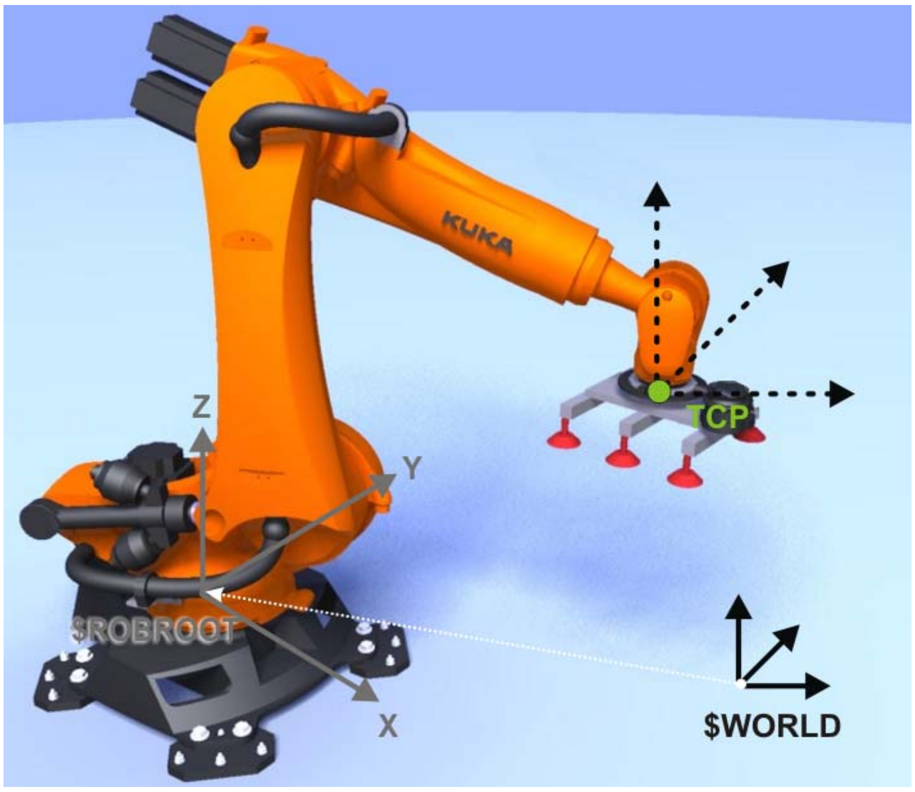


Fig. 2-7: Principle of jogging in the world coordinate system

- The robot tool can be moved with reference to the coordinate axes of the world coordinate system.
  In this case, **all** robot axes move.
- The jog keys or Space Mouse of the KUKA smartPAD are used for this.
- By default, the world coordinate system is located in the base of the robot (Robroot).
- The velocity can be modified (jog override: HOV).
- Jogging is only possible in T1 mode.
- The enabling switch must be pressed.

Space Mouse

- The Space Mouse allows intuitive motion of the robot and is the ideal choice for jogging in the world coordinate system.
- The mouse position and degrees of freedom can be modified.

Principle of jogging in the world coordinate system

A robot can be moved in a coordinate system in two different ways:

- Translational (in a straight line) along the orientation directions of the coordinate system: X, Y, Z
- Rotational (turning/pivoting) about the orientation directions of the coordinate system: angles A, B and C

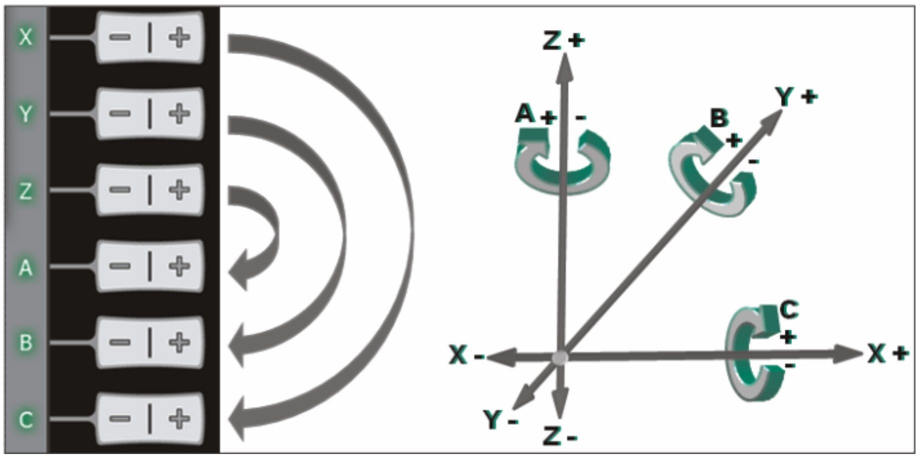


Fig. 2-8: Cartesian coordinate system

In the case of a motion command (e.g. jog key pressed), the controller first calculates a path. The starting point of the path is the tool center point (TCP). The direction of the path is specified by the world coordinate system. The controller then controls the axes to guide the tool along this path (translation) or about it (rotation).

Advantages of using the world coordinate system:

- The motion of the robot is always predictable.
- The motions are always unambiguous, as the origin and coordinate axes are always known.
- The world coordinate system can always be used with a mastered robot.
- The Space Mouse allows intuitive operator control.

Using the Space Mouse

- All motion types are possible with the Space Mouse:
- Translational: by pushing and pulling the Space Mouse

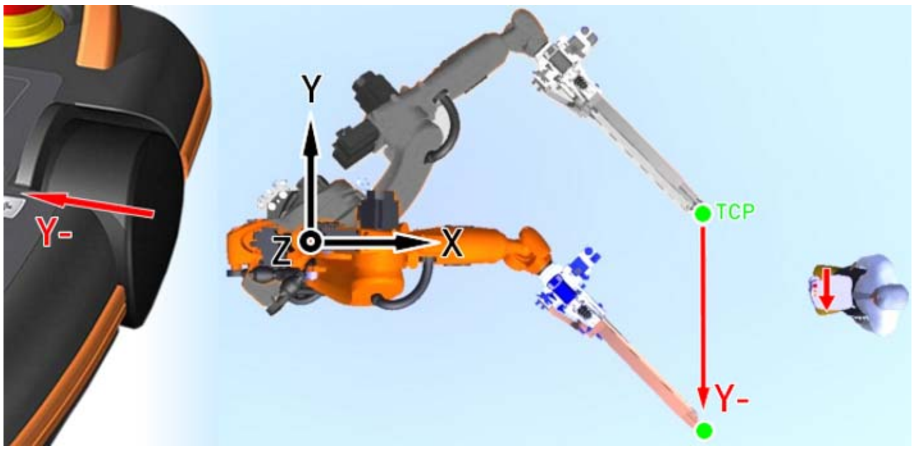


Fig. 2-9: Example: motion to the left

- Rotational: by turning the Space Mouse

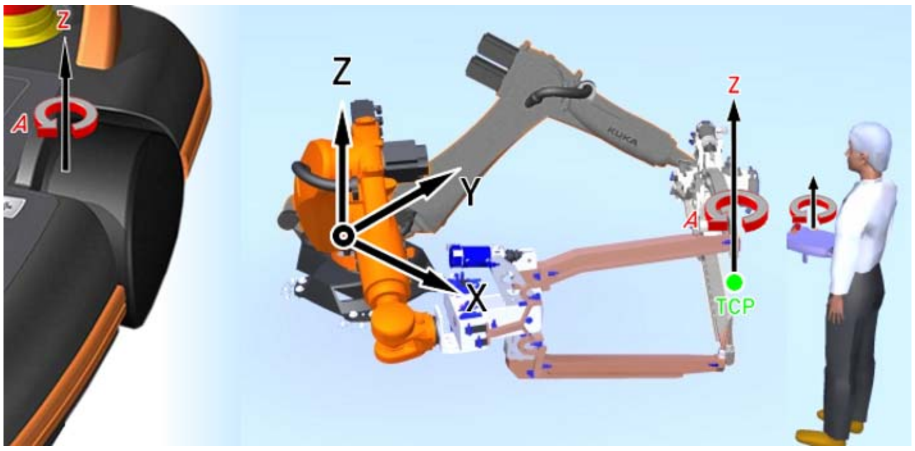


Fig. 2-10: Example: rotational motion about Z – angle A

- The Space Mouse position can be adapted to the position of the operator relative to the robot.

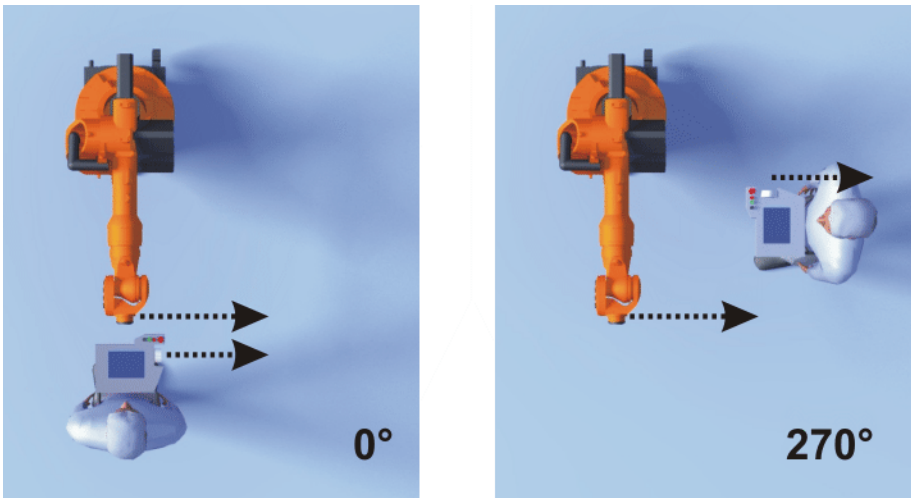
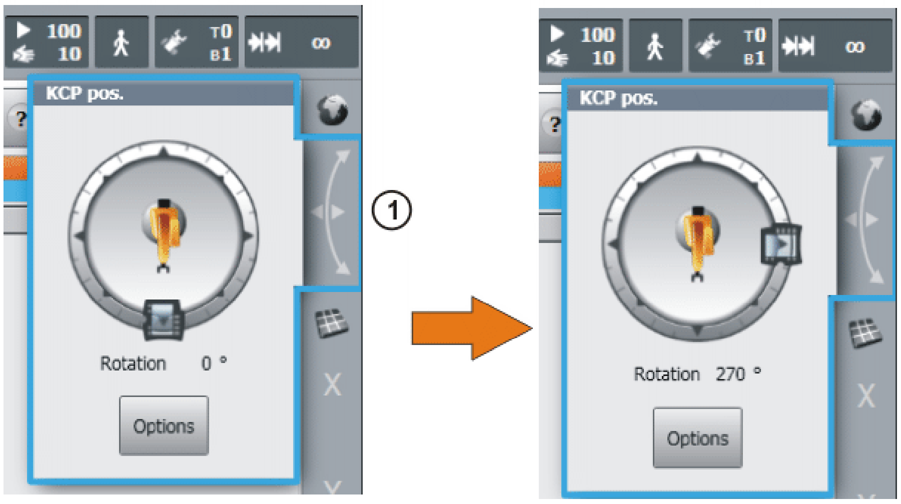


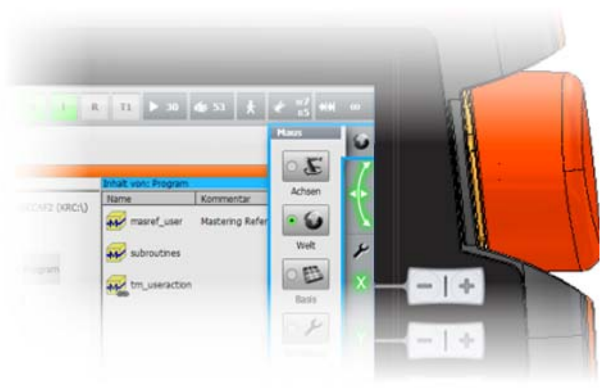
Fig. 2-11: Space Mouse: 0° and 270°

Executing a translational motion (world)

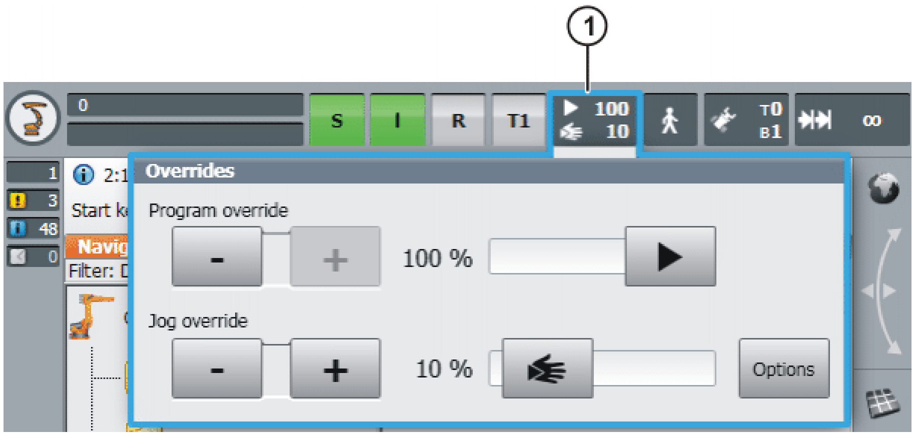
1. Set the KCP position by moving the slider control (1).



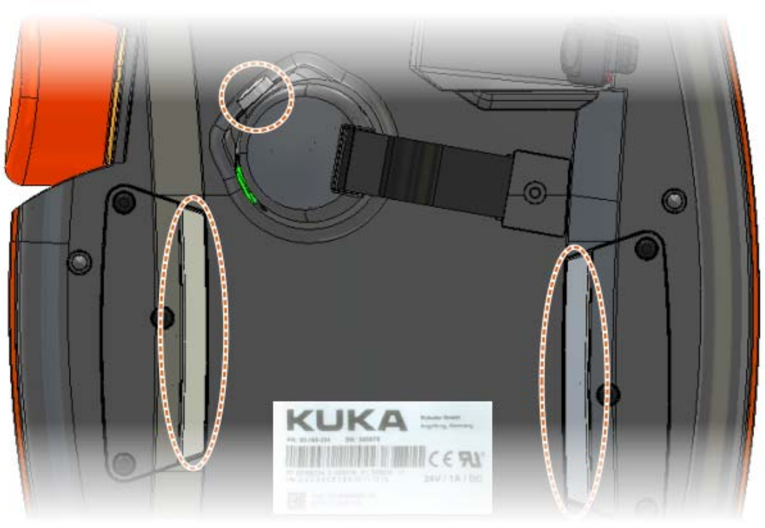
2. Select **World** as the option for the Space Mouse.



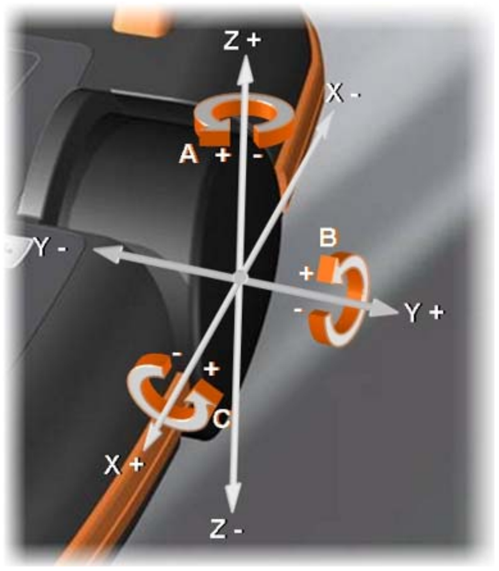
3. Set jog override.



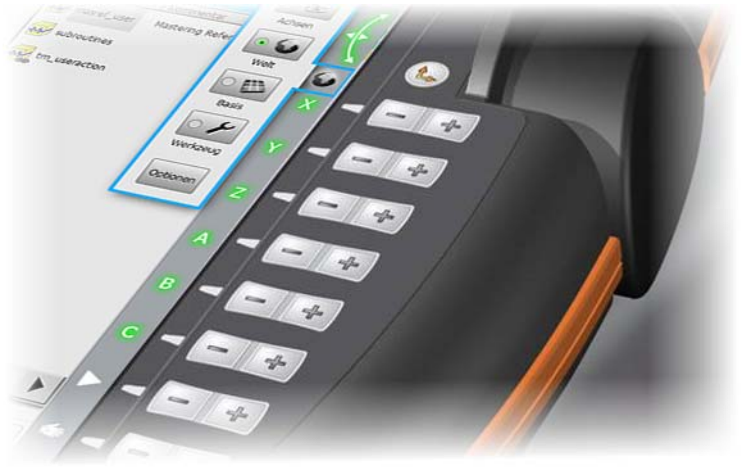
4. Press the enabling switch into the center position and hold it down.



5. Move in the corresponding direction using the Space Mouse.



6. Alternatively, the jog keys can be used:



2.6 Moving the robot in the tool coordinate system

Jogging in the tool coordinate system

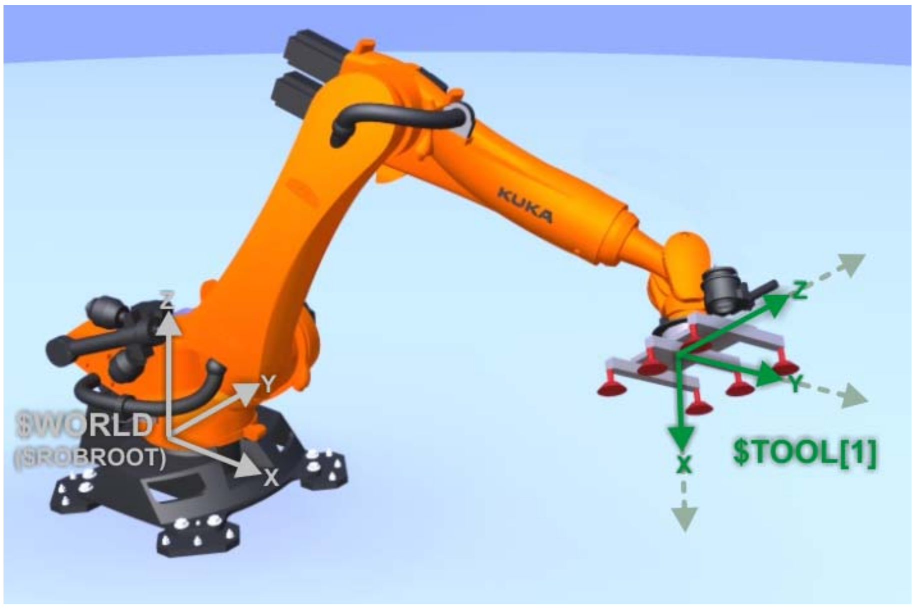


Fig. 2-12: Robot tool coordinate system

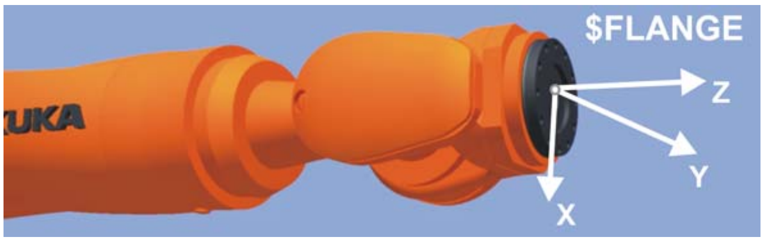
- In the case of jogging in the tool coordinate system, the robot can be moved relative to the coordinate axes of a previously calibrated tool.

  The coordinate system is thus not fixed (cf. world/base coordinate system), but guided by the robot.

  In this case, **all** required robot axes move. Which axes these are is determined by the system and depends on the motion.

  The origin of the tool coordinate system is called the **TCP** and corresponds to the working point of the tool.
- The jog keys or Space Mouse of the KUKA smartPAD are used for this.
- There are 16 tool coordinate systems to choose from.
- The velocity can be modified (jog override: HOV).
- Jogging is only possible in T1 mode.
- The enabling switch must be pressed.

In the case of jogging, uncalibrated tool coordinate systems always correspond to the flange coordinate system.



Principle of jogging – tool

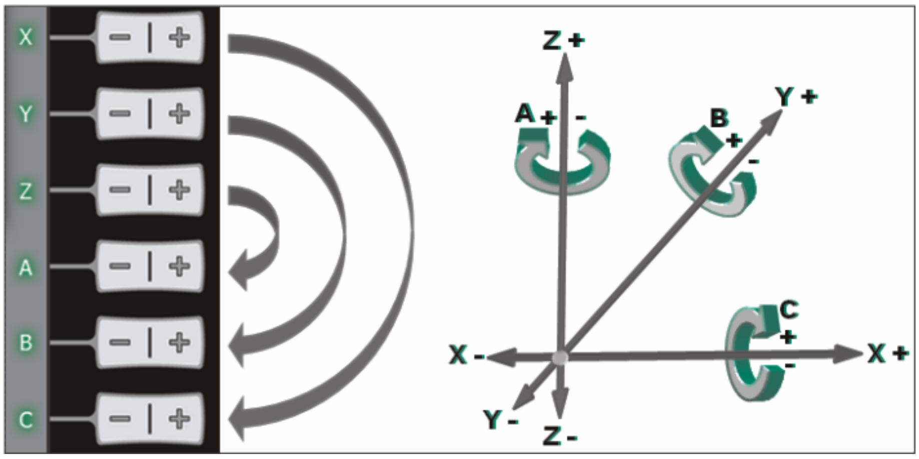


Fig. 2-13: Cartesian coordinate system

A robot can be moved in a coordinate system in two different ways:

- Translational (in a straight line) along the orientation directions of the coordinate system: X, Y, Z
- Rotational (turning/pivoting) about the orientation directions of the coordinate system: angles A, B and C

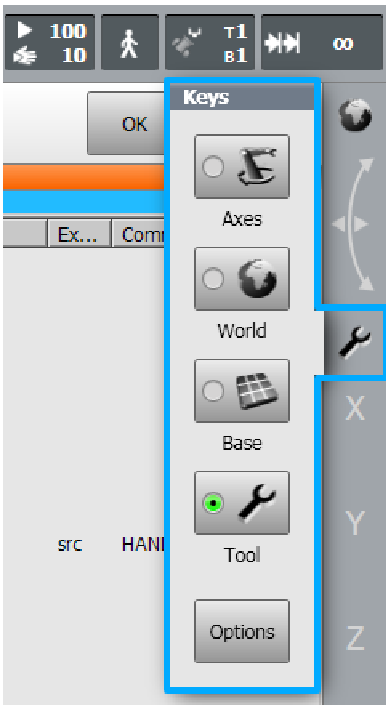
Advantages of using the tool coordinate system:

- The motion of the robot is always predictable as soon as the tool coordinate system is known.
- It is possible to move in the tool direction or to orient about the TCP.

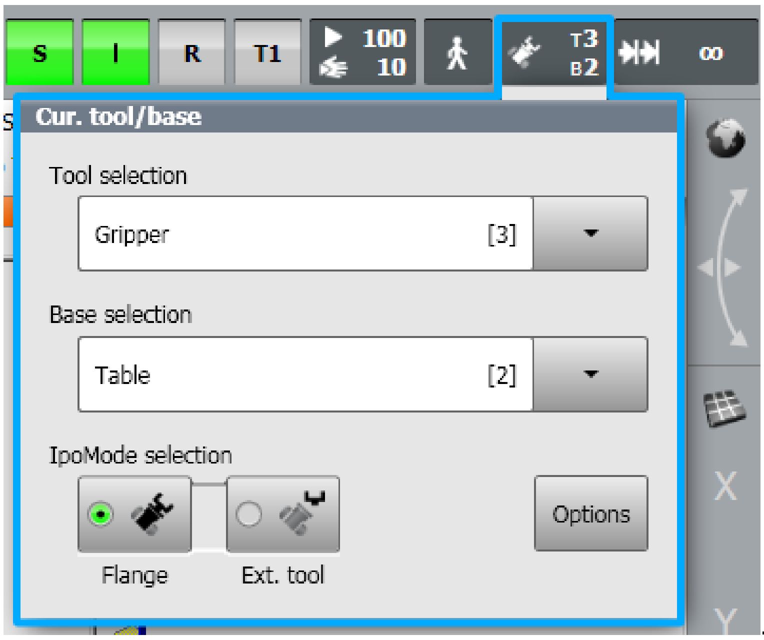
  The *tool direction* is the working or process direction of the tool: the direction in which adhesive is dispensed from an adhesive nozzle, the direction of gripping when gripping a workpiece, etc.

Procedure

1. Select **Tool** as the coordinate system to be used.



2. Select the tool number.

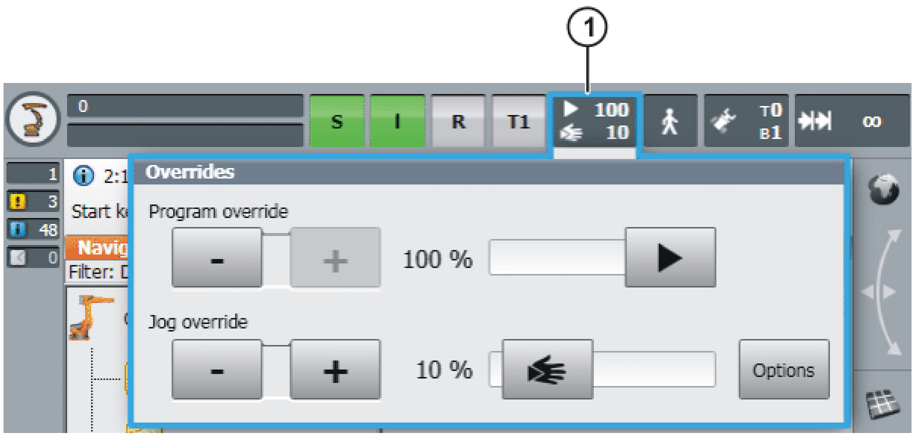


Tool selection: Gripper [3]

Base selection: Table [2]

IpoMode selection: Flange

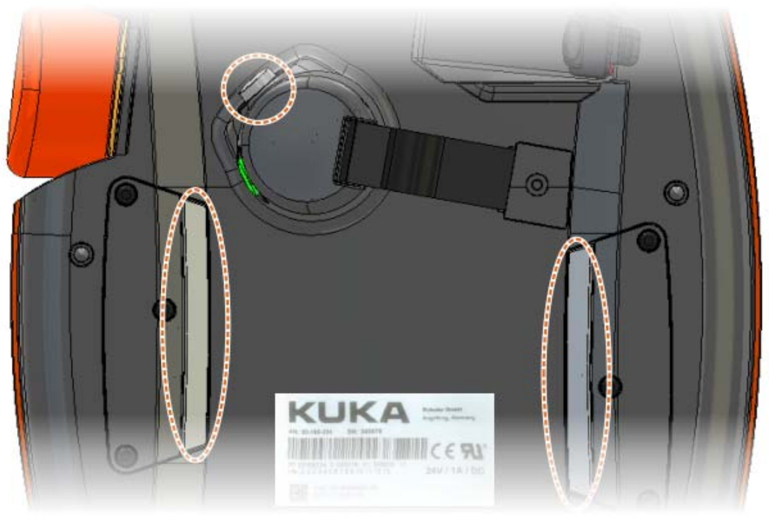
3. Set jog override.



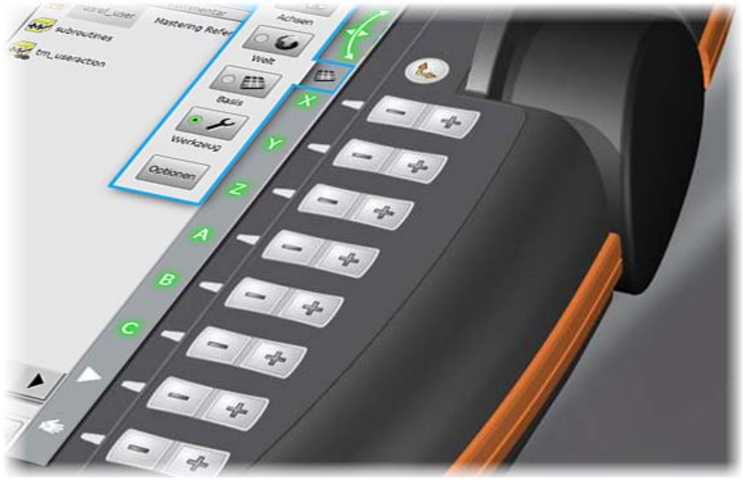
Program override: 100 %

Jog override: 10 %

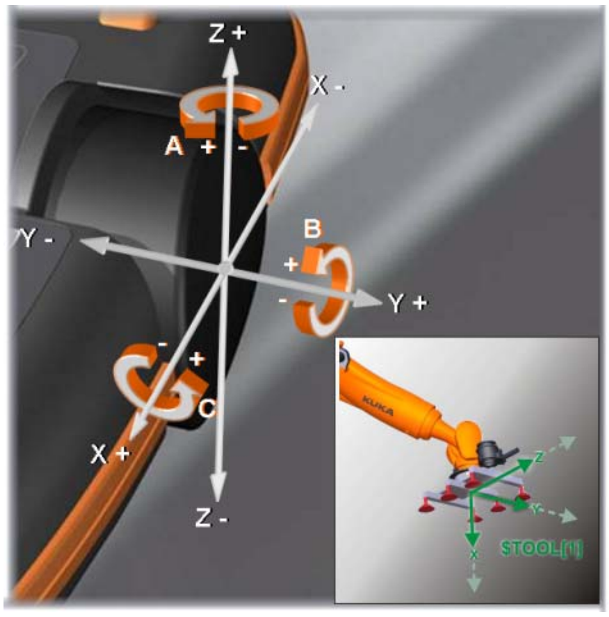
4. Press the enabling switch into the center position and hold it down.



5. Move the robot using the jog keys.



6. Alternatively: Move in the corresponding direction using the Space Mouse.



2.7 Moving the robot in the base coordinate system

Motion in the base coordinate system

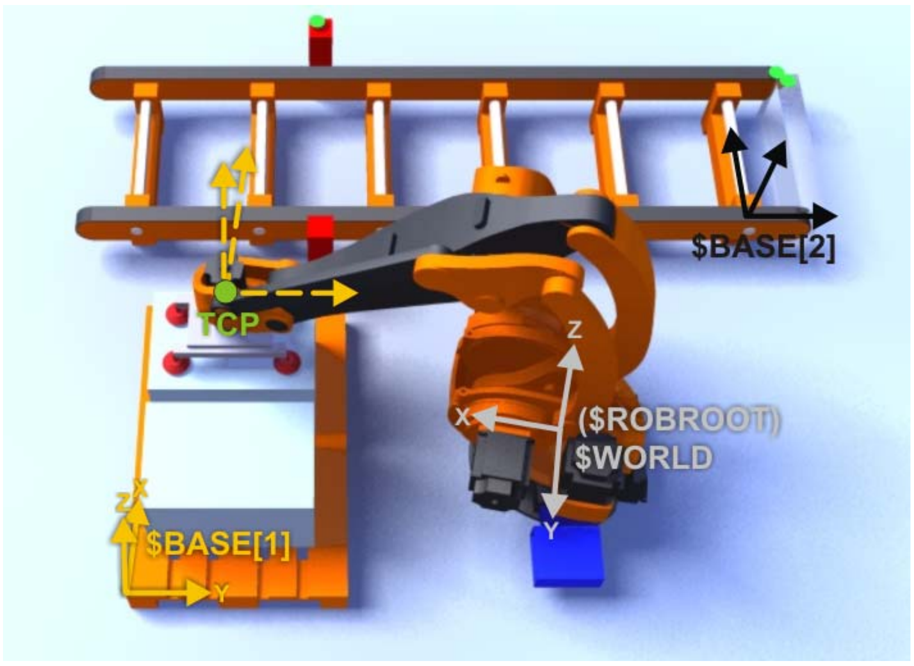


Fig. 2-14: Jogging in the base coordinate system

Description of bases

- The robot tool can be moved with reference to the coordinate axes of the base coordinate system. Base coordinate systems can be calibrated individually and are often oriented along the edges of workpieces, workpiece locations or pallets. This allows convenient jogging!

  In this case, **all** required robot axes move. Which axes these are is determined by the system and depends on the motion.
- The jog keys or Space Mouse of the KUKA smartPAD are used for this.
- There are 32 base coordinate systems to choose from.
- The velocity can be modified (jog override: HOV).
- Jogging is only possible in T1 mode.
- The enabling switch must be pressed.

Principle of jogging – base

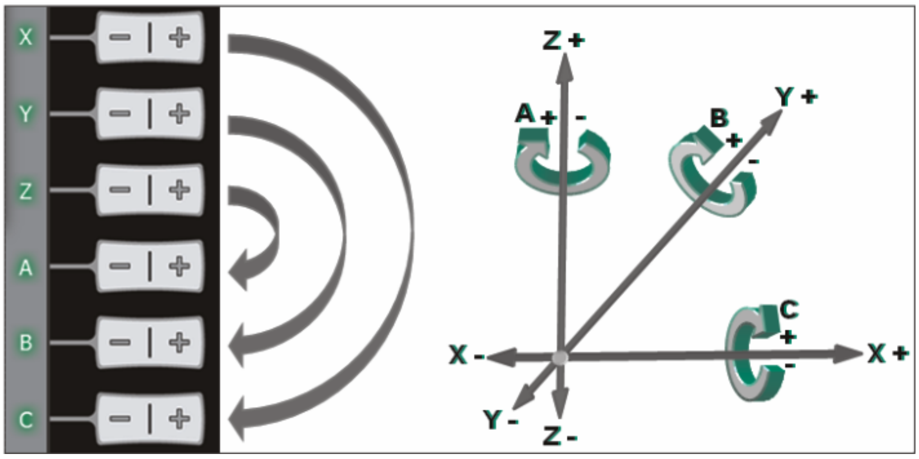


Fig. 2-15: Cartesian coordinate system

A robot can be moved in a coordinate system in two different ways:

- Translational (in a straight line) along the orientation directions of the coordinate system: X, Y, Z

- Rotational (turning/pivoting) about the orientation directions of the coordinate system: angles A, B and C

In the case of a motion command (e.g. jog key pressed), the controller first calculates a path. The starting point of the path is the tool center point (TCP). The direction of the path is specified by the world coordinate system. The controller then controls the axes to guide the tool along this path (translation) or about it (rotation).

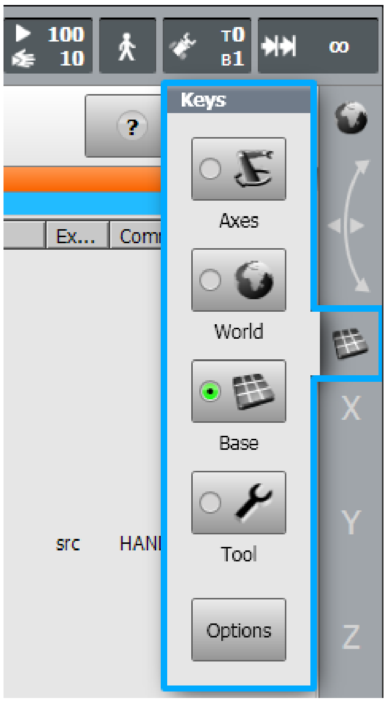
Advantages of using the base coordinate system:

- The motion of the robot is always predictable as soon as the base coordinate system is known.
- Here also, the Space Mouse allows intuitive operator control. A precondition is that the operator is standing correctly relative to the robot or the base coordinate system.

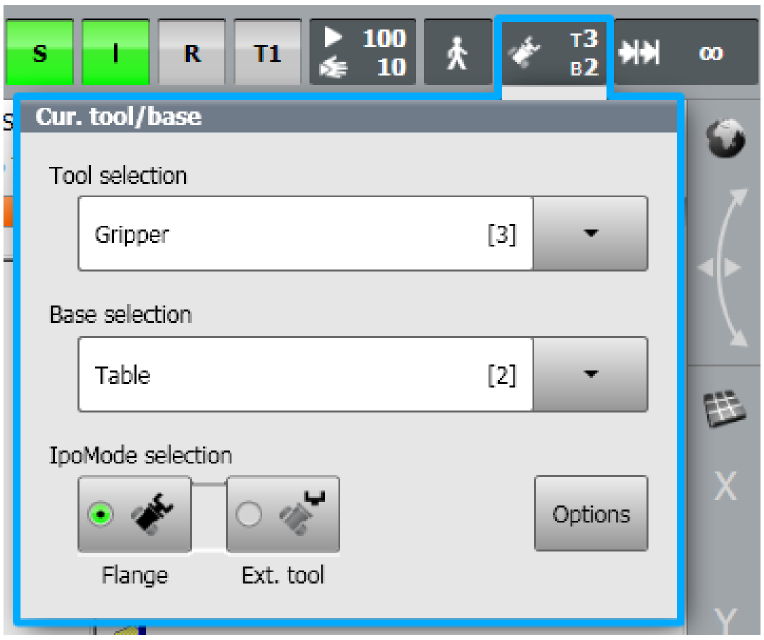
NOTICE If the correct tool coordinate system is also set, re-orientation about the TCP is possible in the base coordinate system.

Procedure

1. Select **Base** as the option for the jog keys.



2. Select tool and base.



Cur. tool/base

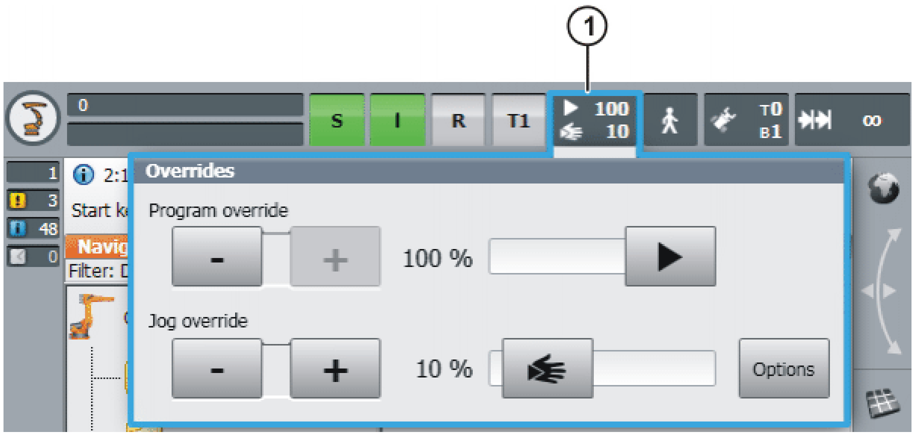
Tool selection: Gripper [3]

Base selection: Table [2]

IpoMode selection: [✓ Flange] Ext. tool

Options

3. Set jog override.



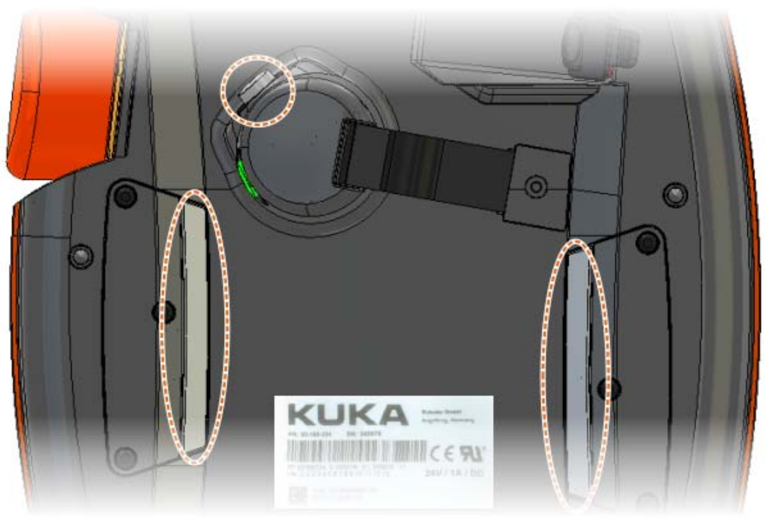
Overrides

Program override - + 100 %

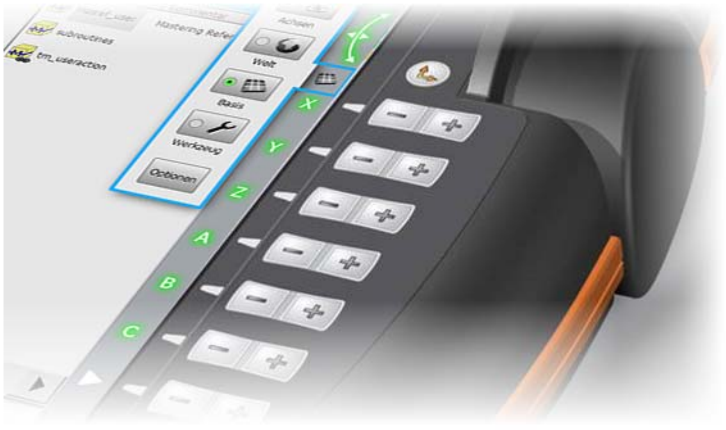
Jog override - + 10 %

Options

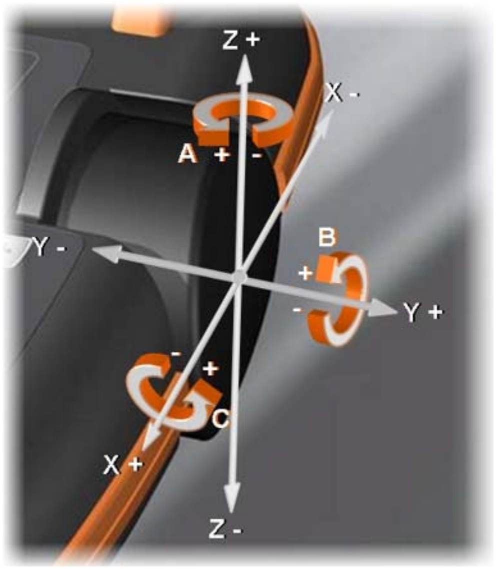
4. Press the enabling switch into the center position and hold it down.



5. Move in the desired direction using the jog keys.



6. Alternatively, jogging can be carried out using the Space Mouse.



Stop reactions

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

Term	Description
Safe operational stop	The safe operational stop is a standstill monitoring function. It does not stop the robot motion, but monitors whether the robot axes are stationary. If these are moved during the safe operational stop, a safety stop STOP 0 is triggered. The safe operational stop can also be triggered externally. When a safe operational stop is triggered, the robot controller sets an output to the field bus. The output is set even if not all the axes were stationary at the time of triggering, thereby causing a safety stop STOP 0 to be triggered.
Safety STOP 0	A stop that is triggered and executed by the safety controller. The safety controller immediately switches off the drives and the power supply to the brakes. **Note:** This stop is called safety STOP 0 in this document.

Term	Description
Safety STOP 1	A stop that is triggered and monitored by the safety controller. The braking process is performed by the non-safety-oriented part of the robot controller and monitored by the safety controller. As soon as the manipulator is at a standstill, the safety controller switches off the drives and the power supply to the brakes. When a safety STOP 1 is triggered, the robot controller sets an output to the field bus. The safety STOP 1 can also be triggered externally. **Note:** This stop is called safety STOP 1 in this document.
Safety STOP 2	A stop that is triggered and monitored by the safety controller. The braking process is performed by the non-safety-oriented part of the robot controller and monitored by the safety controller. The drives remain activated and the brakes released. As soon as the manipulator is at a standstill, a safe operational stop is triggered. When a safety STOP 2 is triggered, the robot controller sets an output to the field bus. The safety STOP 2 can also be triggered externally. **Note:** This stop is called safety STOP 2 in this document.
Stop category 0	The drives are deactivated immediately and the brakes are applied. The manipulator and any external axes (optional) perform path-oriented braking. **Note:** This stop category is called STOP 0 in this document.
Stop category 1	The manipulator and any external axes (optional) perform path-maintaining braking. The drives are deactivated after 1 s and the brakes are applied. **Note:** This stop category is called STOP 1 in this document.
Stop category 2	The drives are not deactivated and the brakes are not applied. The manipulator and any external axes (optional) are braked with a path-maintaining braking ramp. **Note:** This stop category is called STOP 2 in this document.

Trigger	T1, T2	AUT, AUT EXT
Start key released	STOP 2	-
STOP key pressed	STOP 2	
Drives OFF	STOP 1	
“Motion enable” input drops out	STOP 2	
Robot controller switched off (power failure)	STOP 0	
Internal error in non-safety-oriented part of the robot controller	STOP 0 or STOP 1 (dependent on the cause of the error)	
Operating mode changed during operation	Safety stop 2	
Safety gate opened (operator safety)	-	Safety stop 1
Releasing the enabling switch	Safety stop 2	-
Enabling switch pressed fully down or error	Safety stop 1	-
E-STOP pressed	Safety stop 1	
Error in safety controller or periphery of the safety controller	Safety stop 0	

2.8 Exercise: Operator control and jogging

Aim of the exercise

On successful completion of this exercise, you will be able to carry out the following activities:

- Switch the robot controller on and off
- Basic operator control of the robot using the KCP
- Jog the robot (axis-specific and in the WORLD coordinate system) by means of the jog keys and Space Mouse
- Interpret and reset first simple system messages

Preconditions

The following are preconditions for successful completion of this exercise:

- Completion of safety instruction

Note!
Safety instruction must be completed and documented before commencing this exercise!

- Theoretical knowledge of the general operator control of a KUKA industrial robot system
- Theoretical knowledge of axis-specific jogging and jogging in the WORLD coordinate system

Task description

Carry out the following tasks:

1. Switch the control cabinet on and wait for the system to boot.
2. Release and acknowledge the EMERGENCY STOP.
3. Ensure that T1 mode is set.
4. Activate axis-specific jogging.
5. Perform axis-specific jogging of the robot with various different jog override (HOV) settings using the jog keys and Space Mouse.

6. Explore the motion range of the individual axes, being careful to avoid any obstacles present, such as a table or cube magazine with fixed tool (accessibility investigation).
7. On reaching the software limit switches, observe the message window.
8. In joint (axis-specific) mode, move the tool (gripper) to the reference tool (black metal tip) from several different directions.
9. Repeat this procedure in the World coordinate system.

Questions on the exercise

1. How can messages be acknowledged?

. . . .

. . . .

2. Which icon represents the world coordinate system?

a)	b)	c)	d)

3. What is the name of the velocity setting for jog mode?

. . . .

4. What operating modes are there?

. . . .

. . . .

2.9 Jogging with a fixed tool

Advantages and areas of application

Some production and machining processes require the robot to handle the **workpiece** and not the **tool**. The advantage is that it is not necessary to set the workpiece down first before it can be machined – thus saving on clamping fixtures. This is the case, for example, for:

- Adhesive bonding applications
- Welding applications
- etc.

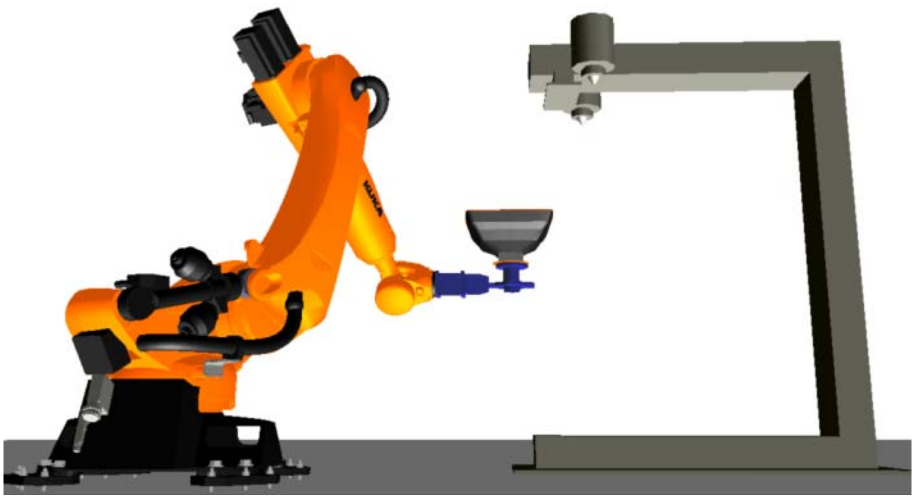


Fig. 2-16: Example of a fixed tool

NOTICE In order to program such an application successfully, both the external TCP of the fixed tool and the workpiece must be calibrated.

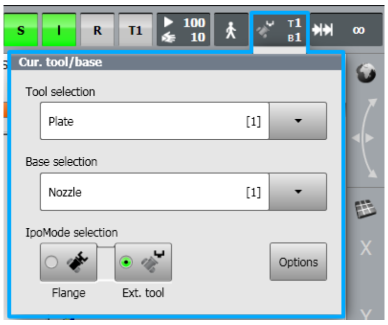
Modified motion sequence with fixed tool

Although the tool is a fixed (non-mobile) object, it nonetheless has a tool reference point with an associated coordinate system. In this case, the reference point is called the **external TCP**. Since it is a non-mobile coordinate system, the data are managed in the same way as a base coordinate system and correspondingly saved as **Base**!

The (mobile) **workpiece**, on the other hand, is saved as **Tool**. This means that motion along the workpiece edges relative to the TCP is possible!

NOTICE It must be taken into consideration that the motions during jogging with a fixed tool are relative to the external TCP!

Procedure for jogging with a fixed tool



Cur. tool/base

Tool selection: Plate [1]

Base selection: Nozzle [1]

IpoMode selection: Flange [✓ Ext. tool]

Fig. 2-17: Selecting the external TCP in the Options menu

1. Select the robot-guided workpiece in the tool selection window.

2. Select the fixed tool in the base selection window.
3. Set IpoMode selection to “External tool”.
4. Set Tool as the option for the jog keys/Space Mouse:
   - Set tool in order to be able to jog in the coordinate system of the workpiece.
   - Set base in order to be able to jog in the coordinate system of the external tool.
5. Set jog override
6. Press the enabling switch into the center position and hold it down.
7. Move in the desired direction using the jog keys/Space Mouse.

Selecting **Ext. tool** in the option window **Jog options** switches the controller: all motions are now carried out relative to the external TCP and not to a robot-guided tool.

2.10 Exercise: Jogging with a fixed tool

Aim of the exercise

On successful completion of this exercise, you will be able to carry out the following activities:

- Jog a robot guiding a workpiece relative to a fixed tool

Preconditions

The following are preconditions for successful completion of this exercise:

- Theoretical knowledge of the general operator control of a KUKA industrial robot system
- Theoretical knowledge of jogging with an external tool

Task description

1. Set the tool coordinate system “Panel”.
2. Set the base coordinate system “External pen”.
3. In the option window “Jog options”, set “Ext. tool”.
4. Move the panel to the external pen.
5. Move and orient the panel at the external pen. Test the differences between Tool and Base.
6. In the option window “Jog options”, set “Flange”.
7. Move and orient the panel at the external pen.

3 Starting up the robot

3.1 Mastering principle

Why is mastering carried out?

An industrial robot can only be used optimally if it is also completely and correctly mastered. Only then can it exploit its pose accuracy and path accuracy to the full, or be moved using programmed motions at all.

During mastering, a reference value is assigned to every axis.

A complete mastering operation includes the mastering of every single axis. With the aid of a technical tool (EMD = Electronic Mastering Device), a reference value (e.g. 0°) is assigned to every axis in its **mechanical zero position**. Since, in this way, the mechanical and electrical positions of the axis are matched, every axis receives an unambiguous angle value.

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

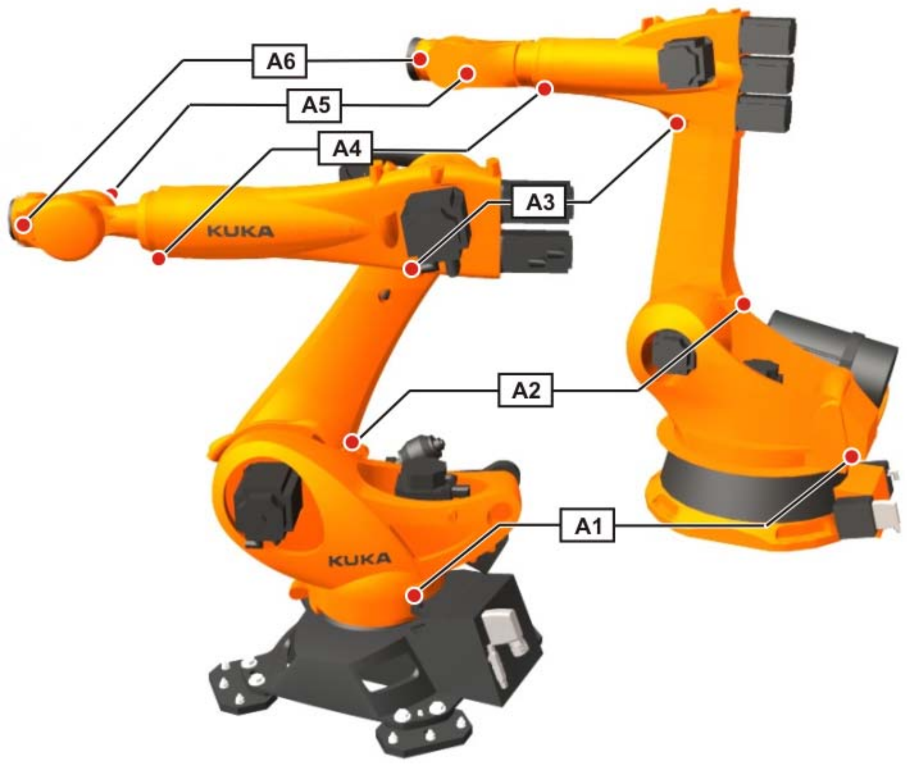


Fig. 3-1: Positions of the mastering cartridges

Angle values of the mechanical zero position (= reference values)

Axis	"Quantec" robot generation	Other robot types (e.g. Series 2000, KR 16, etc.)
A1	-20°	0°
A2	-120°	-90°
A3	+120°	+90°
A4	0°	0°
A5	0°	0°
A6	0°	0°

When is mastering carried out?

A robot must always be mastered. Mastering must be carried out in the following cases:

- During commissioning
- Following maintenance work to components that are involved in the acquisition of position values (e.g. motor with resolver or RDC)
- If robot axes are moved without the controller (e.g. by means of a release device)
- Following mechanical repairs/problems, the robot must first be unmastered before mastering can be carried out:
  - After exchanging a gear unit.
  - After an impact with an end stop at more than 250 mm/s
  - After a collision.

NOTICE Before carrying out maintenance work, it is generally a good idea to check the current mastering.

Safety instructions for mastering

The functionality of the robot is severely restricted if robot axes are not mastered:

- Program mode is not possible: programmed points cannot be executed.
- No translational jogging: motions in the coordinate systems are not possible.
- Software limit switches are deactivated.

⚠ CAUTION **Warning!**
The software limit switches of an unmastered robot are deactivated. The robot can hit the end stop buffers, thus damaging them and making it necessary to exchange them. 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.

Performing mastering

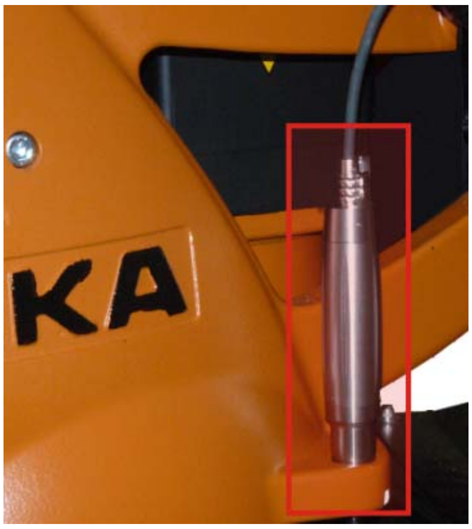


Fig. 3-2: EMD in operation

Mastering is carried out by determining the mechanical zero point of the axis. The axis is moved until the mechanical zero point is reached. This is the case when the gauge pin has reached the lowest point in the reference notch. Every axis is thus equipped with a mastering cartridge and a mastering mark.

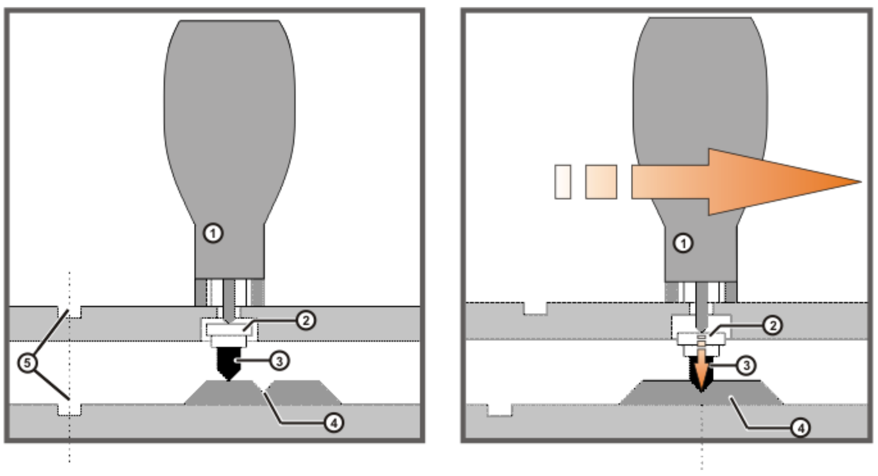


Fig. 3-3: EMD mastering sequence

1	EMD (Electronic Mastering Device)	4	Reference notch
2	Gauge cartridge	5	Premastering mark
3	Gauge pin		

3.2 Mastering the robot

Robot mastering options

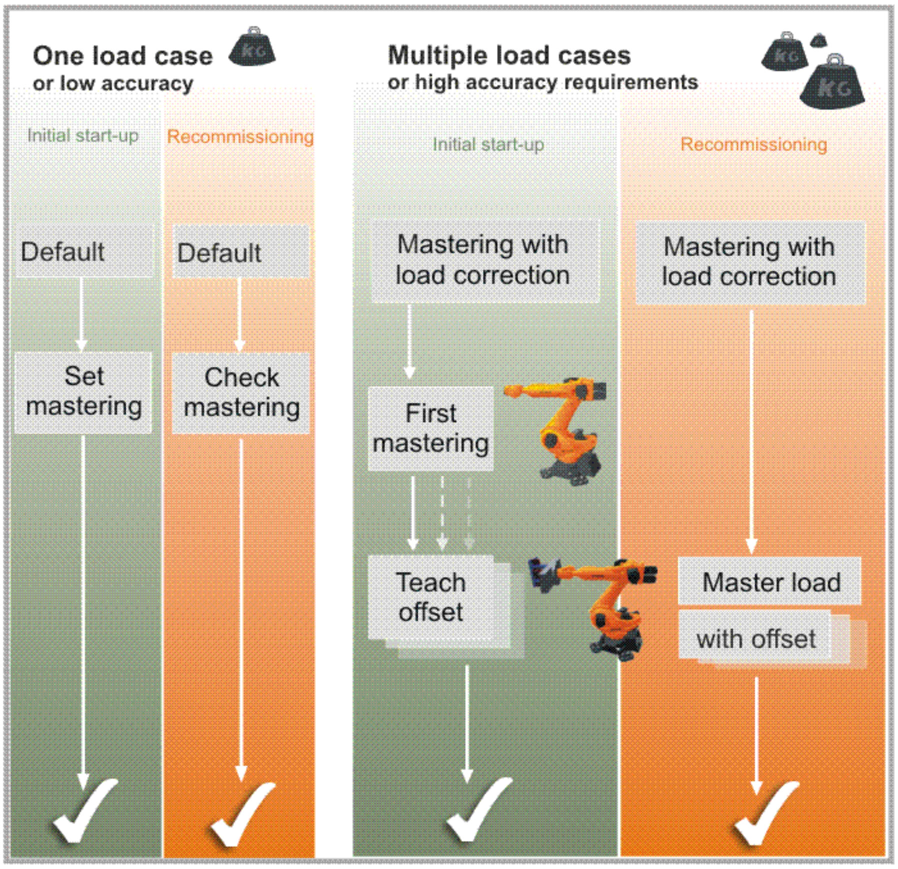


Fig. 3-4: Mastering options

Why teach the offset?

Due to the weight of the tool mounted on the flange, the robot is subjected to a static load. Material-related elasticity of the components and gear units can result in the robot positions being different for a loaded robot and an unloaded robot. These differences of just a few increments affect the accuracy of the robot.

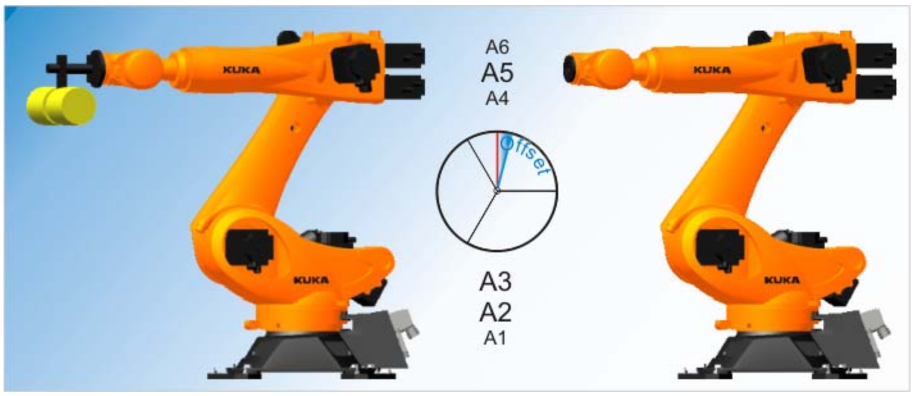


Fig. 3-5: Teach offset

“Teach offset” is carried out with a load. The difference from the first mastering (without a load) is saved.

If the robot is operated with different loads, the function “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.

NOTICE **Mastery.logMastery.logMastering offset values file**

The calculated offsets are saved in the file **Mastery.log**. The file is located in the directory C:\KRC\ROBOTER\LOG on the hard drive and contains the specific mastering data:

- Time stamp (date, time)
- Axis
- Serial number of the robot
- Tool number
- Offset value (*Encoder Difference*) in degrees
- **Sample Mastery.log:**

```
Date: 22.03.11  Time: 10:07:10
Axis 1  Serialno.: 863334  Tool Teaching for Tool No 5
(Encoder Difference: -0.001209)

Date: 22.03.11  Time: 10:08:44
Axis 2  Serialno.: 863334  Tool Teaching for Tool No 5
Encoder Difference: 0.005954)

...
```

Only a robot mastered with load correction has the required accuracy. For this reason, an offset must be taught for every load case! A precondition is that the geometric calibration of the tool has already been carried out and that the tool has thus been assigned a tool number.

Procedure for first mastering

NOTICE First mastering may only be carried out if the robot is without a load. There must be no tool or supplementary load mounted.

1. Move robot to the pre-mastering position.

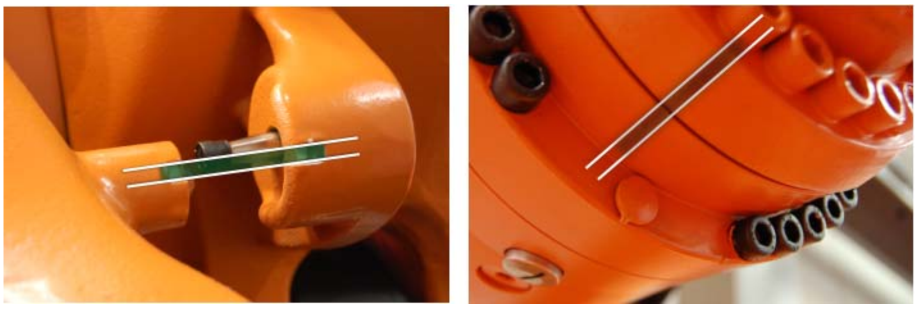


Fig. 3-6: Examples of the pre-mastering position

2. Select **Start-up** > **Master** > **EMD** > **With load correction** > **First mastering** in the main menu.

   A window opens. All axes to be mastered are displayed. The axis with the lowest number is highlighted.

3. Remove the protective cap of the gauge cartridge on the axis highlighted in the window. (Turned around, the EMD can be used as a screwdriver.) Screw the EMD onto the gauge cartridge.

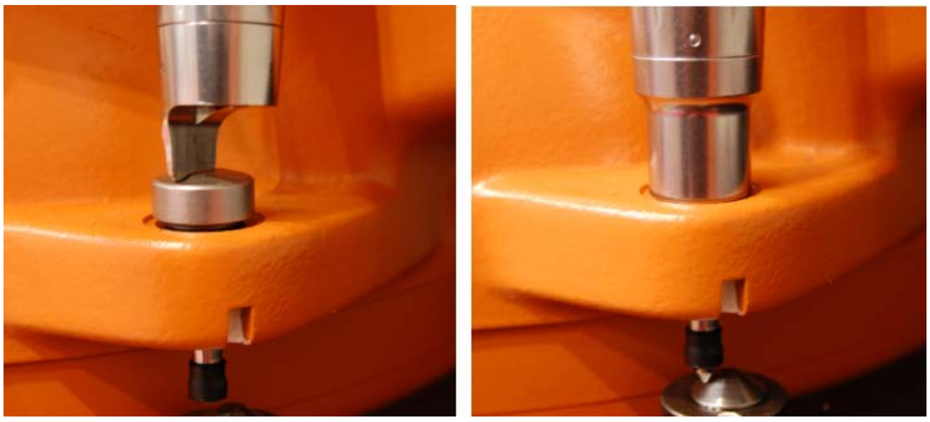


Fig. 3-7: EMD screwed onto gauge cartridge

   Then attach the signal cable to the EMD and plug into connector X32 on the robot junction box.

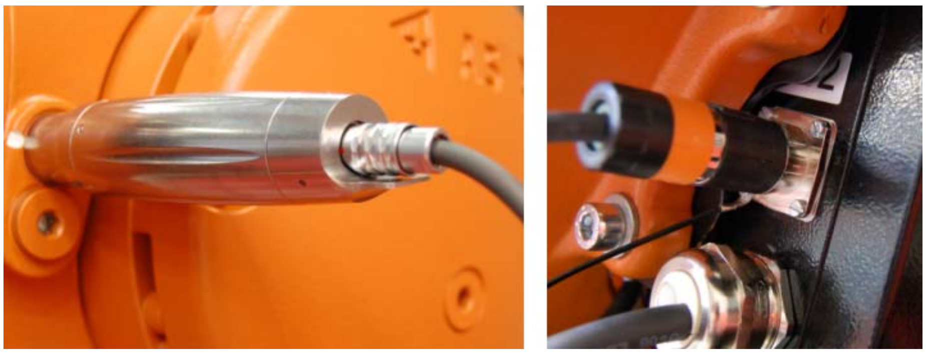


Fig. 3-8: EMD cable, connected

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

4. Press **Master**.
5. Press the enabling switch into the center position and hold it down, and press and hold down the Start key.

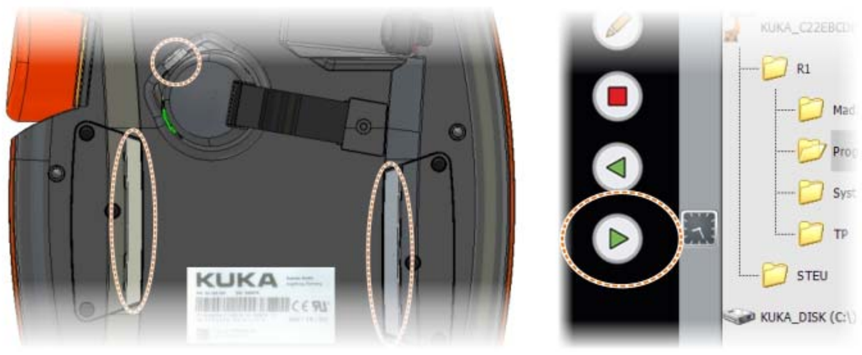


Fig. 3-9: Enabling switch and start key

When the EMD has passed through the lowest point of the reference notch, the mastering position is reached. The robot stops automatically. The values are saved. The axis is no longer displayed in the window.

6. Remove the signal cable from the EMD. Then remove the EMD from the gauge cartridge and replace the protective cap.
7. Repeat steps 2 to 5 for all axes to be mastered.
8. Close the window.
9. Remove signal cable from connection X32.

Procedure for teaching an offset

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

1. Move the robot into the pre-mastering position.
2. Select **Start-up** > **Master** > **EMD** > **With load correction** > **Teach offset** in the main menu.
3. Enter tool number. Confirm with **Tool OK**.

   A window opens. All axes for which the tool has not yet been taught are displayed. The axis with the lowest number is highlighted.
4. Remove the protective cap of the gauge cartridge on the axis highlighted in the window. Screw the EMD onto the gauge cartridge. Then attach signal cable to EMD and plug into connector X32 on the base frame junction box.
5. Press **Learn**.
6. Press an enabling switch and the Start key.

   When the EMD detects the lowest point of the reference notch, the mastering position is reached. The robot stops automatically. A window opens. The deviation of this axis from the first mastering is indicated in degrees and increments.
7. Click **OK** to confirm. The axis is no longer displayed in the window.

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

Procedure for set/check load mastering with offset

Load mastering with offset is carried out with a load. The first mastering is calculated.

1. Move robot to the pre-mastering position.
2. In the main menu, select **Start-up** > **Master** > **EMD** > **With load correction** > **Master load** > **With offset**.
3. Enter tool number. Confirm with **Tool OK**.
4. Remove the cover from connection X32 and connect the signal cable.
5. Remove the protective cap of the gauge cartridge on the axis highlighted in the window. (Turned around, the EMD can be used as a screwdriver.)
6. Screw the EMD onto the gauge cartridge.
7. Attach the signal cable to the EMD, aligning the red dot on the connector with the groove in the EMD.
8. Press **Check**.
9. Hold down an enabling switch and press the Start key.
10. If required, press “Save” to save the values. The old mastering values are deleted. To restore a lost first mastering, always save the values.
11. Remove the signal cable from the EMD. Then remove the EMD from the gauge cartridge and replace the protective cap.
12. Repeat steps 4 to 10 for all axes to be mastered.
13. Close the window.
14. Remove signal cable from connection X32.

3.3 Exercise: Robot mastering

Aim of the exercise

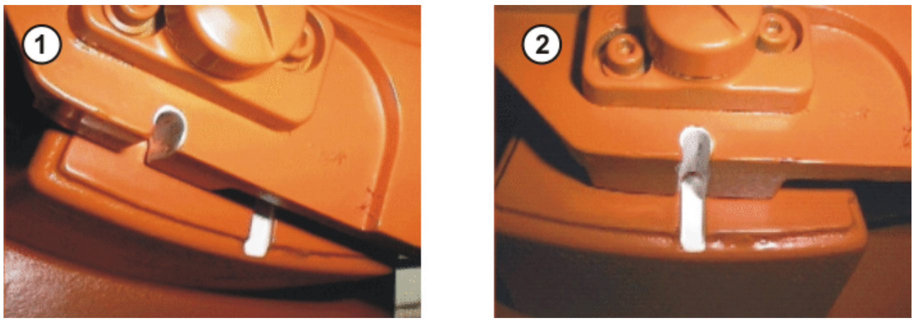
On successful completion of this exercise, you will be able to carry out the following activities:

- Move to pre-mastering position
- Select the correct mastering type
- Work with the “Electronic Mastering Device” (EMD)
- Master all axes using the EMD

Preconditions

The following are preconditions for successful completion of this exercise:

- Theoretical knowledge of the general procedure for mastering
- Theoretical knowledge of the location of the pre-mastering position



1	Axis not in pre-mastering position	✗
2	Axis in pre-mastering position	✓

- Correct connection of the EMD to the robot
- Mastering via the Setup menu

Task description

Carry out the following tasks:

1. Unmaster all robot axes
2. Move all robot axes to the pre-mastering position in joint mode
3. Perform load mastering with offset for all axes using the EMD
4. Display the actual position in joint mode.

Questions on the exercise

1. Why is mastering carried out?

. . . .

. . . .

2. Specify the angles of all 6 axes in the mechanical zero position.

A1:	....	A2:	....
A3:	....	A4:	....
A5:	....	A6:	....

3. What must be taken into consideration with an unmastered robot?

. . . .

. . . .

. . . .

4. Which mastering tool should be used for preference?

. . . .

. . . .

5. What is the danger of moving the robot with the EMD (dial gauge) screwed in place?

. . . .

. . . .

3.4 Loads on the robot

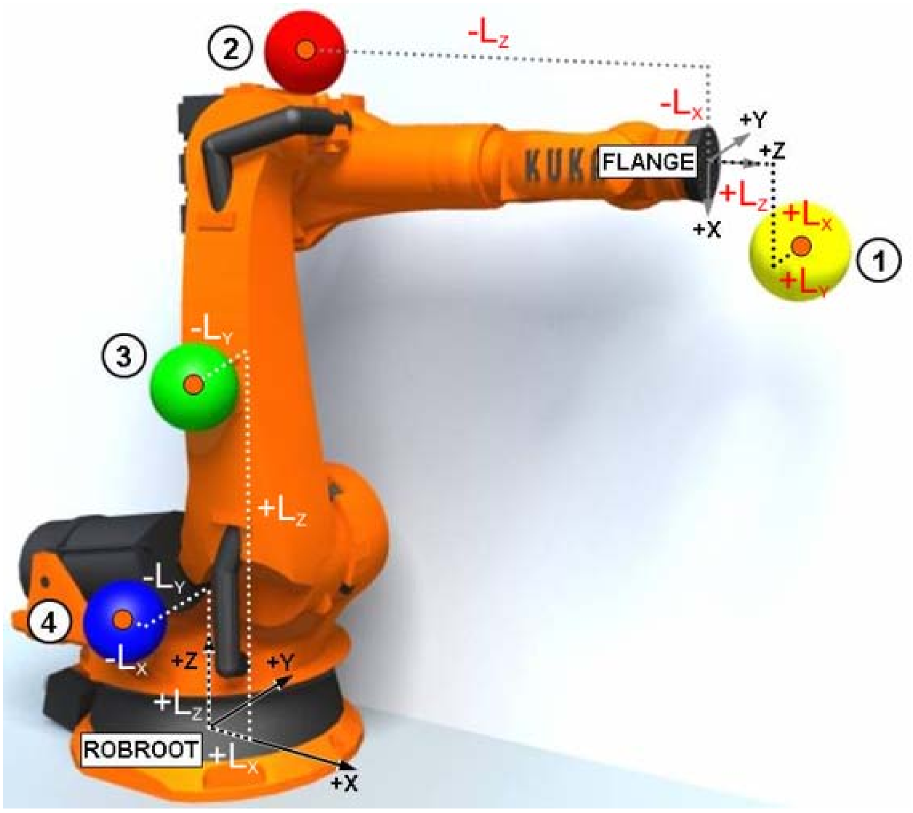


Fig. 3-10: Loads on the robot

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

3.4.1 Tool load data

What are tool load data?

Tool load data refer to all the loads mounted on the robot flange. They form an additional mass mounted on the robot that must also be moved together with the robot.

The values to enter are the mass, the position of the center of gravity (point on which the mass acts) and the mass moments of inertia with the corresponding principal inertia axes.

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

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

Tool load data can be obtained from the following sources:

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

Effects of the load data

The entered load data affect a wide range of controller processes. These include, for example:

- Control algorithms (calculation of acceleration)
- Velocity and acceleration monitoring

- Torque monitoring
- Collision detection
- Energy monitoring
- and many more.

It is thus very important that the load data are entered correctly. If the robot executes its motions with correctly entered load data...

- one profits from its great accuracy.
- motion sequences with optimal cycle times are possible.
- the robot has a long service life (due to reduced wear).

Procedure

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

     (JX is the inertia about the X axis of the coordinate system that is rotated relative to the flange by A, B and C. JY and JZ are the analogous inertia about the Y and Z axes.)
4. Confirm with **Continue**.
5. Press **Save**.

3.4.2 Supplementary loads on the robot

Supplementary loads on the robot

Supplementary loads are additional components mounted on the base frame, link arm or arm, e.g.:

- Energy supply system
- Valves
- Materials feeder
- Materials supply

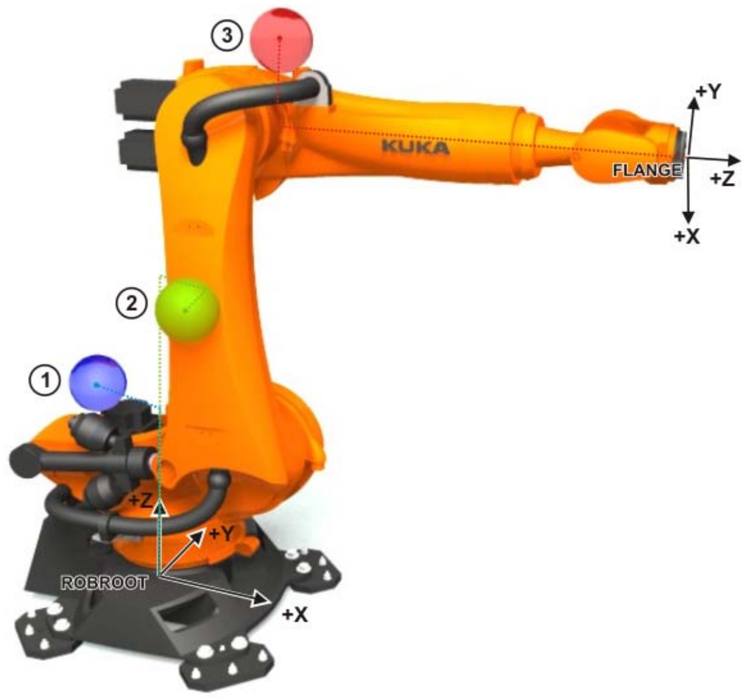


Fig. 3-11: Supplementary loads on the robot

The supplementary load data must be entered in the robot controller. Required specifications include:

- Mass (m) in kg
- Distance from center of mass to the reference system (X, Y and Z) in mm
- Orientation of the principal inertia axes relative to the reference system (A, B and C) in degrees (°)
- Mass moments of inertia about the inertia axes (Jx, Jy and Jz) in kgm²

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

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

Supplementary load data can be obtained from the following sources:

- Manufacturer information
- Manual calculation
- CAD programs

Influence of the supplementary loads on the robot motion

Specification of the load data influences the robot motion in various ways:

- Path planning
- Accelerations
- Cycle time
- Wear

⚠ 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.

Procedure

1. In the main menu, select **Setup** > **Measure** > **Supplementary load data**.
2. Enter the number of the axis on which the supplementary load is to be mounted. Confirm with **Continue**.
3. Enter the load data. Confirm with **Continue**.
4. Press **Save**.

3.5 Tool calibration

Description

Tool *calibration* means the generation of a coordinate system which has its origin in a reference point of the tool. This reference point is called the **TCP** (Tool Center Point); the coordinate system is the **TOOL** coordinate system.

Tool calibration thus consists of calibration...

- of the TCP (origin of the coordinate system).
- of the orientation/alignment of the coordinate system.

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

During calibration, the distance between the origin of the tool coordinate system (in X, Y and Z) and the flange coordinate system, and their rotation relative to one another (angles A, B and C), is saved.

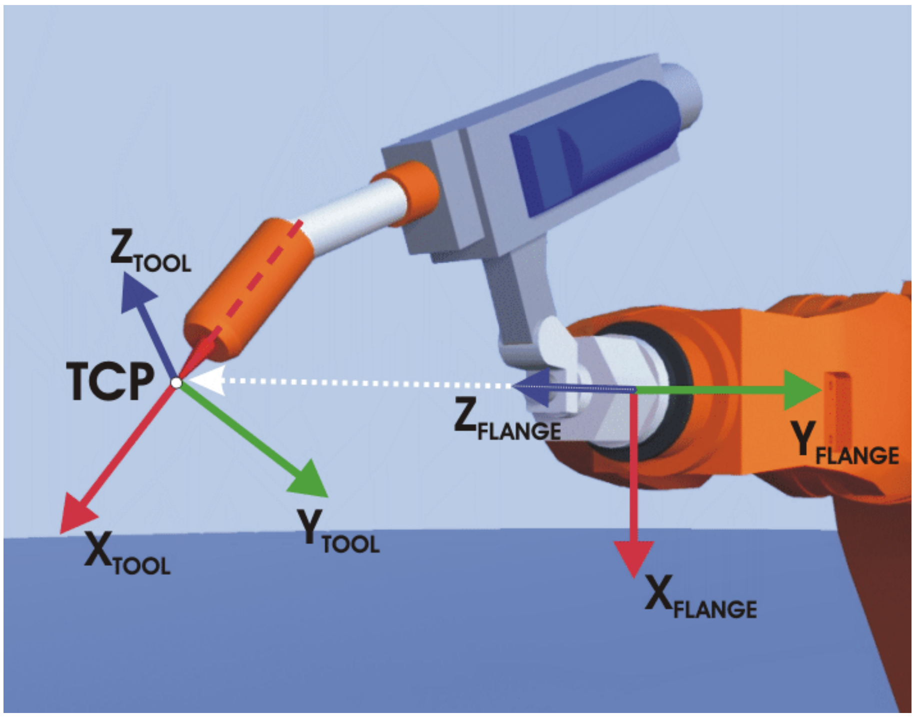


Fig. 3-12: TCP calibration principle

Advantages

If a tool has been calibrated precisely, this has the following practical advantages for the operating and programming personnel:

- Improved jogging
  - Reorientation about the TCP (e.g. tool tip) is possible.

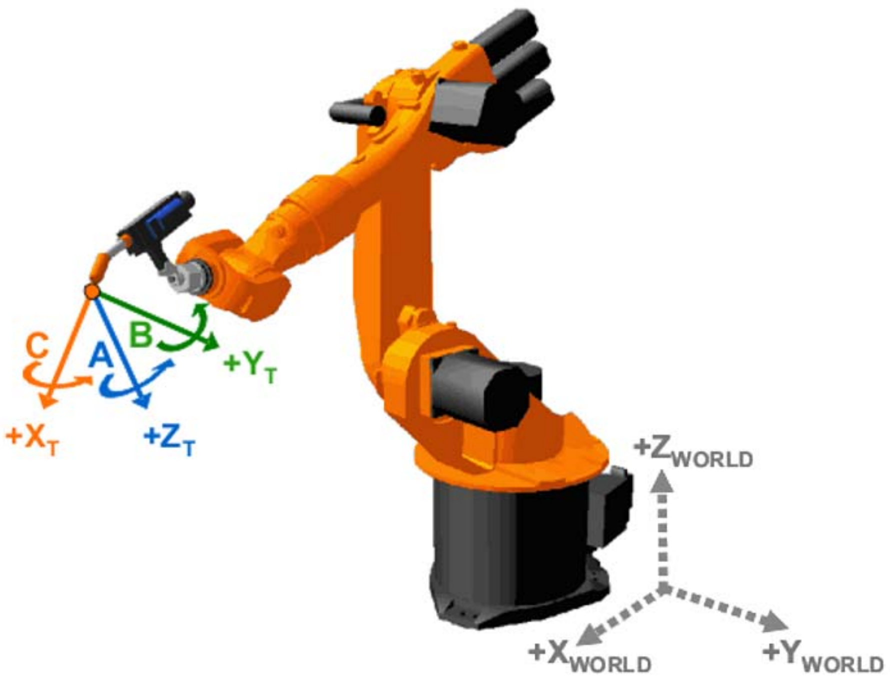


Fig. 3-13: Reorientation about the TCP

- Moving the robot in the tool direction

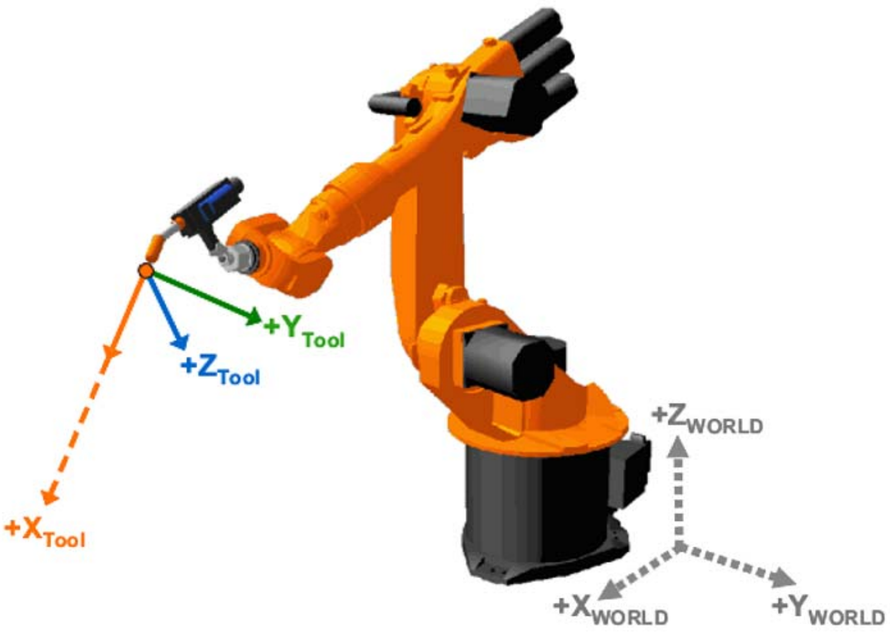


Fig. 3-14: Tool working direction

- Use during motion programming
  - The programmed velocity is maintained at the TCP along the path.

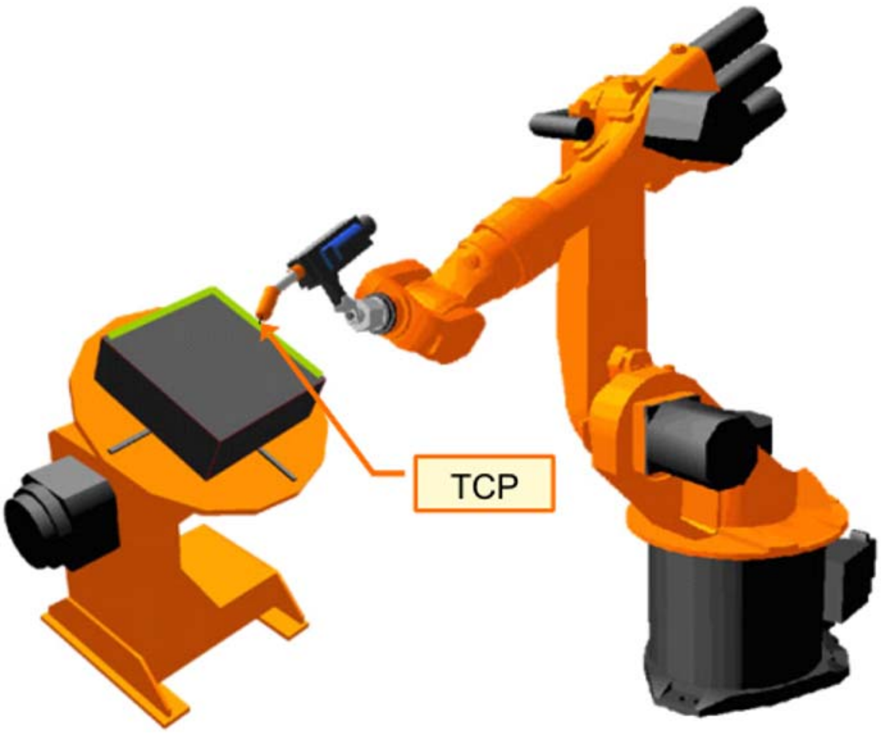


Fig. 3-15: Program mode with TCP

- Furthermore, a defined orientation along the path is possible.

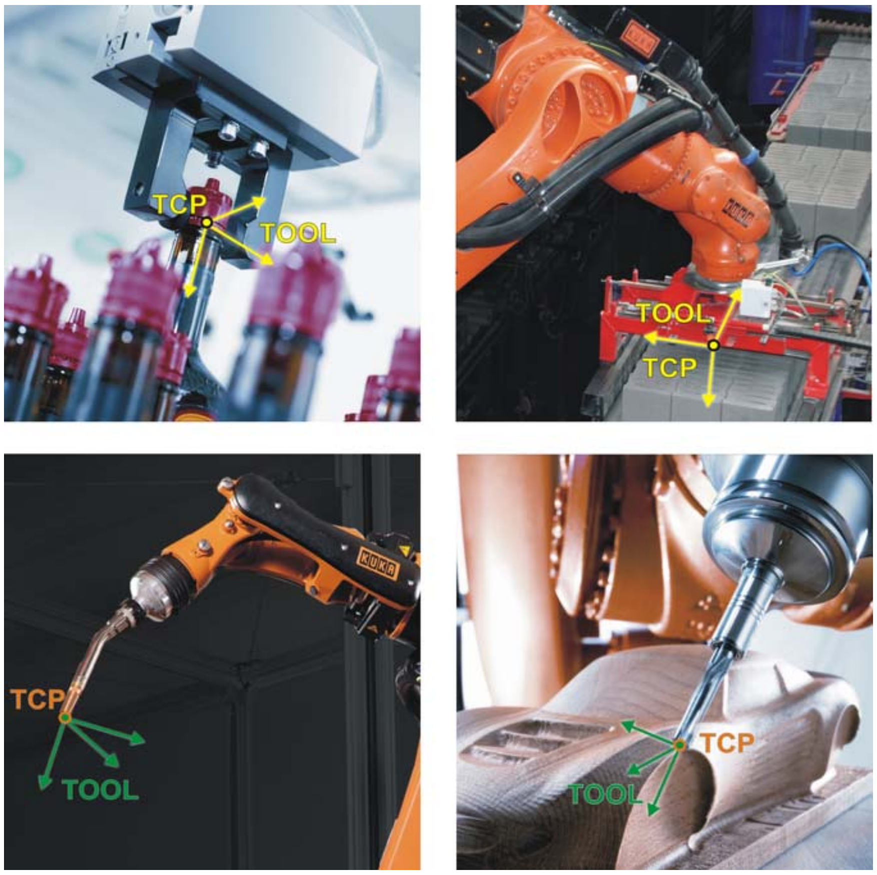


Fig. 3-16: Examples of calibrated tools

Tool calibration options

Tool calibration consists of 2 steps:

Step	Description
1	**Definition of the origin of the TOOL coordinate system** The following methods are available: ■ *XYZ 4-point* ■ *XYZ Reference*
2	**Definition of the orientation of the TOOL coordinate system** The following methods are available: ■ *ABC World* ■ *ABC 2-point*
Alternative	Direct entry of the values for the distance from the flange center point (X,Y,Z) and the rotation (A, B, C). ■ *Numeric input*

TCP calibration: XYZ 4-point method

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

The 4 flange positions at the reference point must be sufficiently different from one another and must not lie in a plane.

Procedure for XYZ 4-point method:

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 **Next**.
3. Move the TCP to a reference point. Press the **Calibrate** softkey and confirm the dialog “Apply current position? Resuming calibration.” with **Yes**.
4. Move the TCP to the reference point from a different direction. Press **Calibrate** again and answer the dialog with **Yes**.

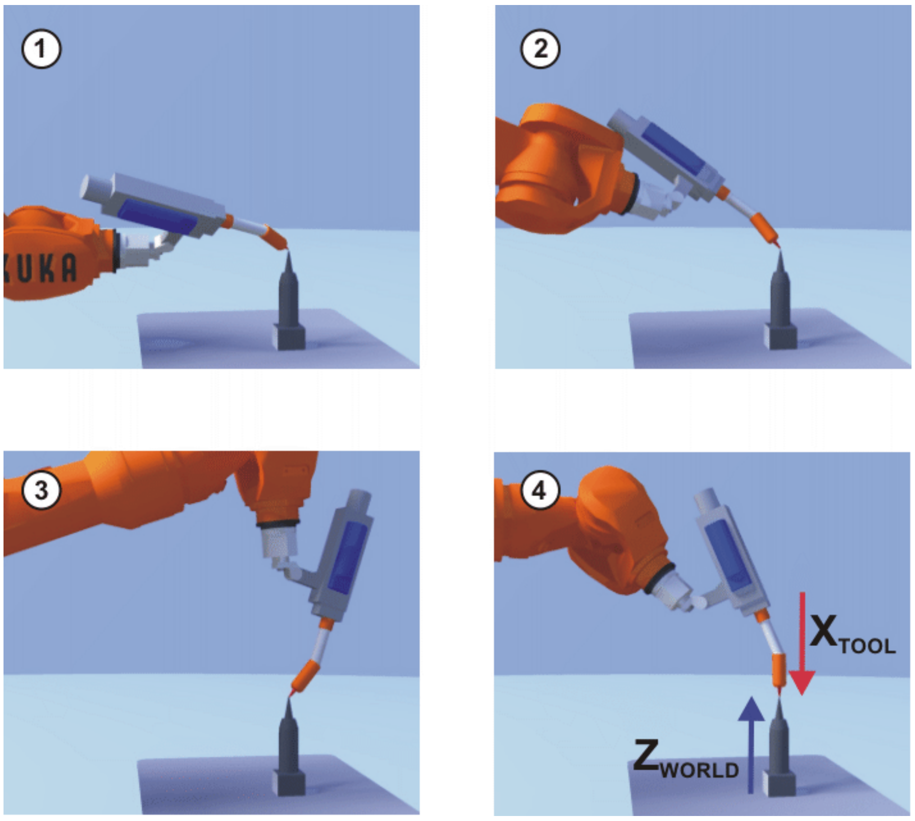


Fig. 3-17: XYZ 4-Point method

5. Repeat step 4 twice.
6. The load data entry window is opened. Enter the load data correctly and confirm with **Next**.
7. The window with the calculated X, Y and Z values for the TCP opens and the calibration inaccuracy can be read under “Errors”. Data can be saved directly by pressing **Save**.

TCP calibration: XYZ Reference method

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.

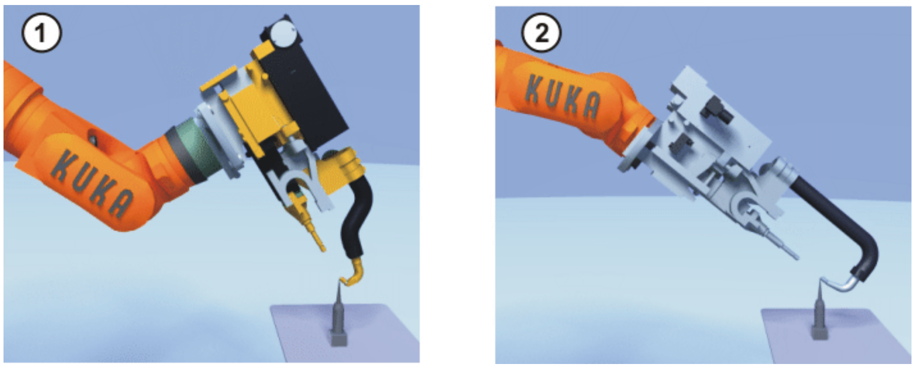


Fig. 3-18

Procedure

1. A precondition here is that a calibrated tool is mounted on the flange and that the data of the TCP are known.
2. In the main menu, select **Start-up** > **Calibrate** > **Tool** > **XYZ Reference**.
3. Assign a number and a name for the new tool. Confirm with **Next**.
4. Enter the TCP data of the calibrated tool. Confirm with **Next**.

5. Move the TCP to a reference point. Press **Calibrate**. Confirm with **Next**.
6. Move the tool away and remove it. Mount the new tool.
7. Move the TCP of the new tool to the reference point. Press **Calibrate**. Confirm with **Next**.
8. Press **Save**. The data are saved and the window is closed.

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

Orientation calibration: ABC World method

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

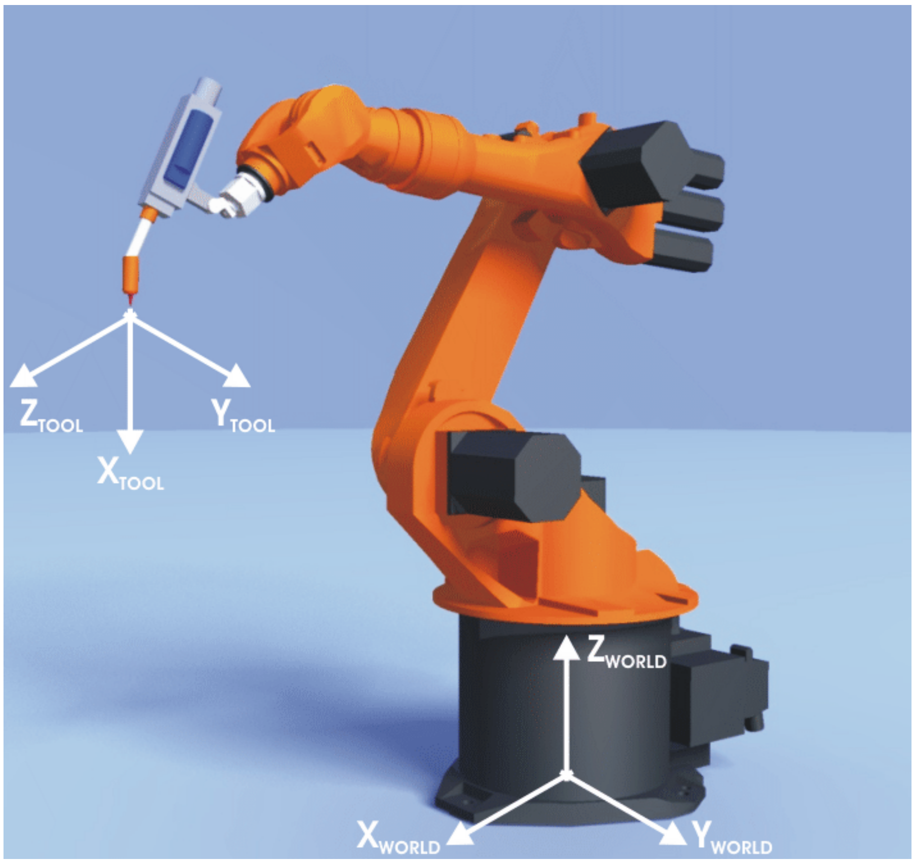


Fig. 3-19: ABC World method

Procedure for ABC World method

a. In the main menu, select **Start-up** > **Calibrate** > **Tool** > **ABC World**.
b. Enter the number of the tool. Confirm with **Next**.
c. Select a variant in the box **5D/6D**. Confirm with **Next**.
d. If **5D** is selected:

   Align $+X_{TOOL}$ parallel to $-Z_{WORLD}$. ($+X_{TOOL}$ = tool direction)
e. If **6D** is selected:

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

Align $+Y_{TOOL}$ parallel to $+Y_{WORLD}$. ($+X_{TOOL}$ = tool direction)

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

f. Confirm with **Calibrate**. Confirm the message “Apply current position? Resuming calibration.” with **Yes**.

g. Another window opens. The load data must be entered here.

h. Then complete the operation with **Next** and **Save**.

i. Close the menu.

Orientation calibration: ABC 2-point method

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.

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

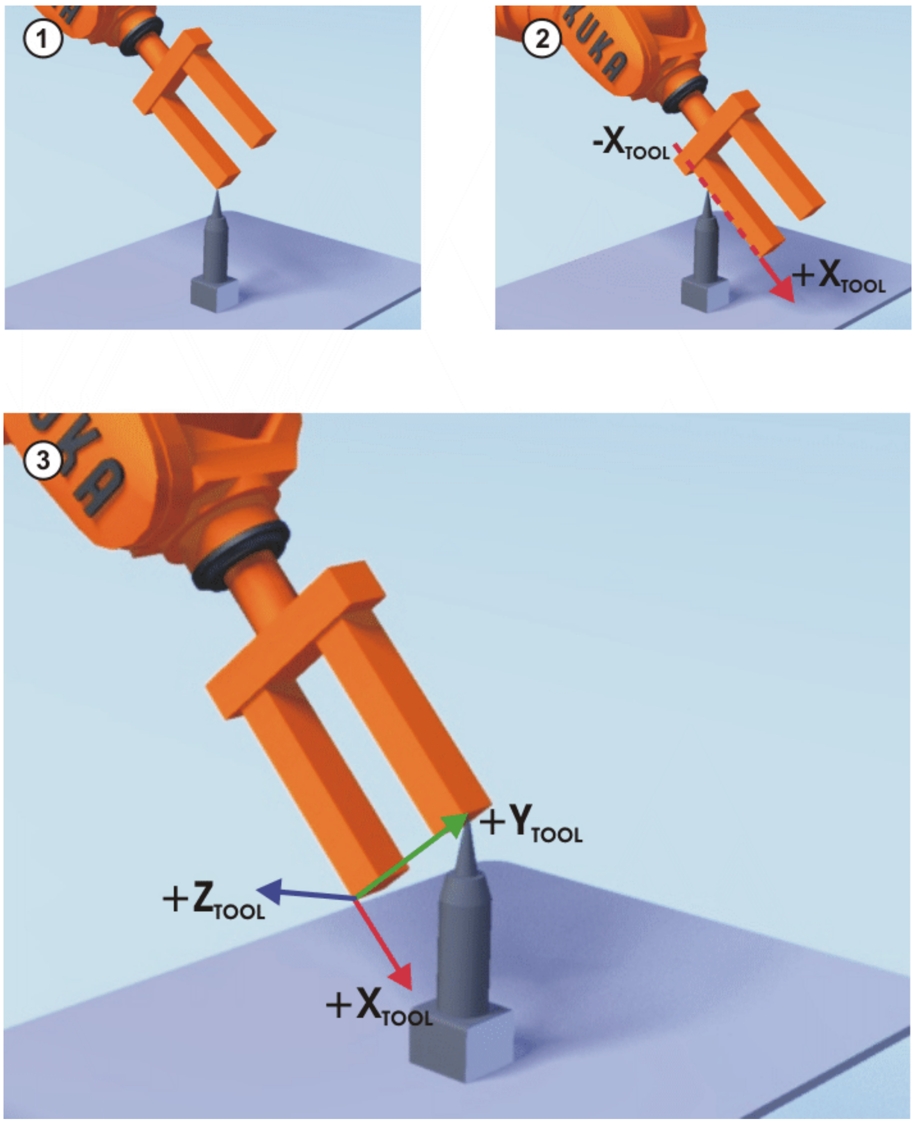


Fig. 3-20: ABC 2-point method

1. A precondition is that the TCP has already been calibrated by means of the XYZ method.
2. In the main menu, select **Start-up** > **Calibrate** > **Tool** > **ABC 2-point**.
3. Enter the number of the mounted tool. Confirm with **Next**.

4. Move the TCP to any reference point. Press **Calibrate**. Confirm with **Next**.
5. Move the tool so that the reference point on the X axis has a negative X value (i.e. move against the tool direction). Press **Calibrate**. Confirm with **Next**.
6. Move the tool so that the reference point in the XY plane has a negative Y value. Press **Calibrate**. Confirm with **Next**.
7. Either press **Save**. The data are saved and the window is closed.

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

Gripper safety instructions for training mode

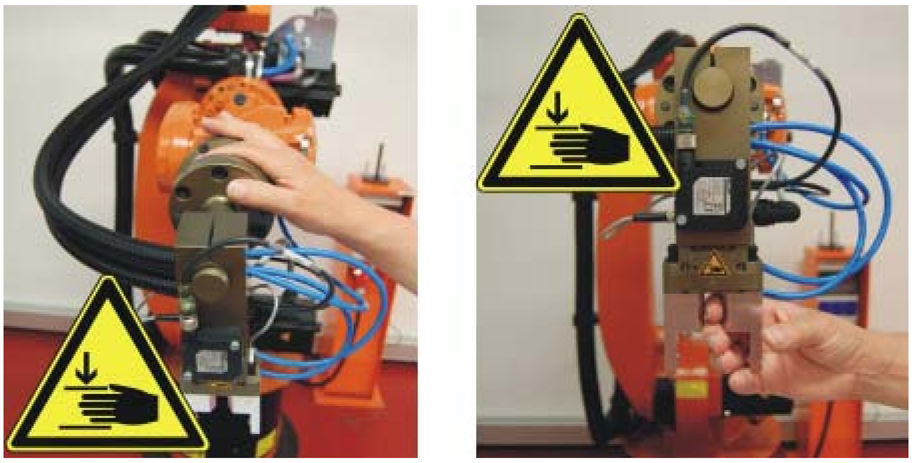


Fig. 3-21: Crushing hazards on the training gripper

⚠ WARNING

Warning!
When using the gripper system there is a risk of crushing and cutting. Anyone using the gripper must ensure that no part of the body can be crushed by the gripper.

Great care must be taken when clamping workpieces (cube, pen).

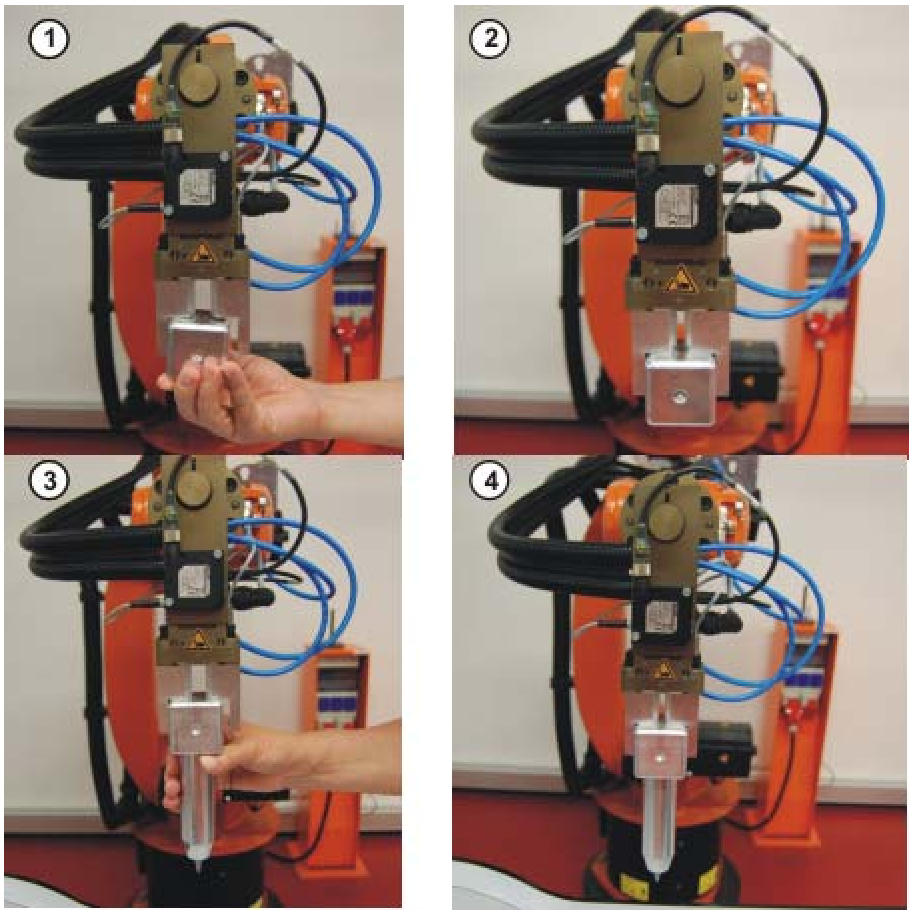


Fig. 3-22: Clamping objects in the training gripper

Item	Comments
1	Clamping the cube
2	Clamped cube
3	Clamping a pen
4	Clamped pen

The collision protection device is triggered in the event of a collision.

The robot can be moved clear after the collision protection device has been triggered during a collision. One course participant presses the button (1) and keeps all parts of his/her body well away from the robot, the collision protection device and the gripper. Before moving the robot clear, the second participant ensures that no-one will be put at risk by the motion of the robot.

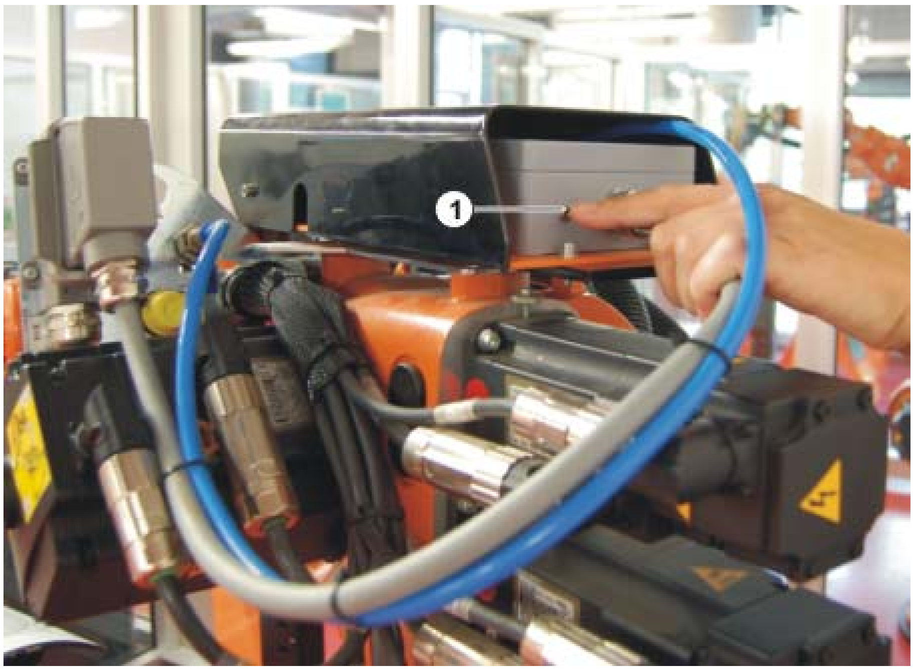


Fig. 3-23: Button for clearing the collision protection device

3.6 Exercise: Tool calibration, pen

Aim of the exercise

On successful completion of this exercise, you will be able to carry out the following activities:

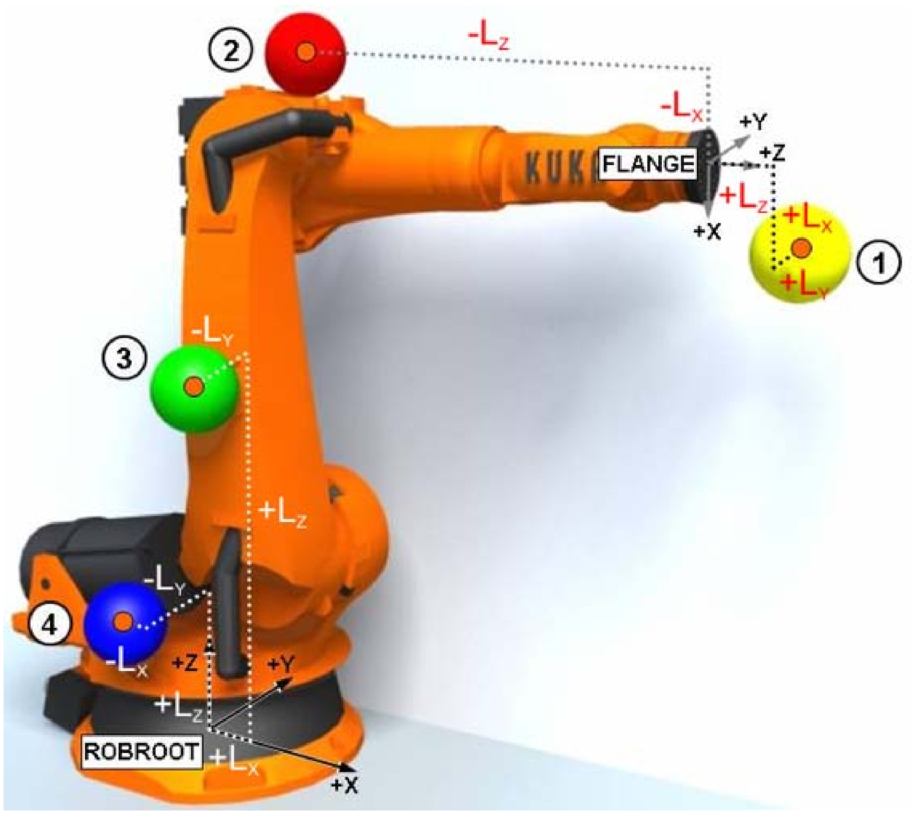
- Calibration of a tool using the XYZ 4-Point and ABC World methods
- Activation of a calibrated tool
- Moving the robot in the tool coordinate system
- Moving the robot in the tool direction
- Reorientation of the tool about the Tool Center Point (TCP)

Preconditions

The following are preconditions for successful completion of this exercise:

- Theoretical knowledge of the various TCP calibration methods, especially the XYZ 4-Point method
- Theoretical knowledge of the various tool orientation calibration methods, especially the ABC World method

- Theoretical knowledge of robot load data and how to enter them



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

Task description

Carry out the following tasks: pen calibration

1. Calibrate the TCP of the pen using the XYZ 4-Point method. Use the black metal tip as the reference point. Remove the uppermost pen from the pen magazine and clamp it in the gripper. Use the **tool number 2** and assign the name **Pen1**. The tolerance should not exceed 0.95 mm.
2. Save the tool data.
3. Calibrate the tool orientation using the ABC World 5D method.
4. Enter the load data.

   Load data for the gripper with pen as tool number 2:

Mass:

M = 7.32 kg

Center of mass:

X= 21 mm	Y= 21 mm	Z= 23 mm

Orientation:

A = 0°	B = 0°	C = 0°

Moments of inertia:

J_X = 0 kgm^2	J_Y = 0.2 kgm^2	J_Z = 0.3 kgm^2

5. Save the TOOL data and test the robot motion with the pen in the TOOL coordinate system

Questions on the exercise

1. Why should a robot-guided tool be calibrated?

....

....

. . . .

2. What is calculated using the XYZ 4-Point method?

. . . .

. . . .

3. What tool calibration methods are there?

. . . .

. . . .

. . . .

. . . .

3.7 Exercise: Tool calibration of gripper, 2-point method

Aim of the exercise

On successful completion of this exercise, you will be able to carry out the following activities:

- Tool calibration using the XYZ 4-point and ABC 2-point methods
- Activation of a calibrated tool
- Moving the robot in the tool coordinate system
- Moving the robot in the tool direction
- Reorientation of the tool about the Tool Center Point (TCP)

Preconditions

The following are preconditions for successful completion of this exercise:

- Theoretical knowledge of the various TCP calibration methods, especially the 2-point method
- Theoretical knowledge of robot load data and how to enter them

Task description

Carry out the following tasks: Gripper calibration to number ...

1. Calibrate the TCP of the gripper using the XYZ 4-point method as illustrated:
2. Calibrate the orientation of the gripper coordinate system using the ABC 2-point method.
3. Enter the load data.

   Load data for the gripper:

Mass:

M = 6.68 kg

Center of mass:

X= 23 mm	Y= 11 mm	Z= 41 mm

Orientation:

A = 0°	B = 0°	C = 0°

Moments of inertia:

J_X = 0 kgm^2	J_Y = 0.4 kgm^2	J_Z = 0.46 kgm^2

4. Save the TOOL data and test jogging with the gripper in the TOOL coordinate system.

Alternatively, the gripper can also be calibrated by means of numeric input:

X	Y	Z	A	B	C
132.05 mm	171.30 mm	173.00 mm	45°	0°	180°

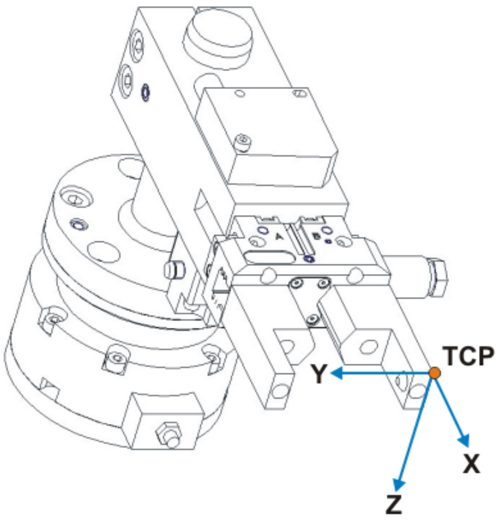


Fig. 3-24: College gripper: position of the TCP

Questions on the exercise

1. Which icon represents the tool coordinate system?

a) ○ b) ○ c) ○ d) ○

2. What is the maximum number of tools that the controller can manage?

. . . .

3. What does the value -1 in the tool load data mean?

. . . .

. . . .

3.8 Base calibration

Description

Calibration of a base means the creation of a coordinate system at a specific point in the robot environment, relative to the world coordinate system. The objective is for the motions and the programmed robot positions to refer to this coordinate system. For this reason, defined edges of workpiece locations, shelves, outer edges of pallets or machines, for example, are useful as reference points for a base coordinate system.

Base calibration is carried out in two steps:

1. Determination of the coordinate origin
2. Definition of the coordinate axes

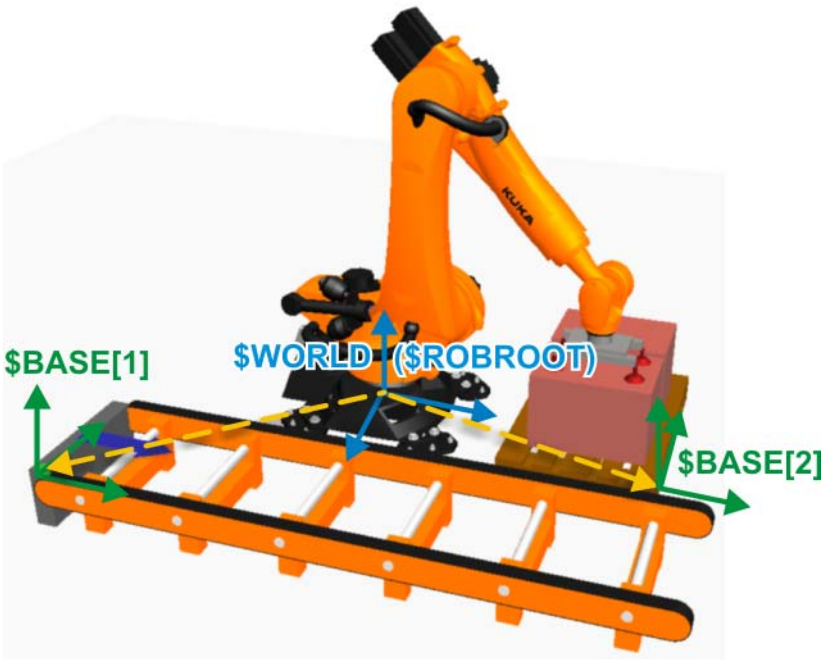


Fig. 3-25: Base calibration

Advantages

Base calibration has the following advantages:

- Motion along the edges of the workpiece:
  The TCP can be jogged along the edges of the work surface or workpiece.

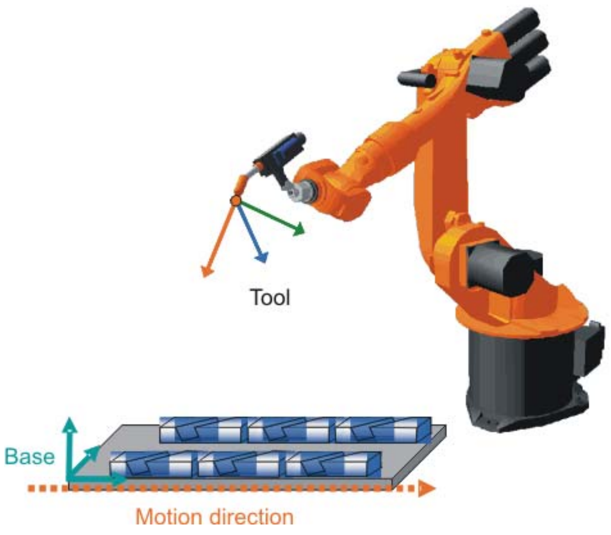


Fig. 3-26: Advantages of base calibration: motion direction

- Reference coordinate system:

  Taught points refer to the selected coordinate system.

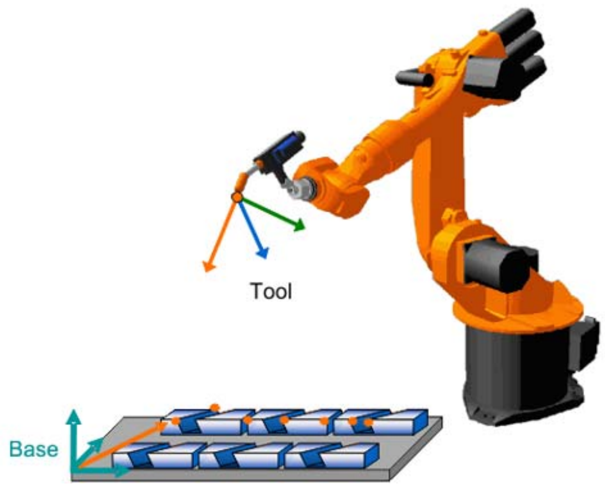


Fig. 3-27: Advantages of base calibration: reference to the desired coordinate system

- Correction / offset of the coordinate system:

  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.

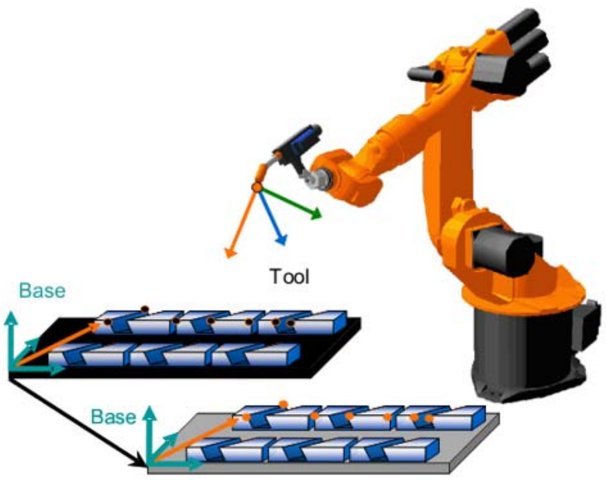


Fig. 3-28: Advantages of base calibration: offset of the base coordinate system

- Using multiple base coordinate systems:

  Up to 32 different coordinate systems can be created and used depending on the specific program sequence.

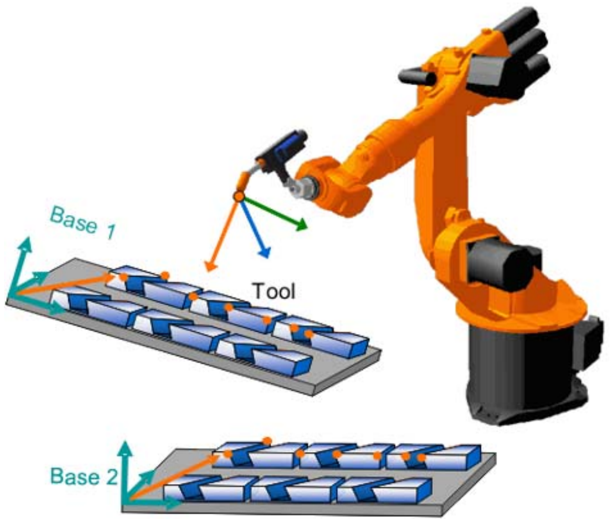


Fig. 3-29: Advantages of base calibration: use of multiple base coordinate systems

Base calibration options

The following base calibration methods are available:

Methods	Description
3-point method	1. Definition of the origin 2. Definition of the positive X axis 3. Definition of the positive Y axis (XY plane)
Indirect method	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 (CAD data). The robot controller calculates the base from these points.
Numeric input	Direct entry of the values for the distance from the world coordinate system (X,Y,Z) and the rotation (A, B, C).

Further information about indirect calibration can be found in the *Operating and Programming Instructions for KUKA System Software 8*.

Procedure for 3-point method

NOTICE Base calibration can only be carried out with a previously calibrated tool (TCP must be known).

1. In the main menu, select **Start-up** > **Calibrate** > **Base** > **ABC 3-point**.
2. Assign a number and a name for the base. Confirm with **Next**.
3. Enter the number of the tool whose TCP is to be used for for base calibration. Confirm with **Next**.
4. Move the TCP to the origin of the new base. Press the **Calibrate** softkey and confirm the position with **Yes**.

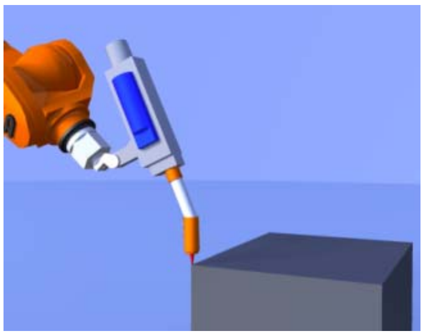


Fig. 3-30: First point: origin

5. Move the TCP to a point on the positive X axis of the new base. Press **Calibrate** and confirm the position with **Yes**.

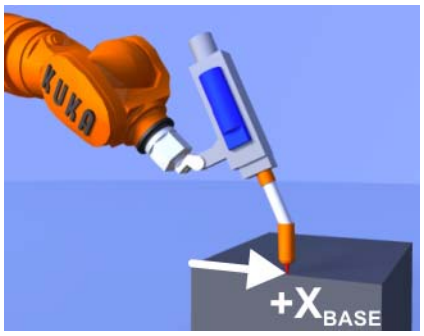


Fig. 3-31: Second point: X axis

6. Move the TCP to a point in the XY plane with a positive Y value. Press **Calibrate** and confirm the position with **Yes**.

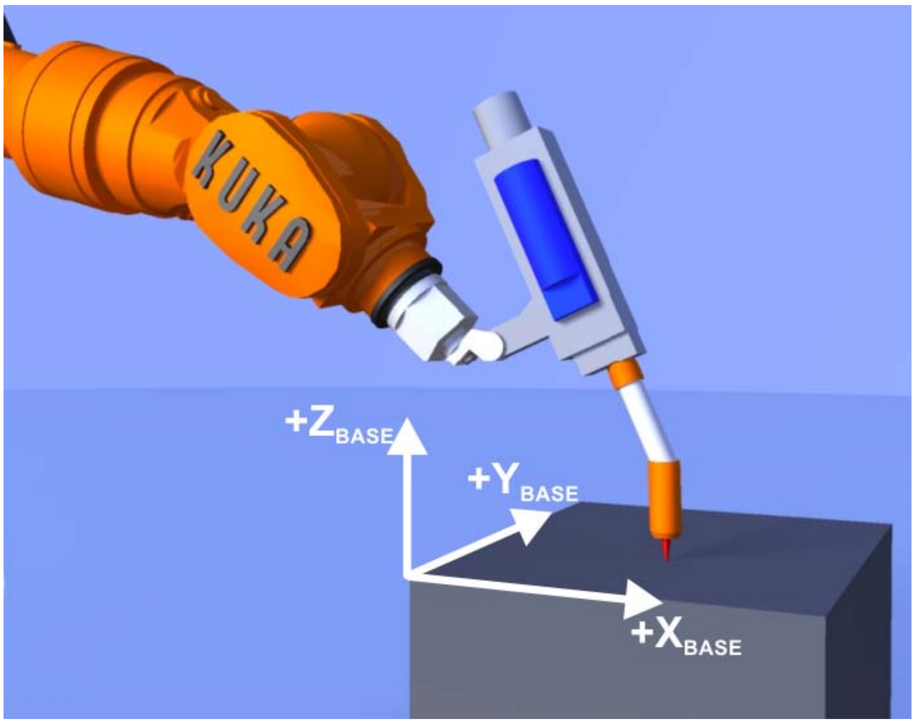


Fig. 3-32: Third point: XY plane

7. Press **Save**.
8. Close the menu.

The three calibration points must not be in a straight line. A minimum angle (default setting 2.5°) between the points must be maintained.

3.9 Displaying the current robot position

Options for displaying robot positions

The current robot position can be displayed in two different ways:

- **Axis-specific:**

$AXIS_ACT={A1...,A2...,A3...,A4...,A5...,A6...,E1...,...,E6...}

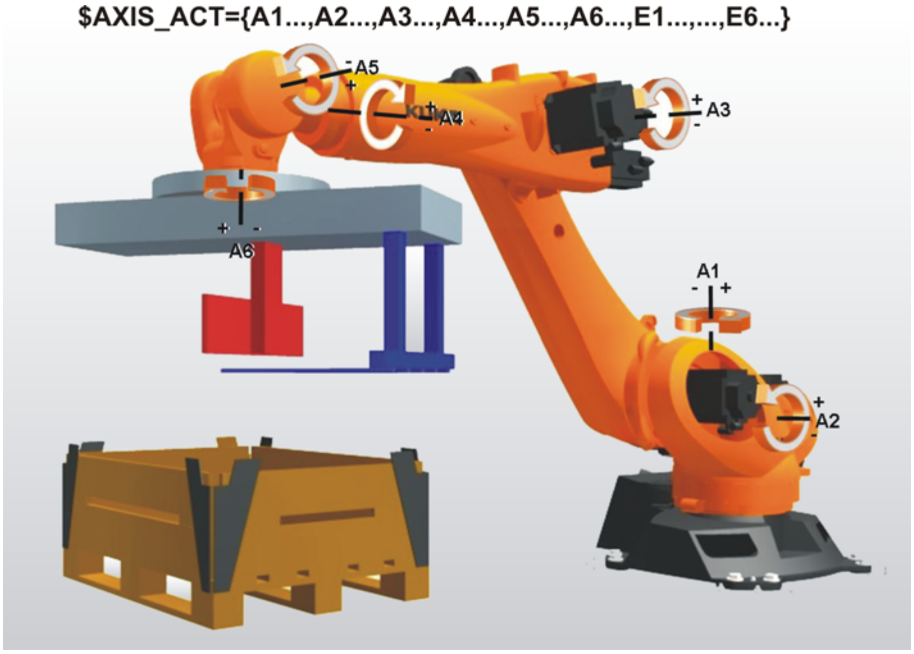


Fig. 3-33: Axis-specific robot position

  The current axis angle is displayed for every axis: this corresponds to the absolute axis angle relative to the mastering position.

- **Cartesian:**

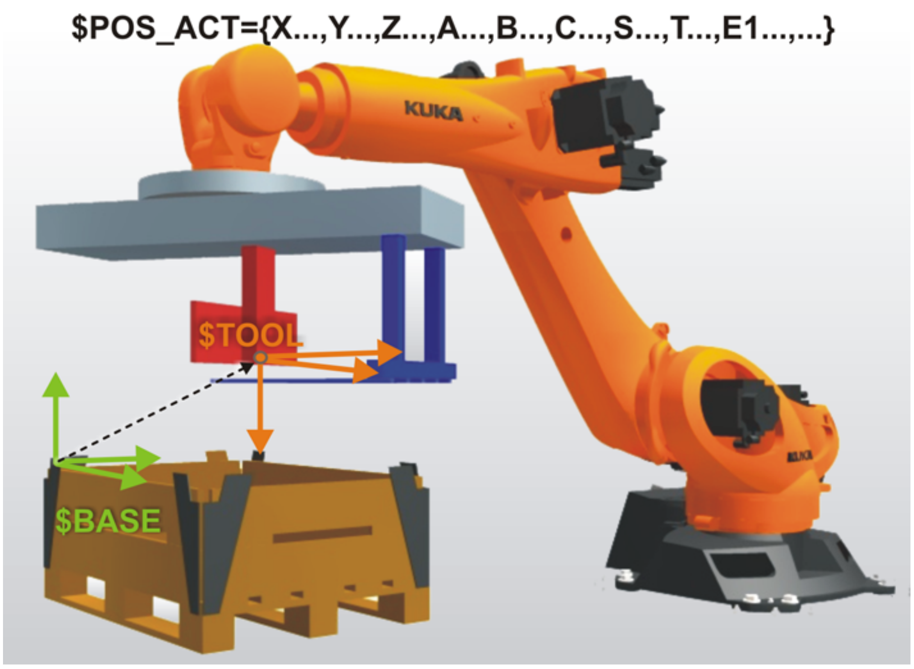

$POS_ACT={X...,Y...,Z...,A...,B...,C...,S...,T...,E1...,...}
$TOOL
$BASE

Fig. 3-34: Cartesian position

The current position of the current TCP (tool coordinate system) is displayed relative to the currently selected base coordinate system.

If no tool coordinate system is selected, the flange coordinate system applies!

If no base coordinate system is selected, the world coordinate system applies!

Cartesian position with different base coordinate systems

Looking at the Figure below, it is immediately apparent that the robot is in the same position in all three instances. The position display contains different values in each of the three cases, however:

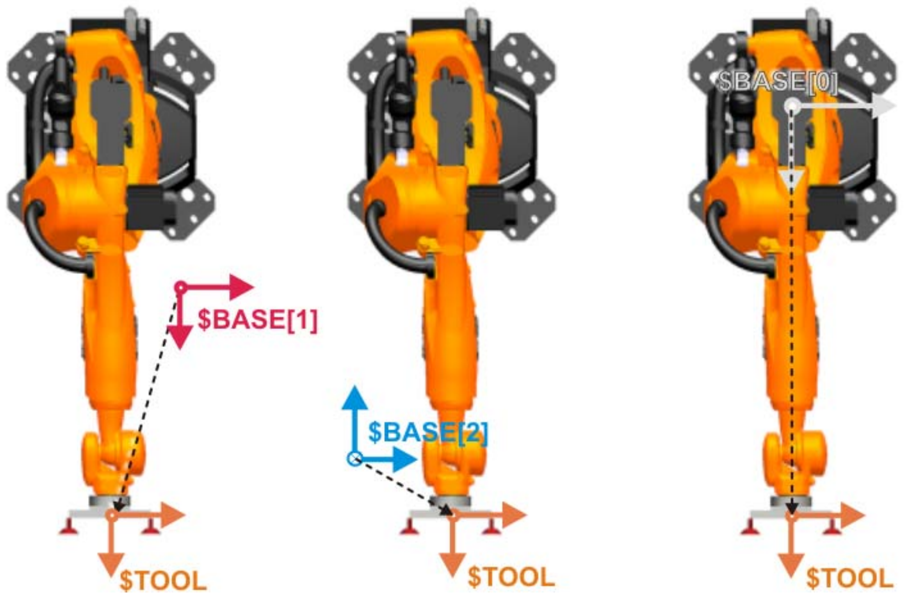


Fig. 3-35: Three robot positions – one robot position!

The position of the tool coordinate system/TCP is displayed in the respective base coordinate system:

- for base **1**
- for base **2**

- for base **$NULLFRAME**: this corresponds to the robot root point coordinate system (in most cases identical to the world coordinate system)!

The Cartesian actual value display only provides the expected values if the correct base and tool are selected!

Displaying the robot position

Procedure:

- Select **Display** > **Actual position** in the menu. The Cartesian actual position is displayed.
- To display the axis-specific actual position, press **Axis spec.**.
- To display the Cartesian actual position again, press **Cartesian**.

3.10 Exercise: Base calibration of table, 3-point method

Aim of the exercise

On successful completion of this exercise, you will be able to carry out the following activities:

- Define any base
- Calibrate a base
- Activate a calibrated base for jogging
- Move the robot in the base coordinate system

Preconditions

The following are preconditions for successful completion of this exercise:

- Theoretical knowledge of the base calibration methods, especially the 3-Point method

Task description

Carry out the following tasks:

1. Calibrate the blue base on the table using the 3-Point method. Assign the **base number 1** and the name **blue**. Use pen1 which has already been calibrated (tool number 2) as the calibration tool.
2. Save the data of the calibrated base.
3. Calibrate the red base on the table using the 3-Point method. Assign the **base number 2** and the name **red**. Use pen1 which has already been calibrated (tool number 2) as the calibration tool.
4. Save the data of the calibrated base.
5. Move the tool to the origin of the blue coordinate system, thus causing the actual position to be displayed in "Cartesian" mode.

X	Y	Z	A	B	C
....	....	....	....	....	....

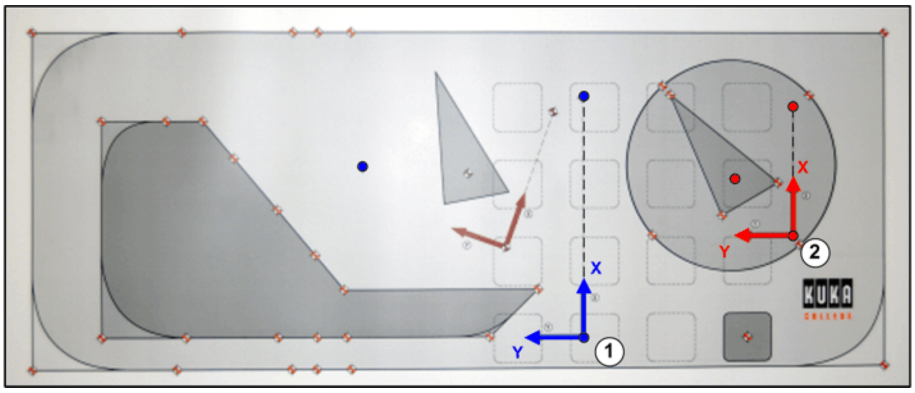


Fig. 3-36: Base calibration on the table

Questions on the exercise

1. Why should a base be calibrated?

. . . .

. . . .

. . . .

2. Which icon represents the base coordinate system?

a) b) c) d)

3. What base calibration methods are there?

. . . .

. . . .

. . . .

4. What is the maximum number of base systems that the controller can manage?

. . . .

5. Describe calibration using the 3-Point method

. . . .

. . . .

. . . .

3.11 Calibration of a fixed tool

Overview

Calibration of the fixed tool consists of two steps:

1. Calculation of the distance between the external TCP of the fixed tool and the origin of the world coordinate system.
2. Orientation of the coordinate system at the external TCP.

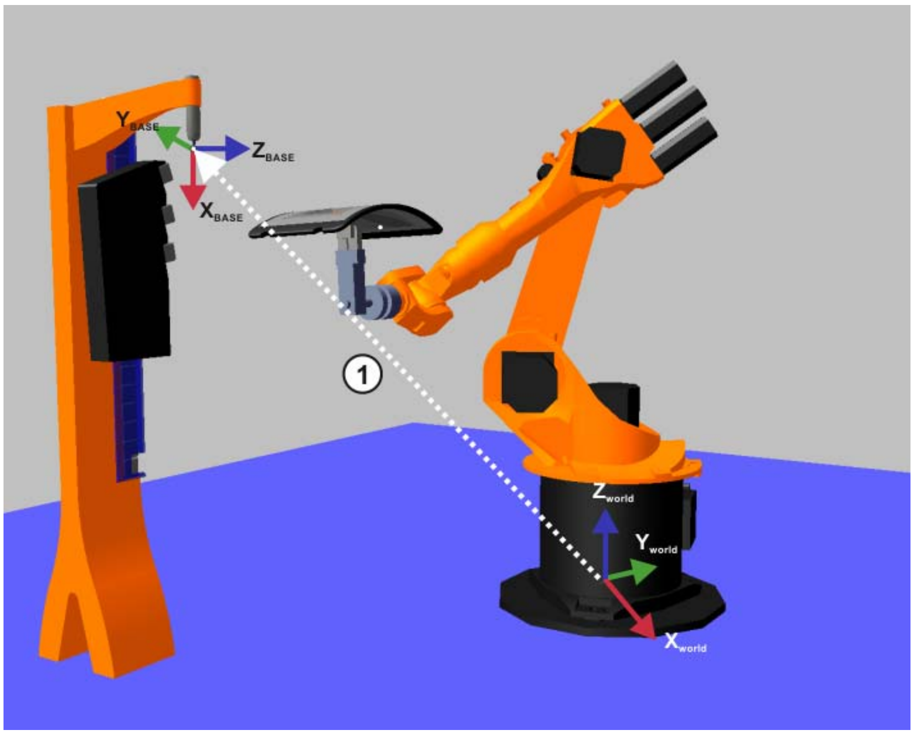


Fig. 3-37: Calibration of the fixed tool

As illustrated in (1) (>>> Fig. 3-37), the external TCP is managed relative to $WORLD (or $ROBROOT), i.e. in the same way as a base coordinate system.

Description of calibration

- A calibrated, robot-guided tool is required for determining the TCP.

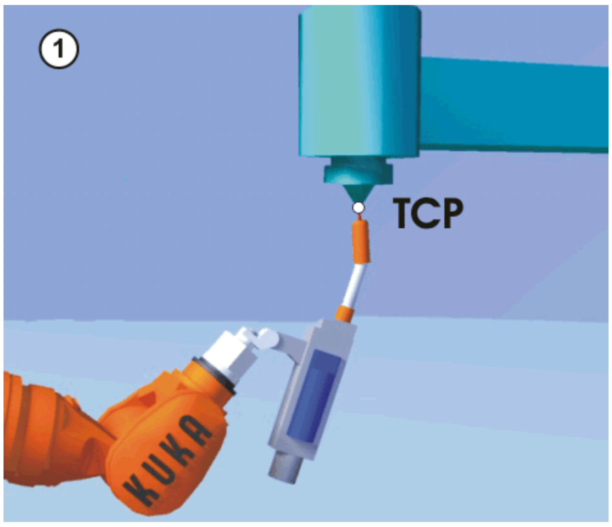


Fig. 3-38: Moving to the external TCP

- To determine the orientation, the flange coordinate system 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 orienta-

tion 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.

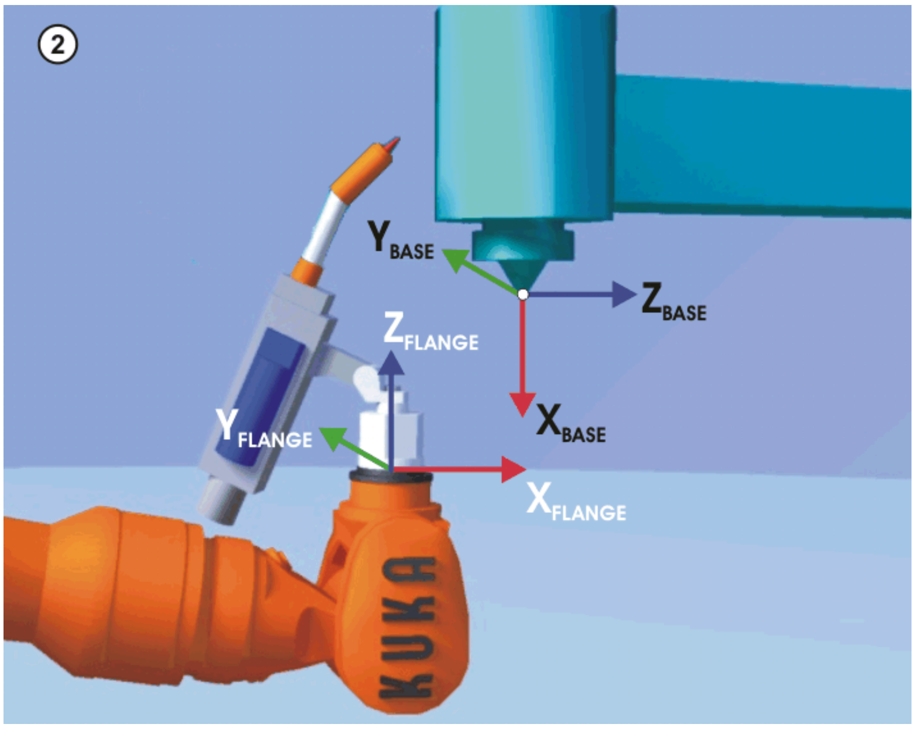


Fig. 3-39: Aligning the coordinate systems parallel to one another

Procedure

1. In the main menu, select **Start-up** > **Calibrate** > **Fixed tool** > **Tool**.
2. Assign a number and a name for the fixed tool. Confirm with **Next**.
3. Enter the number of the reference tool used. Confirm with **Next**.
4. Select a variant in the box **5D/6D**. Confirm with **Next**.
5. Move the TCP of the calibrated tool to the TCP of the fixed tool. Press **Calibrate**. Confirm position with **Yes**.
6. If **5D** is selected:

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

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

   If **6D** is selected:

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

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

     (i.e. align the mounting flange perpendicular to the tool direction.)
   - $+Y_{BASE}$ parallel to $+Y_{FLANGE}$
   - $+Z_{BASE}$ parallel to $+X_{FLANGE}$
7. Press **Calibrate**. Confirm position with **Yes**.
8. Press **Save**.

3.12 Calibration of a robot-guided workpiece

Overview: Direct calibration

NOTICE Only the direct measuring method is explained here. Indirect calibration is extremely rare and is described in greater detail in the documentation *KUKA System Software 8.1 – Operating and Programming Instructions*.

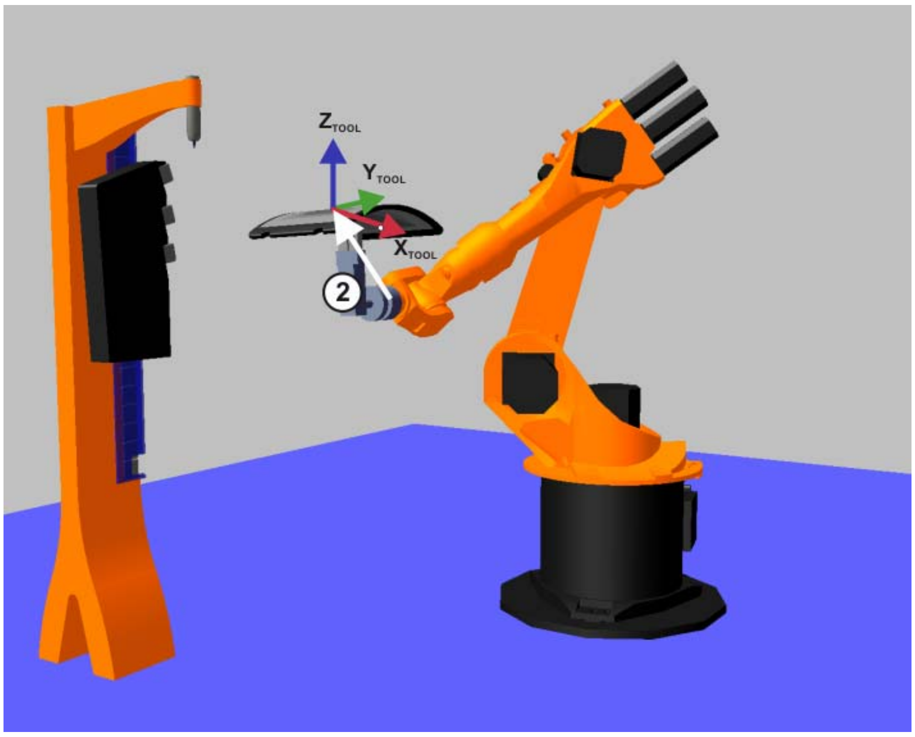


Fig. 3-40: Workpiece calibration by means of direct measuring

Part	Calibration
2	Calibration of the workpiece

Description

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

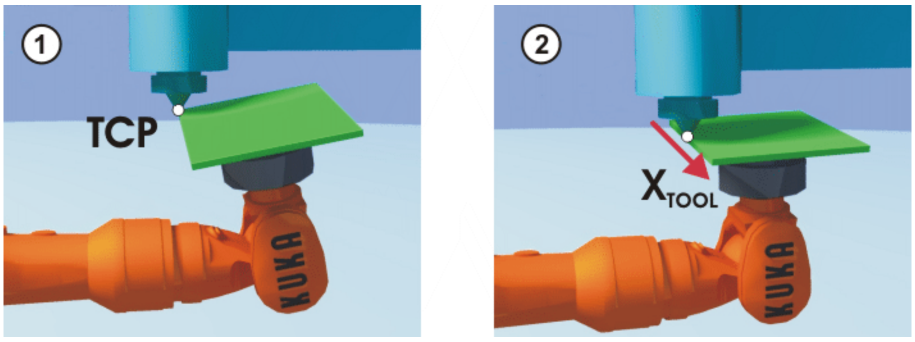


Fig. 3-41

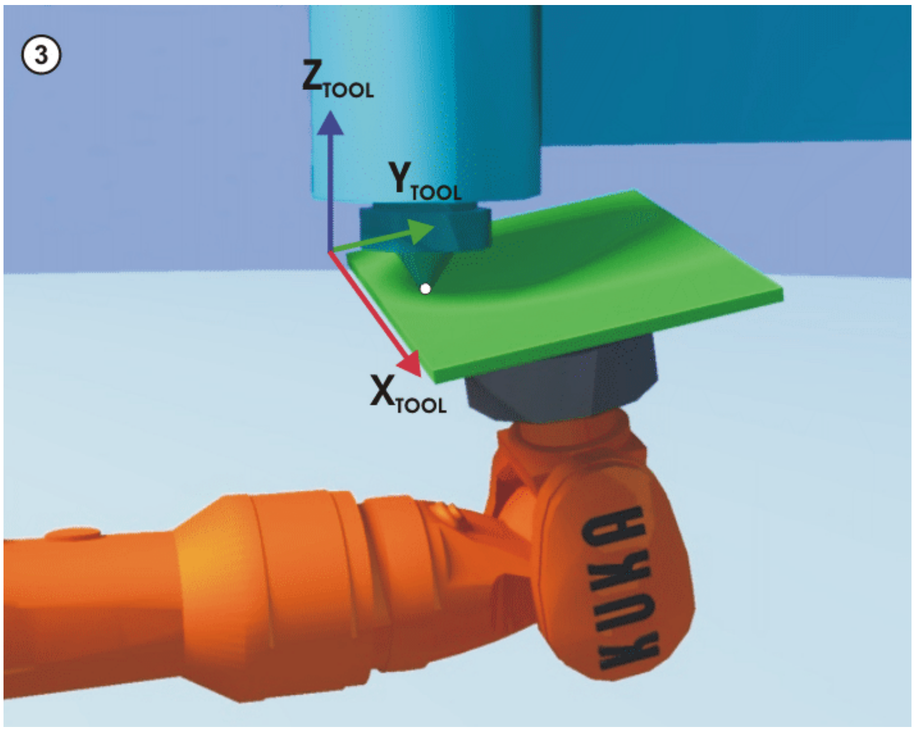


Fig. 3-42: Workpiece calibration: direct method

Procedure

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

   Press **Calibrate** and confirm the position with **Yes**.
5. Move a point on the positive X axis of the workpiece coordinate system to the TCP of the fixed tool.

   Press **Calibrate** and confirm the position with **Yes**.
6. Move a point with a positive Y value in the XY plane of the workpiece coordinate system to the TCP of the fixed tool.

   Press **Calibrate** and confirm the position with **Yes**.
7. Enter the load data of the workpiece and confirm with **Next**.
8. Press **Save**.

3.13 Exercise: Calibrating an external tool and robot-guided workpiece

Aim of the exercise

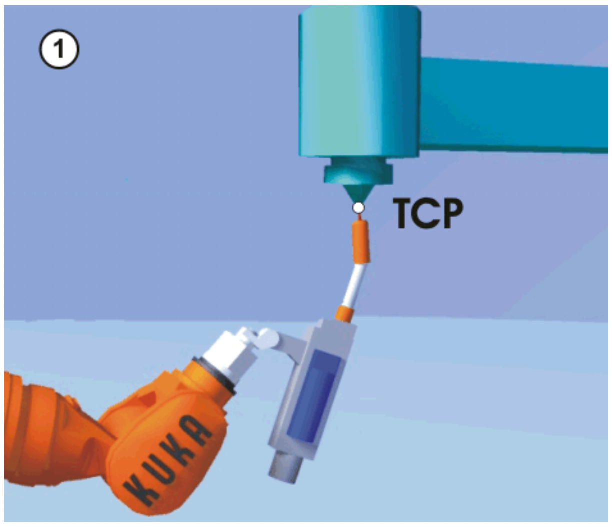
On successful completion of this exercise, you will be able to carry out the following activities:

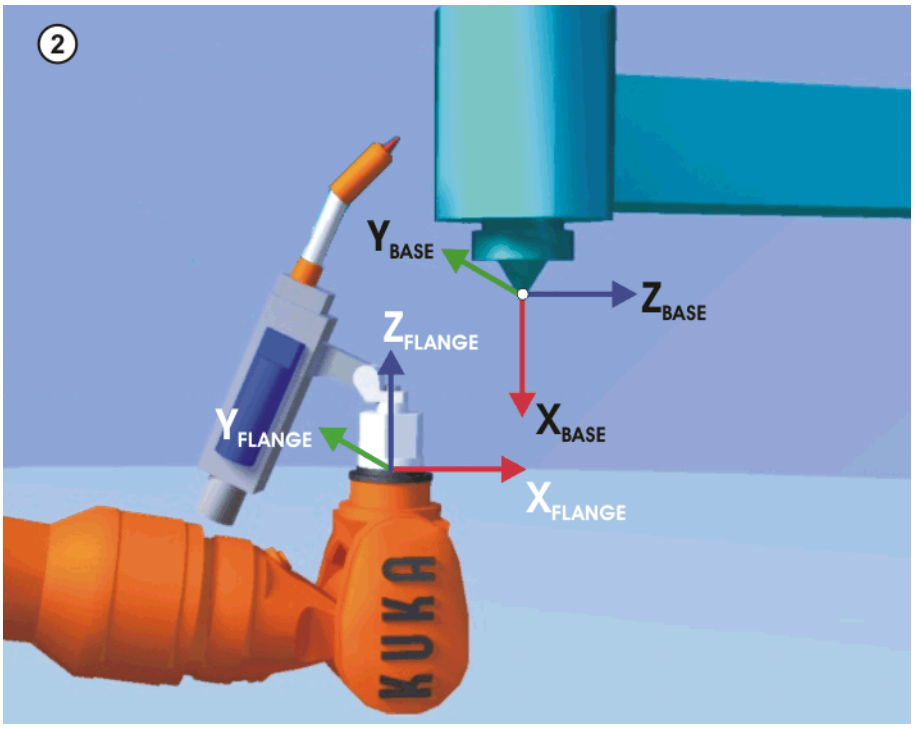
- Calibrate fixed tools
- Calibrate movable workpieces
- Carry out jogging with an external tool

Preconditions

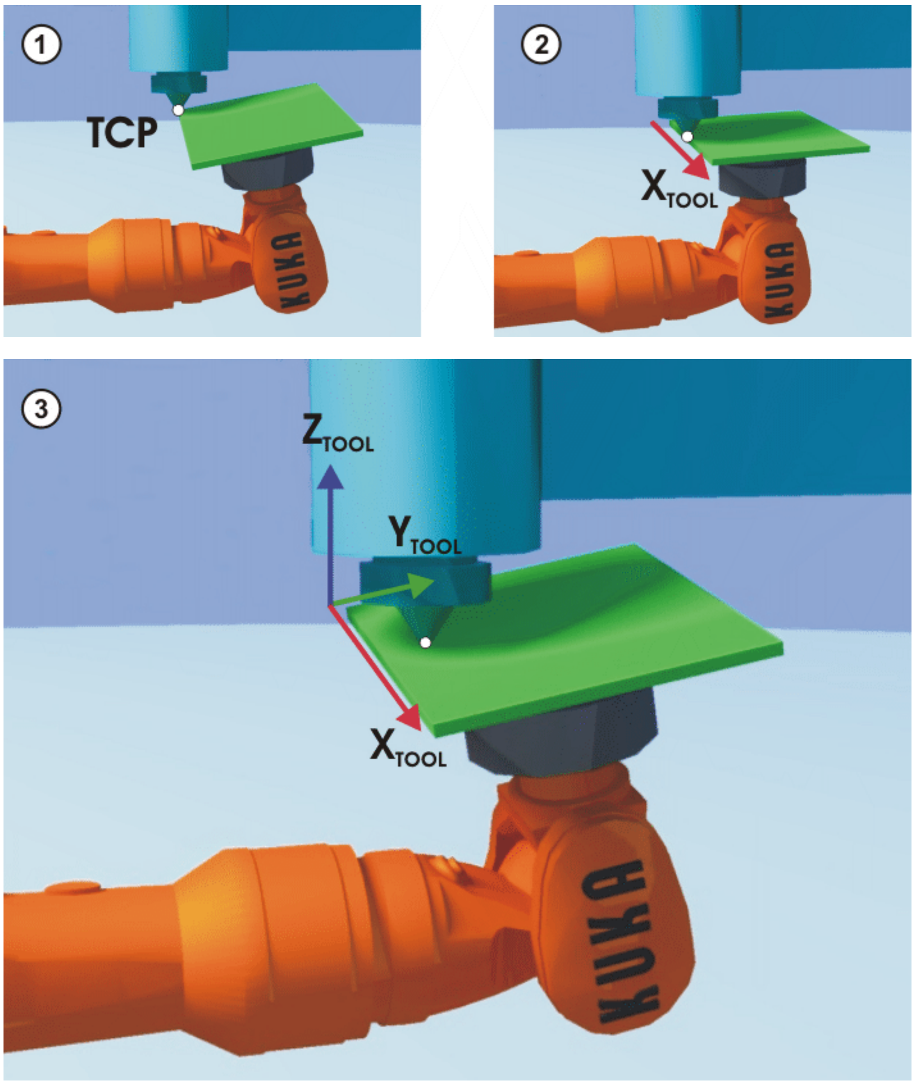
The following are preconditions for successful completion of this exercise:

- Theoretical knowledge of the methods for calibrating fixed tools





- Theoretical knowledge of workpiece calibration with fixed tools, especially the direct method



Task description

Carry out the following tasks: calibrate nozzle and panel

1. For the calibration of the fixed tool, pen1 that is already calibrated (tool number 2) is to be used as a reference tool. Assign the **tool number 10** and the name **Nozzle** for the fixed tool.
   - Be sure to save your data each time you carry out calibration!
2. Calibrate the workpiece guided by the robot. Assign the **workpiece number 12** and the name **Panel**.
   - Enter the load data.

     Load data for the gripper with the panel:

Mass:

M = 8.54 kg

Center of mass:

X= 46 mm	Y= 93 mm	Z= 5 mm

Orientation:

A = 0°	B = 0°	C = 0°

Moments of inertia:

J_X = 0.3 kgm^2	J_Y = 0.5 kgm^2	J_Z = 0.6 kgm^2

3. Once calibration is complete, activate the external tool for jogging. Make appropriate use of the Base and Tool coordinate systems to move the robot.
4. Move the TCP to the BASE coordinate origin of the tool being measured, causing the actual position to be displayed in “Cartesian” mode.

   Actual position:

X	Y	Z	A	B	C
....	....	....	....	....	....

Questions on the exercise

1. How is a base calibrated for a workpiece mounted on the robot flange?

. . . .

. . . .

. . . .

2. How is the TCP of an external tool determined?

. . . .

. . . .

. . . .

3. Why do you need an external TCP?

. . . .

. . . .

. . . .

4. What settings are required to move in the tool direction with an external TCP?

. . . .

. . . .

. . . .

3.14 Disconnecting the smartPAD

Description of smartPAD disconnection

- The smartPAD can be disconnected while the robot controller is running.
- The connected smartPAD assumes the current operating mode of the robot controller.
- A smartPAD can be connected at any time.
- When connecting a smartPAD, it must be ensured that it is the same variant (firmware version) as the device that was disconnected.
- The EMERGENCY STOP and enabling switches are not operational again until 30 s after connection.
- The smartHMI (user interface) is automatically displayed again (this takes no longer than 15 s).

smartPAD disconnection function

⚠ WARNING If the smartPAD is disconnected, the system can no longer be switched off by means of the EMERGENCY STOP button on the smartPAD. For this reason, an external EMERGENCY STOP must be connected to the robot controller.

⚠ WARNING The operator must ensure that disconnected smartPADs are immediately removed from the system and stored out of sight and reach of personnel working on the industrial robot. This serves to prevent operational and non-operational EMERGENCY STOP facilities from becoming interchanged.

⚠ WARNING Failure to observe these precautions may result in death to persons, severe physical injuries or considerable damage to property.

⚠ WARNING The user connecting a smartPAD to the robot controller must subsequently stay with the smartPAD for at least 30 s, i.e. until the EMERGENCY STOP and enabling switches are operational once again. This prevents another user from trying to activate a non-operational EMERGENCY STOP in an emergency situation, for example.

Procedure for disconnecting a smartPAD

Disconnection:

1. Press the disconnect button on the smartPAD.

   A message and a counter are displayed on the smartHMI. The counter runs for 25 s. During this time, the smartPAD can be disconnected from the robot controller.

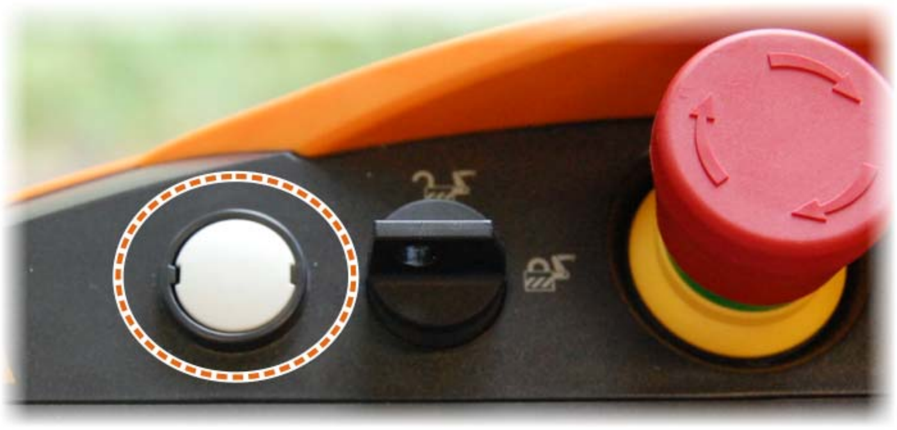


Fig. 3-43: Button for disconnecting the smartPAD

NOTICE If the smartPAD is disconnected without the counter running, this triggers an EMERGENCY STOP. The EMERGENCY STOP can only be canceled by plugging the smartPAD back in.

2. Open the door of the (V)KR C4 control cabinet.
3. Disconnect the smartPAD from the robot controller.

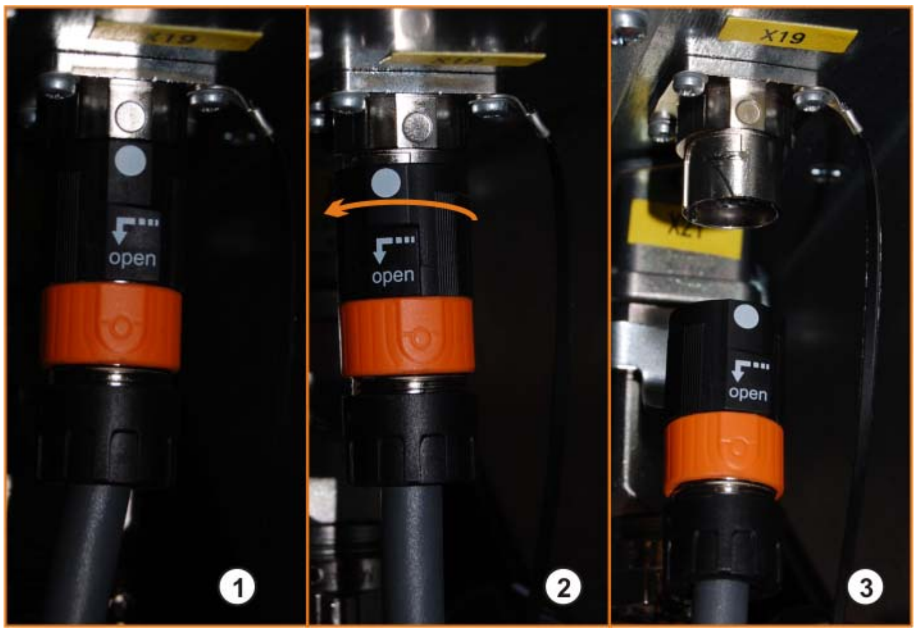


Fig. 3-44: Disconnecting the smartPAD

1	Connector connected
2	Turn the upper black part approx. 25° in the direction indicated by the arrow.
3	Pull the connector downwards.

4. Close the door of the (V)KR C4 control cabinet.

NOTICE If the counter expires without the smartPAD having been disconnected, this has no effect. The disconnect button can be pressed again at any time to display the counter again.

Connection:

1. Ensure that the same smartPAD variant is used again.
2. Open the door of the (V)KR C4 control cabinet.
3. Connect the smartPAD connector.

NOTICE Pay heed to the markings on the female connector and the smartPAD connector.

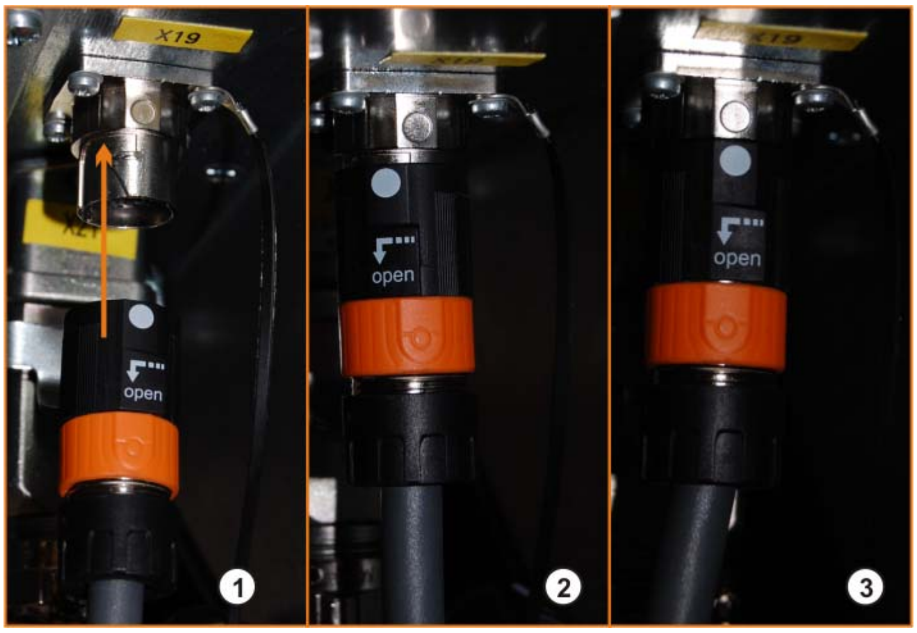


Fig. 3-45: Connecting the smartPAD

1	Connector disconnected (observe marking)
2	Push connector upwards. The upper black part automatically turns approx. 25° while it is being pushed up.
3	The connector automatically locks in place, i.e. the markings are aligned.

⚠ WARNING The user connecting a smartPAD to the robot controller must subsequently stay with the smartPAD for at least 30 s, i.e. until the EMERGENCY STOP and enabling switches are operational once again. This prevents another user from trying to activate a non-operational EMERGENCY STOP in an emergency situation, for example.

4. Close the door of the (V)KR C4 control cabinet.

4 Executing robot programs

4.1 Performing an initialization run

BCO run

The initialization run of a KUKA robot is called a BCO run.

BCO stands for **B**lock **CO**incidence. Coincidence means “coming together” of events in time/space.

A BCO run is carried out in the following cases:

- Program selection (example 1)
- Program reset (example 1)
- Jogging in program mode (example 1)
- Program modifications (example 2)
- Block selection (example 3)

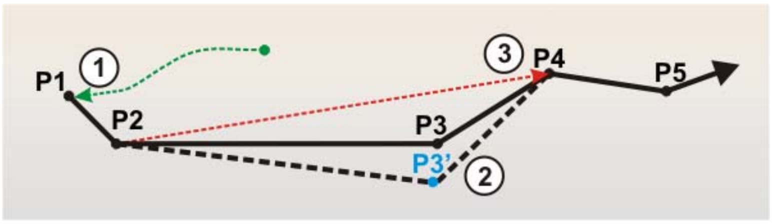


Fig. 4-1: Examples of reasons for a BCO run

Examples for the performance of a BCO run

1. BCO run to the home position following program selection or reset
2. BCO run following modification of a motion command: point deleted, taught, etc.
3. BCO run following block selection

Reasons for a BCO run

A BCO run is necessary to ensure that the current robot position matches the coordinates of the current point in the robot program.

Path planning can only be carried out if the current robot position is the same as a programmed position. The TCP must therefore always be moved onto the path.

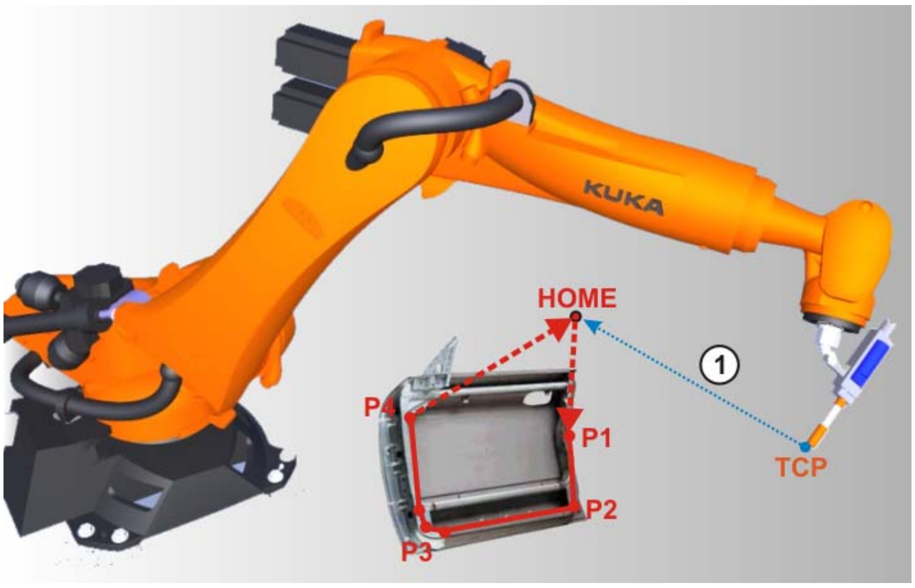


Fig. 4-2: Example of a BCO run

1 BCO run to the home position following program selection or reset

4.2 Selecting and starting robot programs

Selecting and starting robot programs

If a robot program is to be executed, it must be selected. The robot programs are available in the Navigator in the user interface. Motion programs are generally created in folders. The Cell program (management program for controlling the robot from a PLC) is always located in the folder "R1".

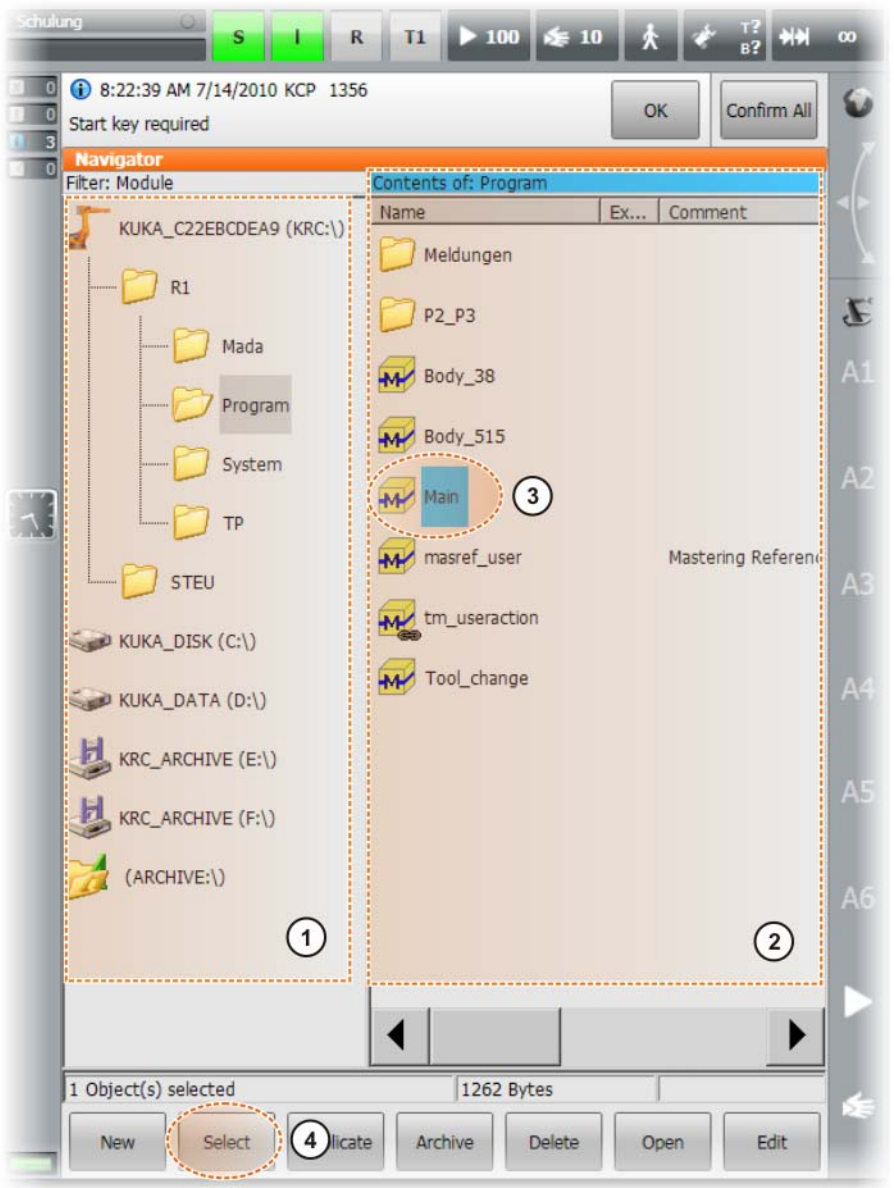


Fig. 4-3: Navigator

1	Navigator: directory/drive structure
2	Navigator: directory/data list
3	Selected program
4	Button for selecting a program

The Start forwards ▷ and Start backwards ◁ keys are available for starting a program.

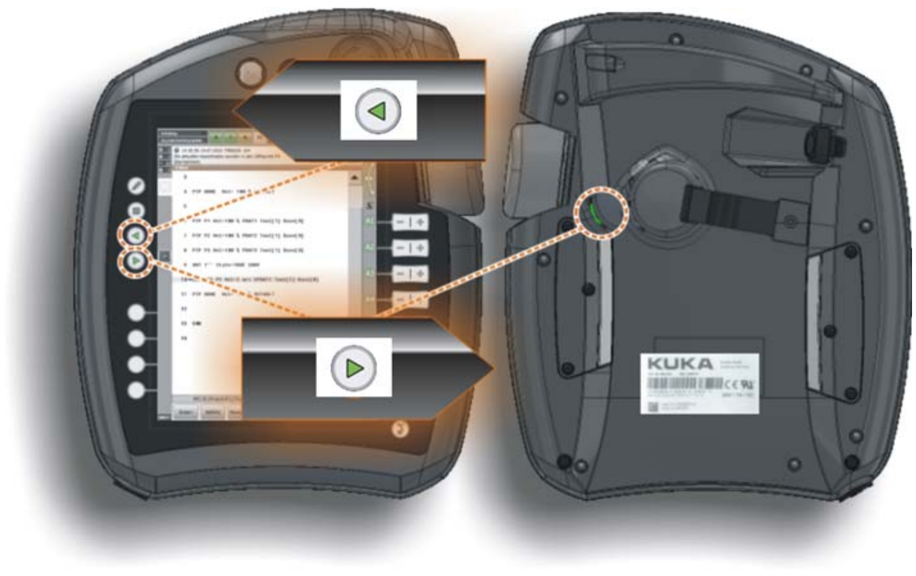
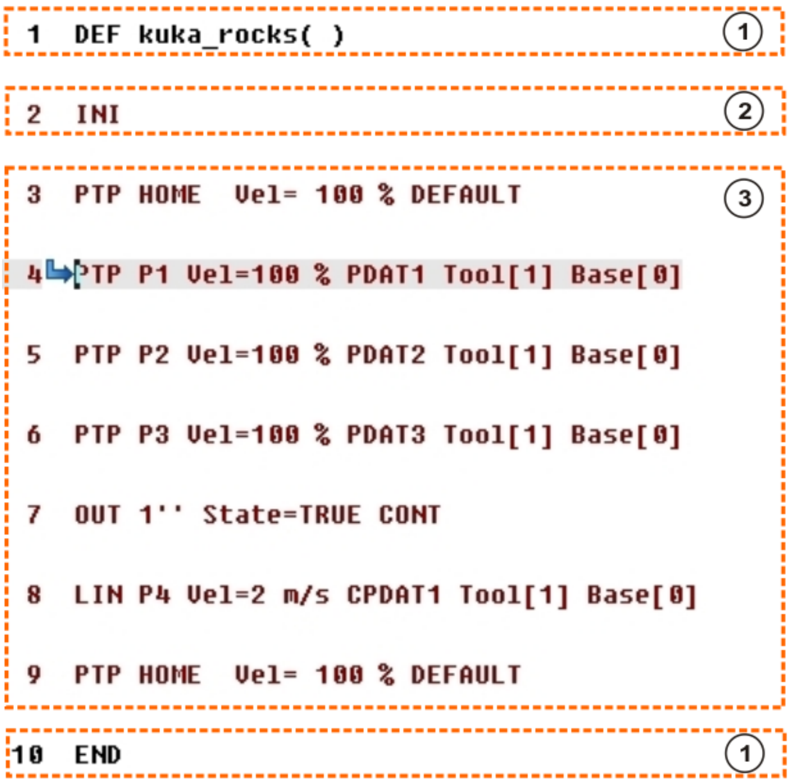


Fig. 4-4: Program execution directions: forwards/backwards

When a program is executed, there are various **program run modes** available for program-controlled robot motion:

	GO ■ Program runs continuously to the end of the program. ■ In test mode, the Start key must be held down.
	MSTEP ■ In the program run mode Motion Step, each motion command is executed separately. ■ At the end of each motion, Start must be pressed again.
	ISTEP | Only available in the user group “Expert”! ■ In Incremental Step mode, the program is executed line by line (irrespective of the contents of the individual lines). ■ The Start key must be pressed again after every line.

What does a robot program look like?



```
DEF kuka_rocks( )

INI

PTP HOME  Vel= 100 % DEFAULT

PTP P1 Vel=100 % PDAT1 Tool[1] Base[0]

PTP P2 Vel=100 % PDAT2 Tool[1] Base[0]

PTP P3 Vel=100 % PDAT3 Tool[1] Base[0]

OUT 1'' State=TRUE CONT

LIN P4 Vel=2 m/s CPDAT1 Tool[1] Base[0]

PTP HOME  Vel= 100 % DEFAULT

END
```

Fig. 4-5: Structure of a robot program

Only visible in the user group “Expert”:

1
- “DEF *Program name()*” always stands at the start of a program.
- “END” defines the end of a program.

2
- The “INI” line contains calls of standard parameters that are required for correct execution of the program.
- The “INI” line must always be executed first!

3
- Actual program text with motion commands, wait commands, logic commands, etc.
- The motion command “PTP Home” is often used at the start and end of a program, as this is a known and clearly defined position.

Program state

Icon	Color	Description
⬜ 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 at the end of the selected program.

Starting a program

Procedure for starting robot programs:

1. Select program.

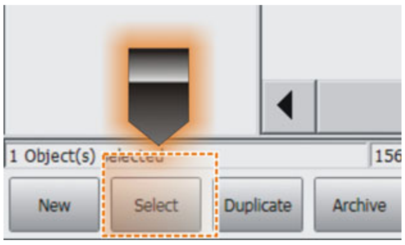


Fig. 4-6: Program selection

2. Set program velocity (program override, POV).

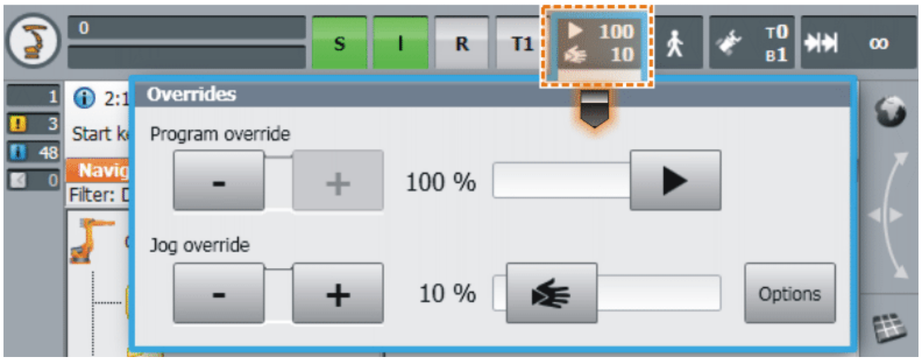


Fig. 4-7: POV setting

3. Press enabling switch.

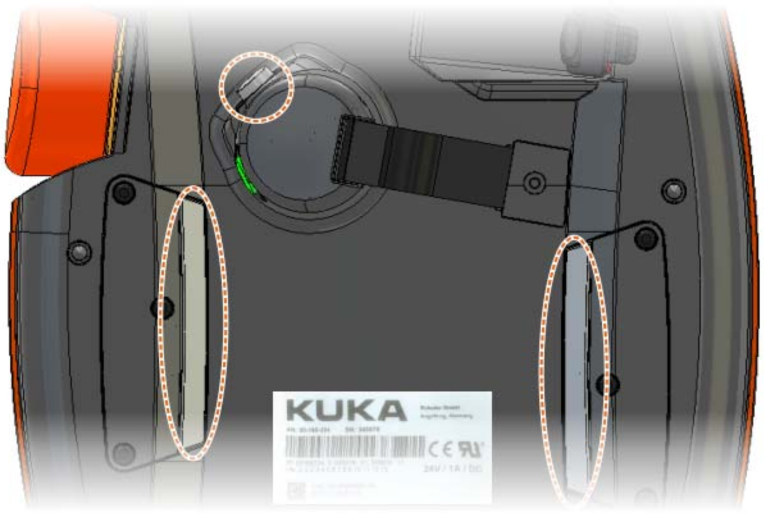


Fig. 4-8: Enabling switches

4. Press and hold down the Start (+) key.
   - The “INI” line is executed.
   - The robot performs the BCO run.

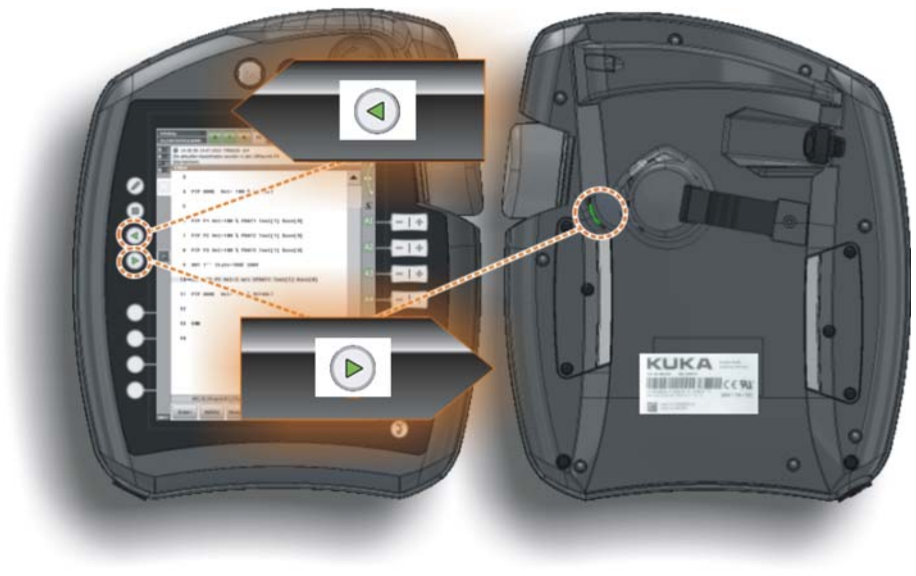
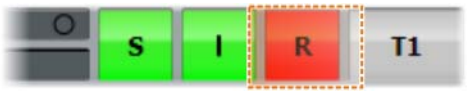


Fig. 4-9: Program execution directions: forwards/backwards

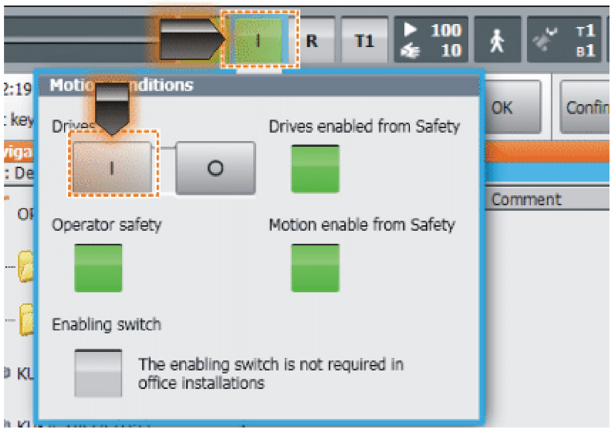
⚠ WARNING A BCO run is executed as a PTP motion from the actual position to the target position if the selected motion block contains the motion command PTP. If the selected motion block contains LIN or CIRC, the BCO run is executed as a LIN motion. Observe the motion to avoid collisions. The velocity is automatically reduced during the BCO run.

5. Once the end position has been reached, the motion is stopped.



The notification message “*Programmed path reached (BCO)*” is displayed.

6. Continued sequence (depending on what operating mode is set):
   - **T1 and T2**: Continue the program by pressing the Start key.
   - **AUT**: Activate drives.



The enabling switch is not required in office installations

Then start the program by pressing *Start*.
   - In the Cell program, switch the operating mode to **EXT** and transfer the motion command from the PLC.

4.3 Exercise: Executing robot programs

Aim of the exercise

On successful completion of this exercise, you will be able to carry out the following activities:

- Select and deselect programs

- Run, stop and reset programs in the required operating modes (test program execution)
- Perform and understand block selection
- Carry out a BCO run

Preconditions

The following are preconditions for successful completion of this exercise:

- Theoretical knowledge of how to use the Navigator
- Knowledge of selecting and canceling programs

Task description

1. Select the module “Air”.

Danger!
The safety regulations contained in the safety instruction must be observed!

2. Test the program in the different operating modes as follows:
   - T1 with 100%
   - T2 with 10%, 30%, 50%, 75%, 100%
   - Automatic with 100%
3. Test the program in the program run modes Go and MSTEP.

5 Working with program files

5.1 Creating program modules

Program modules in Navigator

Program modules should always be stored in the folder “Program”. It is also possible to create new folders and save program modules there. Modules are identified by the icon with the letter “M”. A module can be provided with a comment. A comment may contain a brief description of the functions of the program, for example.

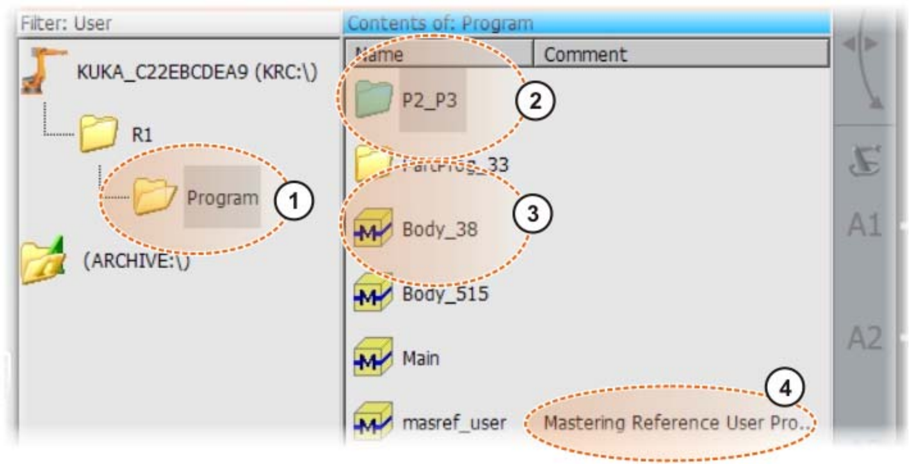


Fig. 5-1: Modules in Navigator

1 Main folder for programs: “Program”
2 Subfolder for additional programs
3 Program module/module
4 Comment of a program module

Properties of program modules

A module always consists of two parts:

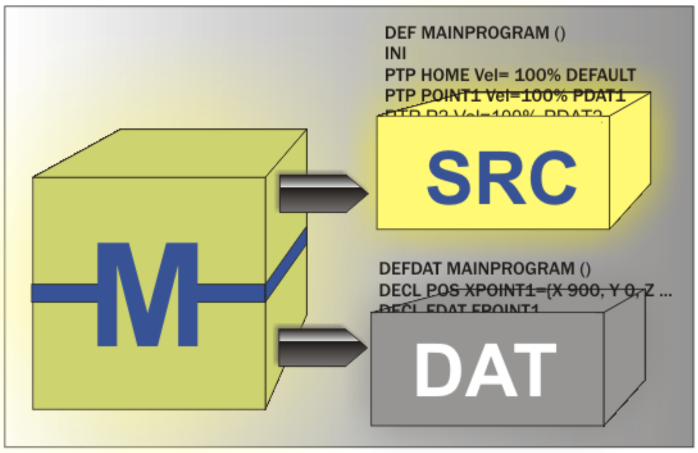

DEF MAINPROGRAM ()
INI
PTP HOME Vel= 100% DEFAULT
PTP POINT1 Vel=100% PDAT1
SRC
M
DEFDAT MAINPROGRAM ()
DECL POS XPOINT1={X 900, Y 0, Z ...
DAT

Fig. 5-2

- **Source code:** The SRC file contains the program code.

```
DEF MAINPROGRAM ()
INI
PTP HOME Vel= 100% DEFAULT
PTP POINT1 Vel=100% PDAT1 TOOL[1] BASE[2]
PTP P2 Vel=100% PDAT2 TOOL[1] BASE[2]
…
END
```

- **Data list:** The DAT file contains permanent data and point coordinates.

```
DEFDAT MAINPROGRAM ()
DECL E6POS XPOINT1={X 900, Y 0, Z 800, A 0, B 0, C 0, S 6, T 27, E1
0, E2 0, E3 0, E4 0, E5 0, E6 0}
DECL FDAT FPOINT1 …
…
ENDDAT
```

Procedure for creating program modules

1. In the directory structure, select the folder in which the program is to be created, e.g. the folder **Program** and then switch to the file list.
2. Press the **New** softkey.
3. Enter a name for the program, and a comment if desired, and confirm it with **OK**.

5.2 Editing program modules

Editing options

Just like in other commonly-used file systems, program modules can also be edited in the Navigator of the KUKA smartPad.

Editing tasks include:

- Duplicate/Copy
- Delete
- Rename

Procedure for deleting a program

1. In the directory structure, select the folder in which the file is located.
2. Select the file in the file list.
3. Press the **Delete** softkey.
4. Confirm the request for confirmation with **Yes**. The module is deleted.

In the user group “Expert” with the filter setting “Detail”, two files are displayed in the Navigator for each module (SRC and DAT file). If this is the case, both files must be deleted! Deleted files cannot be restored!

Procedure for renaming a program

1. In the directory structure, select the folder in which the file is located.
2. Select the file in the file list.
3. Select the softkey **Program** > **Rename**.
4. Overwrite the file name with the new name and confirm with **OK**.

In the user group “Expert” with the filter setting “Detail”, two files are displayed in the Navigator for each module (SRC and DAT file). If this is the case, both files must be renamed!

Procedure for duplicating a program

1. In the directory structure, select the folder in which the file is located.
2. Select the file in the file list.
3. Press the **Duplicate** softkey.
4. Give the new module a new file name and confirm it with **OK**.

In the user group “Expert” with the filter setting “Detail”, two files are displayed in the Navigator for each module (SRC and DAT file). If this is the case, both files must be duplicated!

5.3 Archiving and restoring robot programs

Archiving options

Every archiving operation generates a ZIP file on the corresponding target medium with the same name as the robot. The name of the individual file can be modified under **Robot data**.

File paths: There are three different file paths available:

- **USB (KCP)** | USB stick on KCP (smartPAD)
- **USB (cabinet)** | USB stick on robot control cabinet
- **Network** | Archiving to a network path

  The desired network path must be configured under **Robot data**.

Parallel to the ZIP file generated on the selected storage medium during every archiving operation, an additional archive file (INTERN.ZIP) is stored on drive D:\.

Data: The following selections can be made for archiving data:

- **All:**

  The data that are required to restore an existing system are archived.
- **Applications:**

  All user-defined KRL modules (programs) and their corresponding system files are archived.
- **Machine data:**

  The machine data are archived.
- **Log data:**

  The log files are archived.
- **KrcDiag:**

  Archiving of data for fault analysis by KUKA Roboter GmbH. A folder is generated here (name **KRCDiag**) in which up to ten ZIP files can be written. Parallel to this, archiving is carried out on the controller under C:\KUKA\KRCDiag.

Restoring data

WARNING Generally, only archives with the right software version may be loaded. If other archives are loaded, the following may occur:

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

The following menu items are available for restoring data:

- **All**
- **Applications**
- **Configuration**

The system generates an error message in the following cases:

- If the archived data have a different version from those in the system.
- If the version of the technology packages does not match the installed version.

Procedure for archiving

NOTICE Only the KUKA.USB data stick may be used. Data may be lost or modified if any other USB stick is used.

1. Select the menu sequence **File** > **Archive** > **USB (KCP)** or **USB (cabinet)** and the desired menu item.
2. Confirm the request for confirmation with **Yes**.

   Once the archiving is completed, this is indicated in the message window.
3. The stick can be removed when the LED on the stick is no longer lit.

Procedure for restoration

1. Select the menu sequence **File** > **Restore** and then the desired subitems.
2. Confirm the request for confirmation with **Yes**. Archived files are restored to the robot controller. A message indicates completion of the restoration process.
3. If data have been restored from a USB stick: remove the USB storage medium.

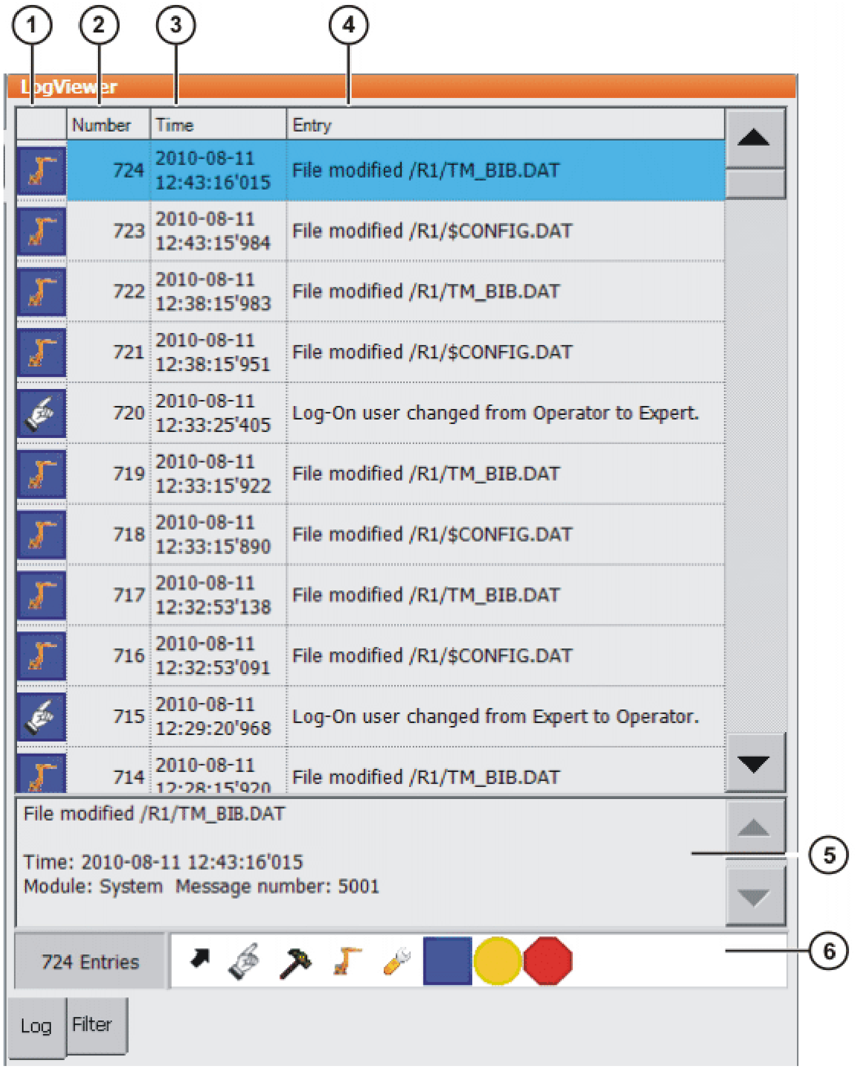
NOTICE In the case of restoring data from a USB medium: the medium must not be removed until the LED on the USB medium is no longer lit. Otherwise, the medium could be damaged.

4. Reboot the robot controller.

5.4 Tracking program modifications and changes of state by means of the logbook

Logging options

The operator actions on the smartPAD are automatically logged. The command **Logbook** displays the logbook.



	Number	Time	Entry
	724	2010-08-11 12:43:16'015	File modified /R1/TM_BIB.DAT
	723	2010-08-11 12:43:15'984	File modified /R1/$CONFIG.DAT
	722	2010-08-11 12:38:15'983	File modified /R1/TM_BIB.DAT
	721	2010-08-11 12:38:15'951	File modified /R1/$CONFIG.DAT
	720	2010-08-11 12:33:25'405	Log-On user changed from Operator to Expert.
	719	2010-08-11 12:33:15'922	File modified /R1/TM_BIB.DAT
	718	2010-08-11 12:33:15'890	File modified /R1/$CONFIG.DAT
	717	2010-08-11 12:32:53'138	File modified /R1/TM_BIB.DAT
	716	2010-08-11 12:32:53'091	File modified /R1/$CONFIG.DAT
	715	2010-08-11 12:29:20'968	Log-On user changed from Expert to Operator.
	714	2010-08-11 12:28:15'920	File modified /R1/TM_BIB.DAT

```
File modified /R1/TM_BIB.DAT

Time: 2010-08-11 12:43:16'015
Module: System  Message number: 5001
```

724 Entries

Fig. 5-3: Logbook, Log tab

Item	Description
1	Type of log event The individual filter types and filter classes are listed on the **Filter** tab.
2	Log event number
3	Date and time of the log event
4	Brief description of the log event
5	Detailed description of the selected log event
6	Indication of the active filter

Filtering log events

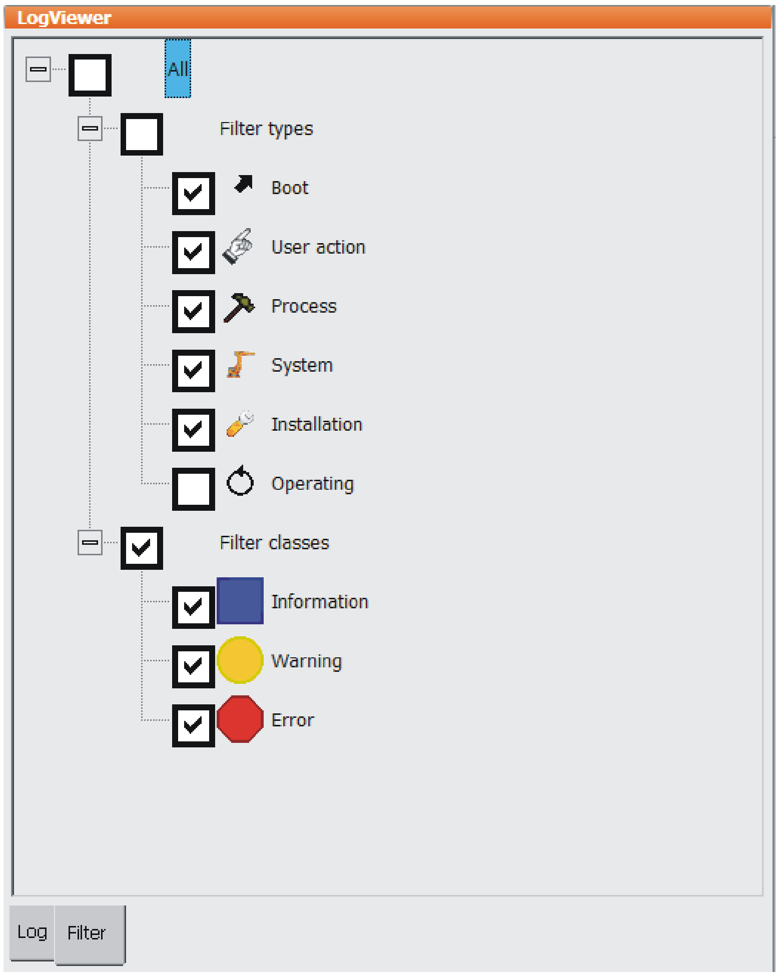


Fig. 5-4: Logbook, Filter tab

Using the logbook function

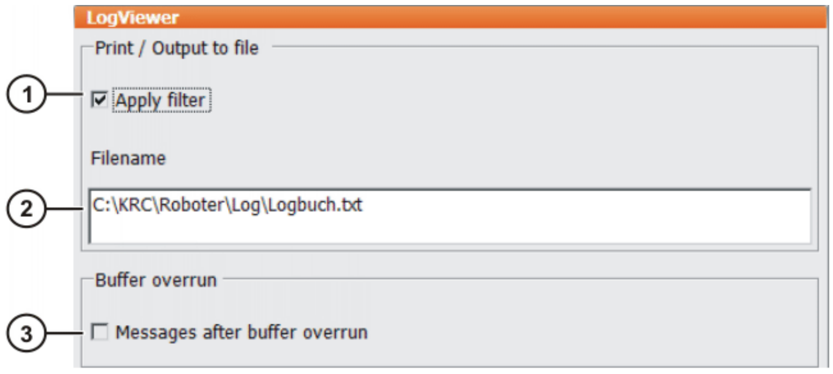
Viewing and configuration can be carried out in any user group.

Displaying the logbook:

- In the main menu, select **Diagnosis** > **Logbook** > **Display**.

Configuring the logbook:

1. In the main menu, select **Diagnosis** > **Logbook** > **Configuration**.
2. Make the relevant settings:
   - Add/remove filter types
   - Add/remove filter classes
3. Press **OK** to save the configuration and close the window.



☑ Apply filter

Filename

C:\KRC\Roboter\Log\Logbuch.txt

☐ Messages after buffer overrun

Fig. 5-5: Logbook configuration window

1 Apply filter settings for the display. If the check box is not activated, the display is unfiltered.

2 Path for the text file.

3 Log data deleted because of a buffer overflow are indicated in gray in the text file.

6 Creating and modifying programmed motions

6.1 Creating new motion commands

Programming robot motions

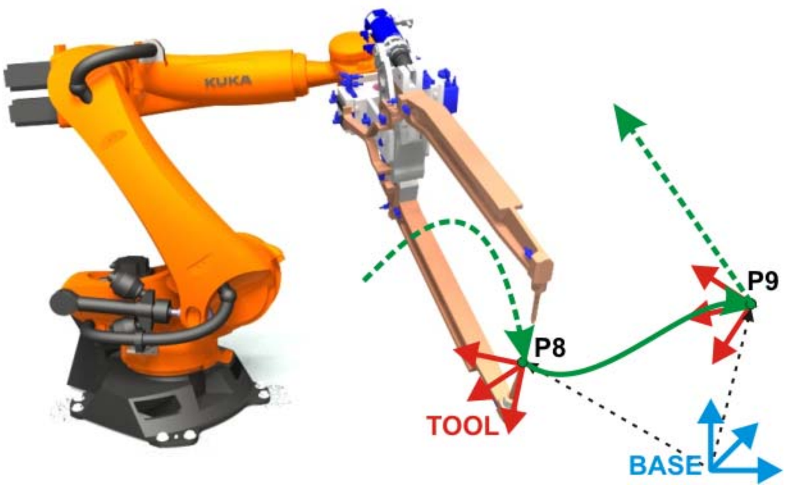
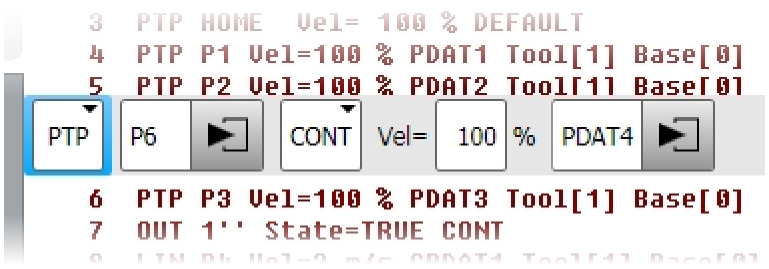


Fig. 6-1: Robot motion

When robot motions have to be programmed, many questions are raised:

Question	Solution	Keyword
How does the robot remember its positions?	The positions of the tool in space are saved (robot position in accordance with the tool and base that are set).	POS
How does the robot know how to move?	From the specification of the motion type: point-to-point, linear or circular.	PTP LIN CIRC
How fast does the robot move?	The velocity between two points and the acceleration are specified during programming.	Vel. Acc.
Does the robot have to stop at every point?	To save cycle time, points can also be approximated; no exact positioning is carried out in this case.	CONT
What orientation does the tool adopt when a point is reached?	The orientation control can be set individually for each motion.	ORI_TYPE
Does the robot recognize obstacles?	No, the robot “stubbornly” follows its programmed path. The programmer is responsible for ensuring that there is no risk of collisions. There is also a collision monitoring function, however, for protecting the machine.	Collision detection

This information must be transferred when programming robot motions using the teaching method. Inline forms, into which the information can easily be entered, are used for this.



```
PTP HOME  Vel= 100 % DEFAULT
PTP P1 Vel=100 % PDAT1 Tool[1] Base[0]
PTP P2 Vel=100 % PDAT2 Tool[1] Base[0]
PTP  P6  CONT  Vel= 100 %  PDAT4
PTP P3 Vel=100 % PDAT3 Tool[1] Base[0]
OUT 1'' State=TRUE CONT
LIN P4 Vel=2 m/s CPDAT1 Tool[1] Base[0]
```

Fig. 6-2: Inline form for motion programming

Motion types

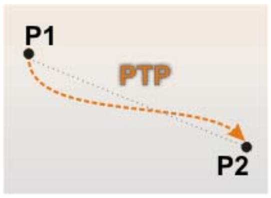
Various motion types are available for programming motion commands. Motions can be programmed in accordance with the specific requirements of the robot's work process.

- Axis-specific motions (PTP: point-to-point)
- CP motions: LIN (linear) and CIRC (circular)
- SPLINE: Spline is a motion type that is suitable for particularly complex, curved paths. Such paths can generally also be generated using LIN and CIRC motions, but Spline nonetheless has advantages.

Spline motions are not covered by this training documentation. More detailed information can be found in the *Operating and Programming Instructions for KUKA System Software 8.2*.

6.2 Creating cycle-time optimized motion (axis motion)

PTP

Motion type	Meaning	Application example
	*Point-to-point*: ■ Axis-specific 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. ■ The leading axis is the axis that takes longest to reach the end point. ■ SYNCHRO PTP: All axes start together and also stop in a synchronized manner. ■ The first motion in the program must be a PTP motion, as Status and Turn are evaluated only here.	Point applications, e.g.: ■ Spot welding ■ Transfer ■ Measuring, inspection Auxiliary positions: ■ Intermediate points ■ Free points in space

Approximate positioning

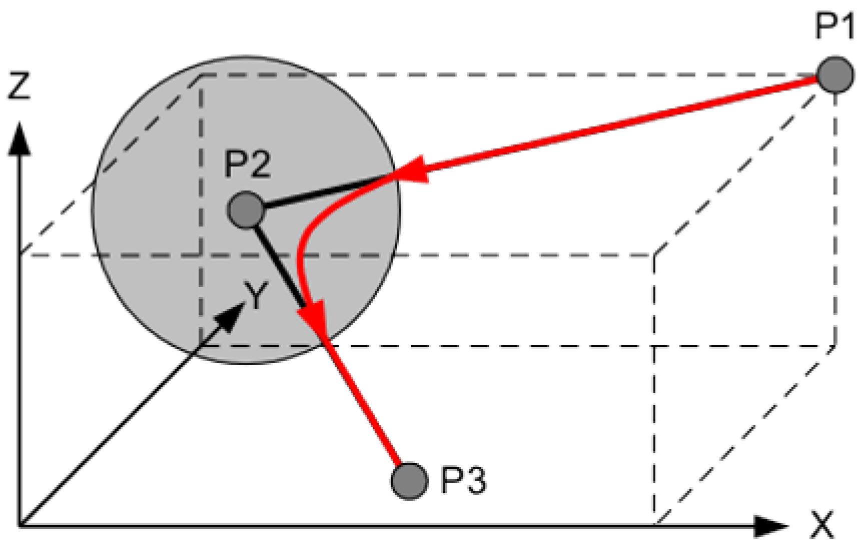


Fig. 6-3: Approximating a point

In order to accelerate the motion sequence, the controller is able to approximate motion commands labeled with CONT. Approximate positioning means that the point coordinates are not addressed exactly. The robot leaves the path of the exact positioning contour before it reaches them. The TCP is guided along an approximate positioning contour that leads into the exact positioning contour of the next motion command.

Advantages of approximate positioning

- Reduced wear
- Shorter cycle times

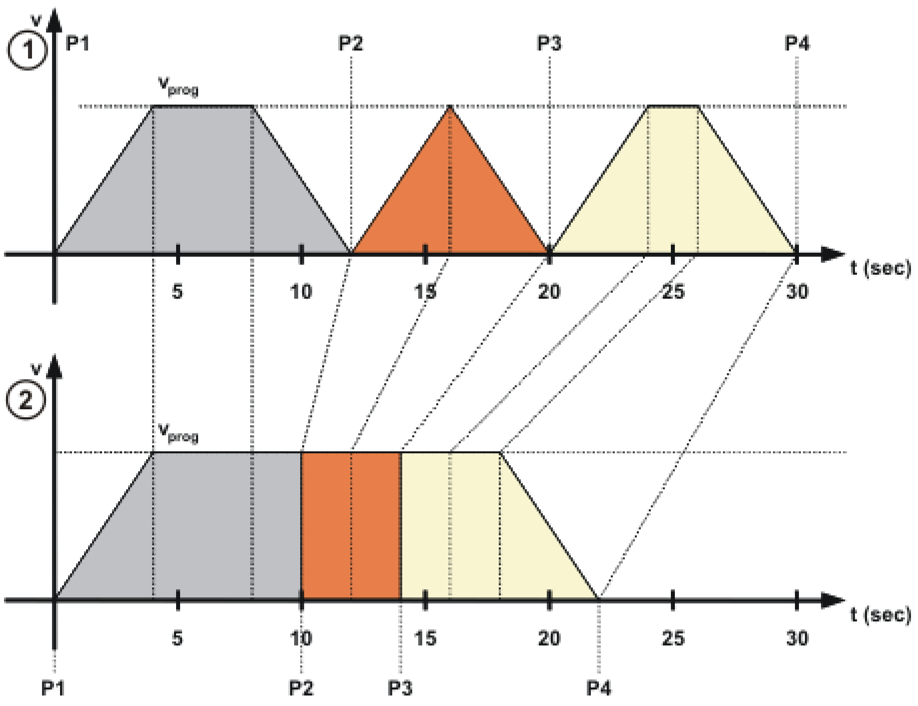
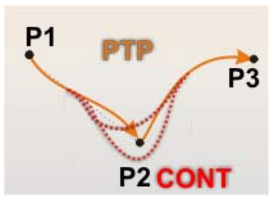


Fig. 6-4: Comparison of exact positioning and approximate positioning

In order to be able to perform an approximate positioning motion, the controller must be able to load the following motion commands. This is carried out by the computer advance run.

Approximate positioning in the PTP motion type

Motion type	Feature	Approximation distance
	■ The approximate positioning contour cannot be predicted!	Specified in %

Procedure for creating PTP motions

Preconditions

- T1 mode is set.
- A robot program is selected.

1. Move the TCP to the position that is to be taught as the end point.

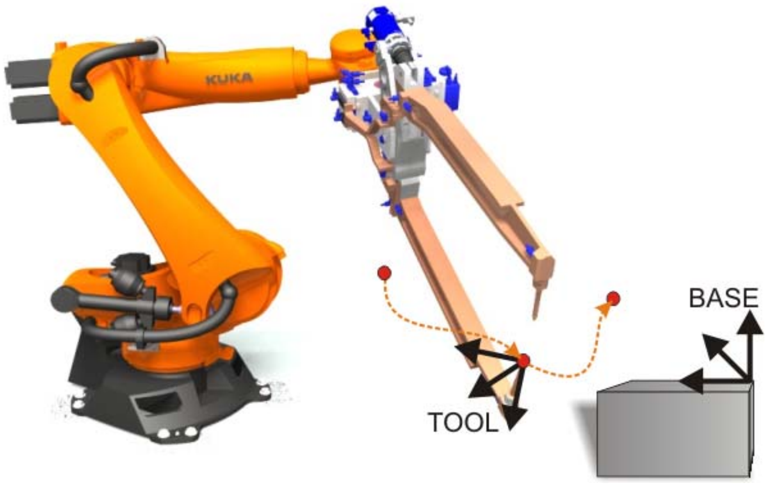


Fig. 6-5: Motion command

2. Position the cursor in the line after which the motion instruction is to be inserted.
3. Menu sequence **Commands** > **Motion** > **PTP**.
   Alternatively, the softkey **Motion** can be pressed in the corresponding line.

   An inline form appears:

   - **Inline form “PTP”**

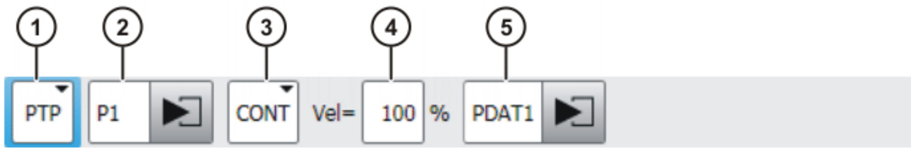
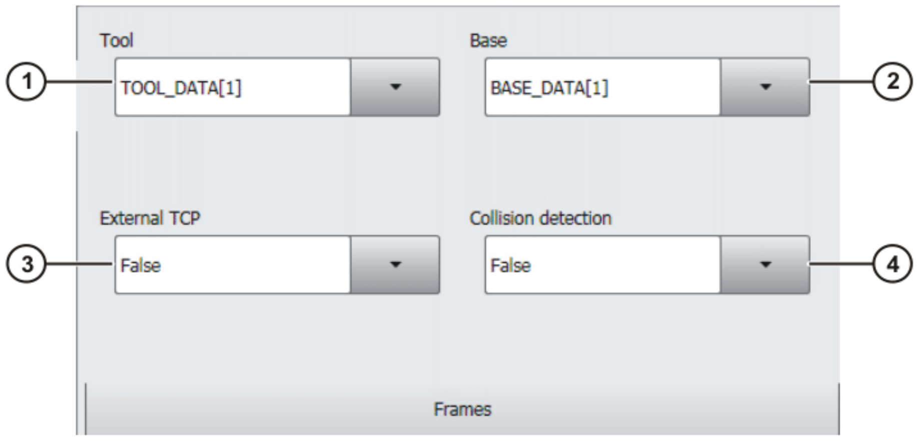


Fig. 6-6: Inline form for PTP motions

4. Enter parameters in the inline form.

Item	Description
1	Motion type **PTP**, **LIN** or **CIRC**
2	The name of the end point is issued automatically, but can be over-written as required. To edit the point data, touch the arrow; the option window **Frames** opens. In the case of **CIRC**, an auxiliary point must be taught in addition to the end point: move to the position of the auxiliary point and press **Teach Aux**.
3	■ **CONT**: end point is approximated. ■ **[Empty box]**: the motion stops exactly at the end point.
4	Velocity ■ PTP motions: **1 … 100%** ■ CP motions: **0.001 … 2 m/s**
5	Motion data set: ■ Acceleration ■ Approximation distance (if CONT is entered in box (3)) ■ Orientation control (only for CP motions)

5. Enter the correct data for the tool and base coordinate system in the option window “Frames”, together with details of the interpolation mode (external TCP: on/off) and the collision monitoring function.



Tool: TOOL_DATA[1]
Base: BASE_DATA[1]
External TCP: False
Collision detection: False

Fig. 6-7: Option window “Frames”

Item	Description
1	Tool selection. If **True** in the box **External TCP**: workpiece selection. Range of values: [1] … [16]
2	Base selection. If **True** in the box **External TCP**: fixed tool selection. Range of values: [1] … [32]

Item	Description
3	Interpolation mode ▪ **False**: The tool is mounted on the mounting flange. ▪ **True**: the tool is a fixed tool.
4	▪ **True**: For this motion, the robot controller calculates the axis torques. These are required for collision detection. ▪ **False**: For this motion, the robot controller does not calculate the axis torques. Collision detection is thus not possible for this motion.

6. The acceleration can be reduced from the maximum value in the option window “Motion parameters”. If approximate positioning has been activated, the approximation distance can also be modified. Depending on the configuration, the distance is set in **mm** or **%**.

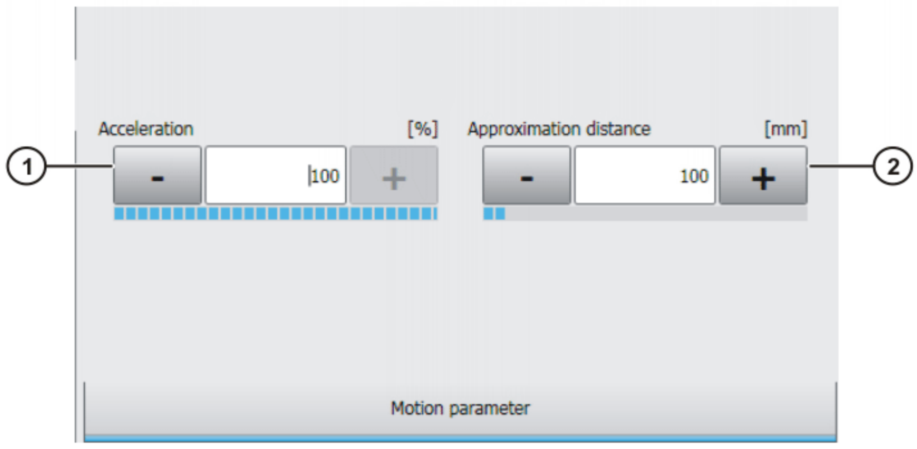


Fig. 6-8: Option window “Motion parameter” (PTP)

Item	Description
1	Acceleration Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. The acceleration applies to the leading axis for this motion block. ▪ **1 … 100%**
2	This box is only displayed if **CONT** was selected in the inline form. Furthest distance before the end point at which approximate positioning can begin. Maximum distance: half the distance between the start point and the end point relative to the contour of the PTP motion without approximate positioning ▪ **1 … 100%** ▪ **1 ... 1000 mm**

7. Save instruction with **Cmd Ok**. The current position of the TCP is taught as the end point.

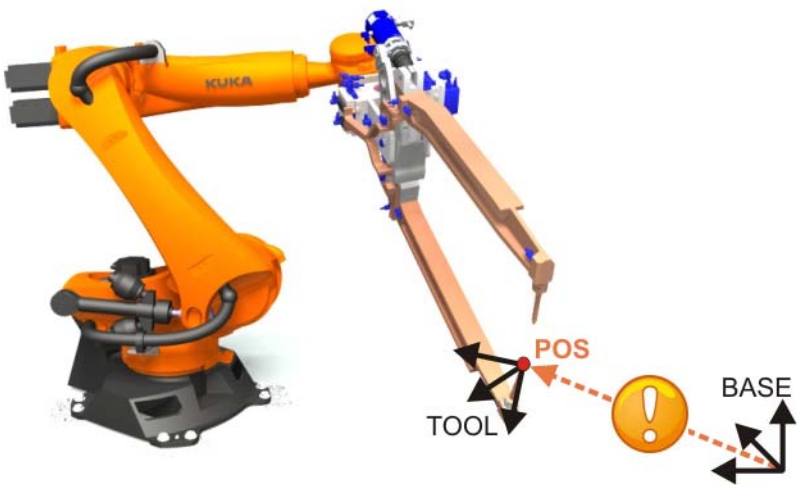


Fig. 6-9: Saving the point coordinates with “Cmd OK” and “Touchup”

6.3 Exercise: Dummy program – program handling and PTP motions

Aim of the exercise

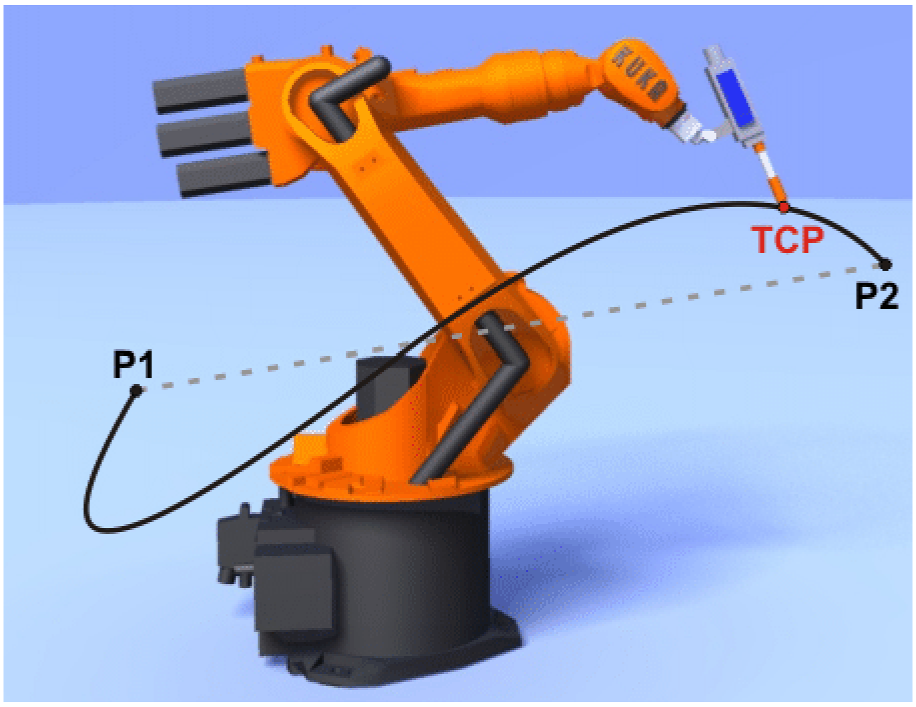
On successful completion of this exercise, you will be able to carry out the following activities:

- Select and deselect programs
- Run, stop and reset programs in the required operating modes (test program execution)
- Delete motion blocks and insert new PTP motions
- Change the program run mode and carry out step-by-step movement to programmed points
- Perform and understand block selection
- Carry out a BCO run

Preconditions

The following are preconditions for successful completion of this exercise:

- Theoretical knowledge of how to use the Navigator
- Theoretical knowledge of the PTP motion type



Task description

Carry out the following tasks: Create and test programs.

1. Create a new module with the name **Air_PROG**.

Danger!
The safety regulations contained in the safety instruction must be observed!

2. Create a sequence of approx. five PTP motion blocks.
3. If the motion is not collision-free, delete the relevant point(s) and create a new one in each case.
4. Test the program in T1 mode at different program velocities (POV).
5. Test the program in T2 mode at different program velocities (POV).
6. Test the program in Automatic mode.

Task, part B

Carry out the following tasks: Program correction

1. Set various velocities for your space points.
2. Call the same point several times in the program.
3. Delete the motion blocks and insert new ones at a different point in the program.
4. Carry out a block selection.
5. Stop your program during testing and use the function **Program start backwards**.
6. Test your program in the modes T1, T2 and Automatic.

Questions on the exercise

1. What is the difference between selecting a program and opening it?

. . . .

. . . .

2. What program run modes are there and what are they used for?

. . . .

. . . .

. . . .

3. What is a BCO run?

. . . .

. . . .

. . . .

4. How can you influence the program velocity?

. . . .

. . . .

. . . .

5. What are the characteristics of a PTP motion?

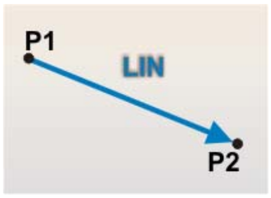
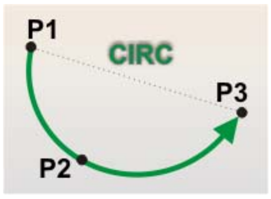
. . . .

. . . .

. . . .

6.4 Creating CP motions

LIN and CIRC

Motion type	Meaning	Application example
	*Linear*: ■ Motion in a straight line: ■ The TCP of the tool is guided from the start point to the end point with constant velocity and a defined orientation. ■ The velocity and orientation refer to the TCP.	Path applications, e.g.: ■ Arc welding ■ Adhesive bonding ■ Laser welding / cutting
	*Circular*: ■ Circular path motion is defined by a start point, auxiliary point and end point. ■ The TCP of the tool is guided from the start point to the end point with constant velocity and a defined orientation. ■ The velocity and orientation refer to the TCP.	Path applications, similar to LIN: ■ Circles, radii, curves

Singularity positions

KUKA robots with 6 degrees of freedom have 3 different singularity positions.

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. This is a mathematical property, not a mechanical one, and thus only exists for CP motions and not axis motions.

Overhead singularity α1

In the overhead singularity, the wrist root point (= center point of axis A5) is located vertically above axis A1 of the robot.

The position of axis A1 cannot be determined unambiguously by means of reverse transformation and can thus take any value.

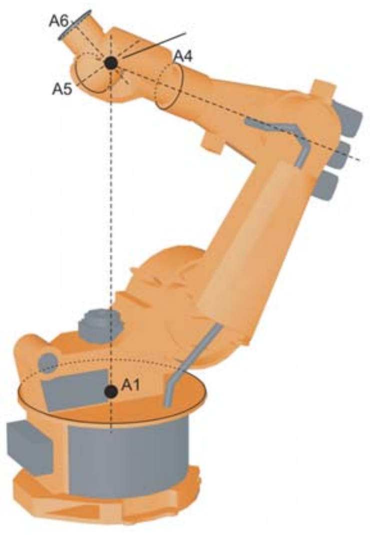


Fig. 6-10: Overhead singularity (α1 position)

Extended position singularity α2

In the extended position, the wrist root point (= center point of axis A5) 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.

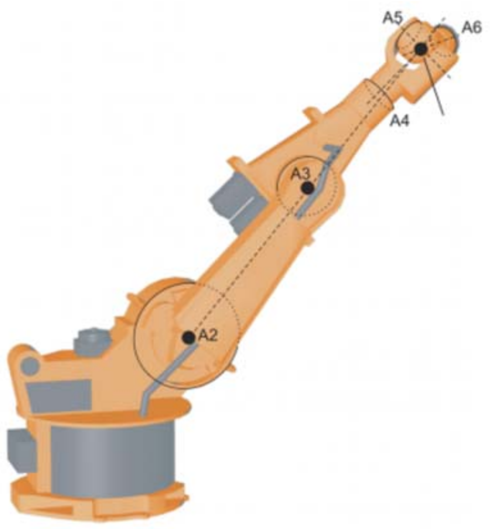


Fig. 6-11: Extended position (α2 position)

Wrist axis singularity α5

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.

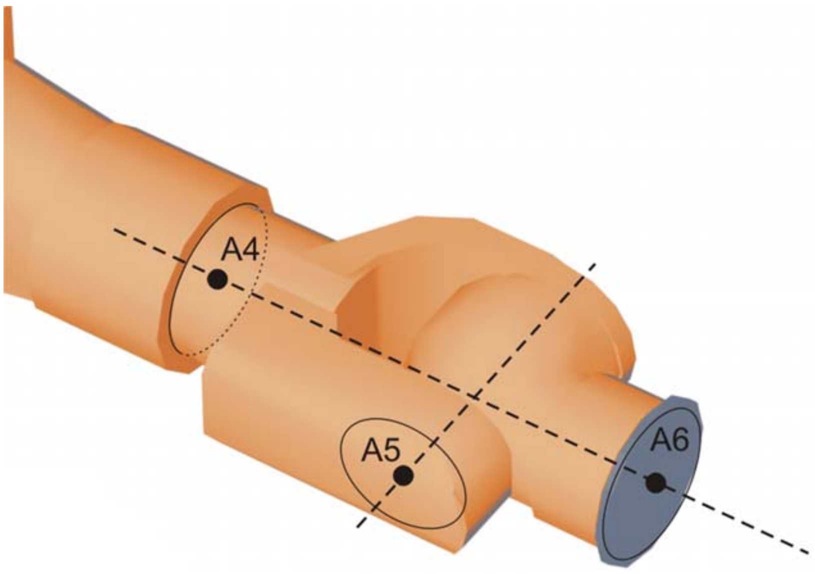


Fig. 6-12: Wrist axis singularity (α5 position)

Orientation control with CP motions

In the case of CP motions, it is possible to define the orientation control precisely. The orientation of a tool can be different at the start point and end point of a motion.

Orientation control with the motion type **LIN**

- **Standard** or **Wrist PTP**

  The orientation of the tool changes continuously during the motion.

  Use Wrist PTP if, with Standard, the robot passes through a wrist axis singularity, as the orientation is carried out by means of linear transformation (axis-specific jogging) of the wrist axis angles.

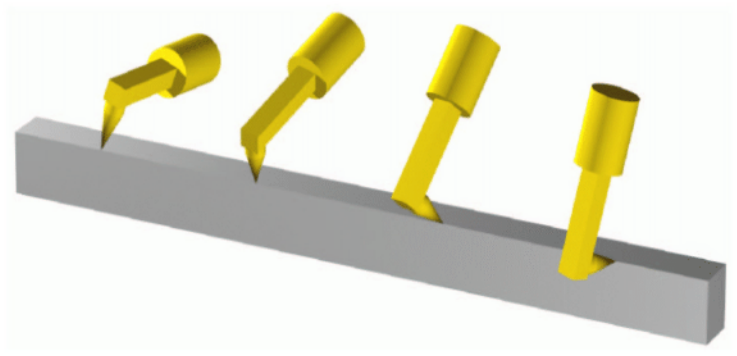


Fig. 6-13: Standard

- **Constant**

  The orientation of the tool remains constant during the motion, i.e. as taught at the start point. The orientation taught at the end point is disregarded.

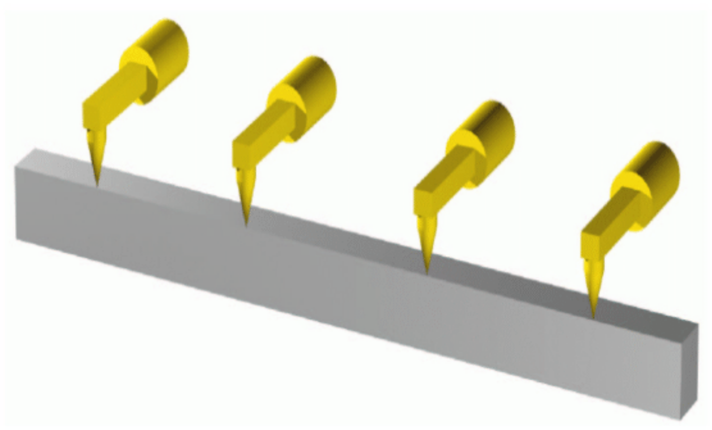


Fig. 6-14: Orientation control - Constant

Orientation control with the motion type **CIRC**

- **Standard** or **Wrist PTP**

  The orientation of the tool changes continuously during the motion.

  Use Wrist PTP if, with Standard, the robot passes through a wrist axis singularity, as the orientation is carried out by means of linear transformation (axis-specific jogging) of the wrist axis angles.

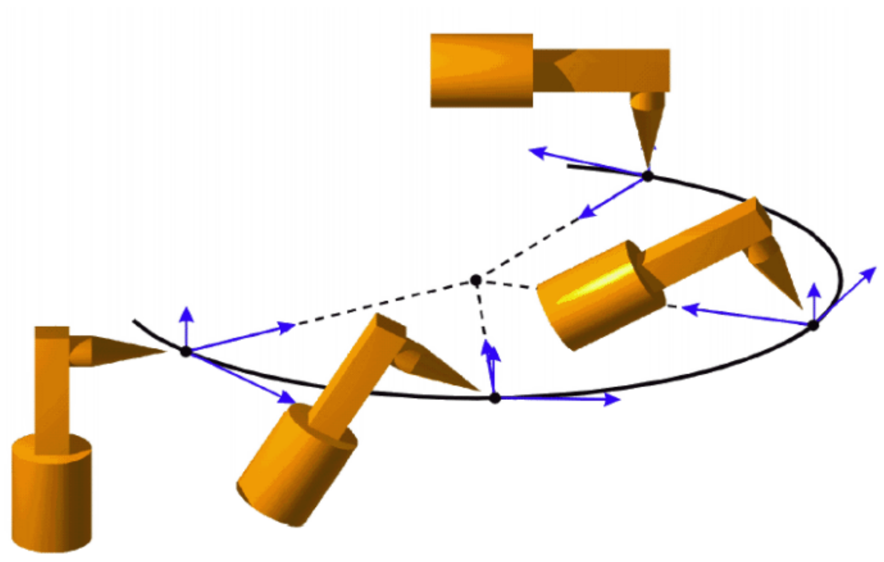


Fig. 6-15: Standard + base-related

- **Constant**

  The orientation of the tool remains constant during the motion, i.e. as taught at the start point. The orientation taught at the end point is disregarded.

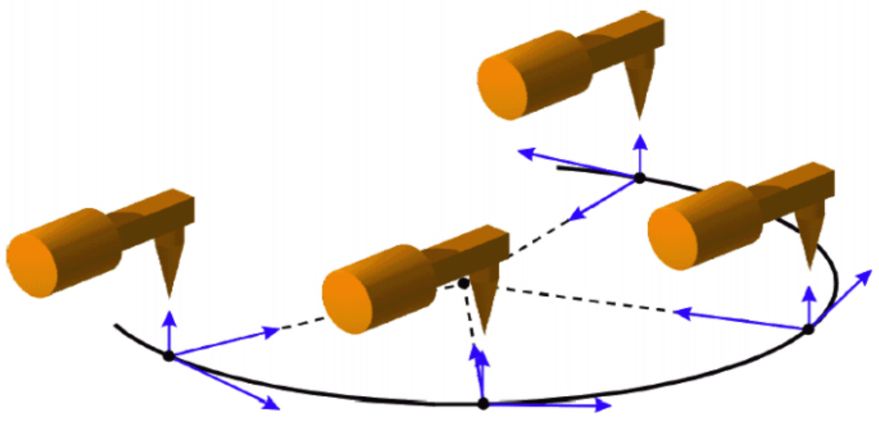
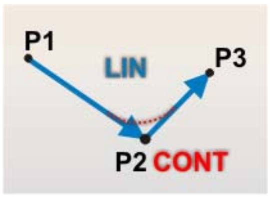
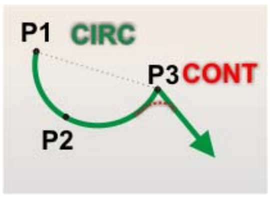


Fig. 6-16: Constant orientation control + base-related

Approximation of CP motions

> The approximate positioning function is not suitable for generating circular motions. It is purely for preventing an exact stop at the point.

Approximate positioning in the motion types LIN and CIRC

Motion type	Feature	Approximation distance
	■ Path corresponds to two parabolic branches	Specified in mm
	■ Path corresponds to two parabolic branches	Specified in mm

Procedure for the creation of LIN and CIRC motions

Preconditions

- T1 mode is set.
- A robot program is selected.

1. Move the TCP to the position that is to be taught as the end point.

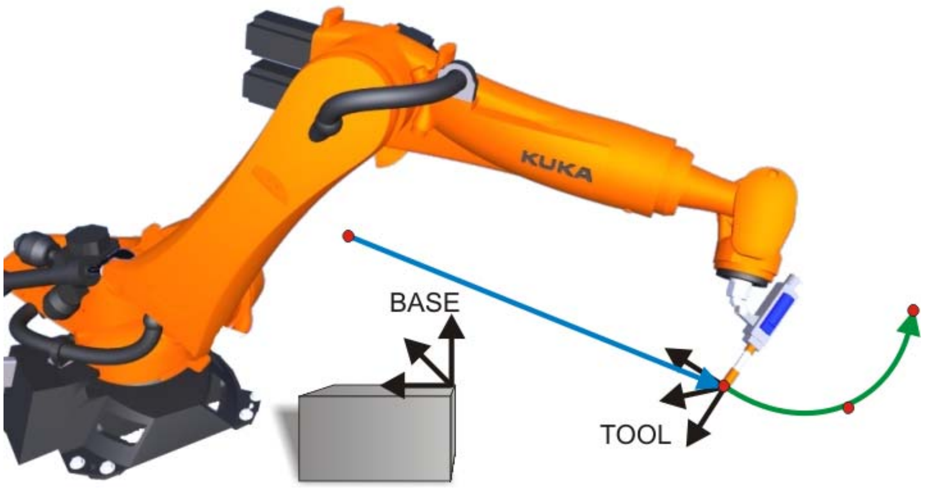


Fig. 6-17: Motion command with LIN and CIRC

2. Position the cursor in the line after which the motion instruction is to be inserted.
3. Select the menu sequence **Commands** > **Motion** > **LIN** or **CIRC**.

   Alternatively, the softkey **Motion** can be pressed in the corresponding line.

   An inline form appears:

   - **Inline form "LIN"**

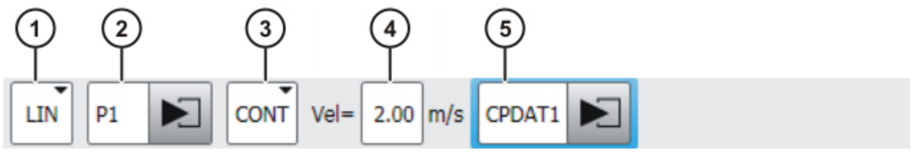


Fig. 6-18: Inline form for LIN motions

   - **Inline form "CIRC"**

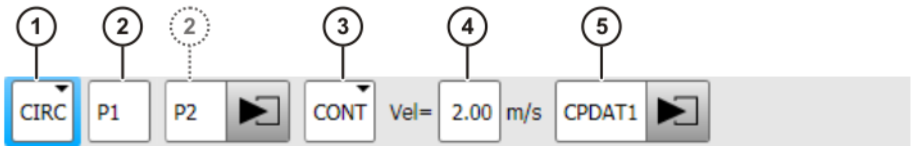


Fig. 6-19: Inline form for CIRC motions

4. Enter parameters in the inline form.

Item	Description
1	Motion type **PTP**, **LIN** or **CIRC**
2	The name of the end point is issued automatically, but can be over-written as required. To edit the point data, touch the arrow; the option window **Frames** opens. In the case of **CIRC**, an auxiliary point must be taught in addition to the end point: move to the position of the auxiliary point and press **Teach Aux**. The orientation of the tool at the auxiliary point is irrele-vant.
3	■ **CONT**: end point is approximated. ■ **[Empty box]**: the motion stops exactly at the end point.
4	Velocity ■ PTP motions: **1 … 100%** ■ CP motions: **0.001 … 2 m/s**
5	Motion data set: ■ Acceleration ■ Approximation distance (if CONT is entered in box (3)) ■ Orientation control (only for CP motions)

5. Enter the correct data for the tool and base coordinate system in the option window “Frames”, together with details of the interpolation mode (external TCP: on/off) and the collision monitoring function.

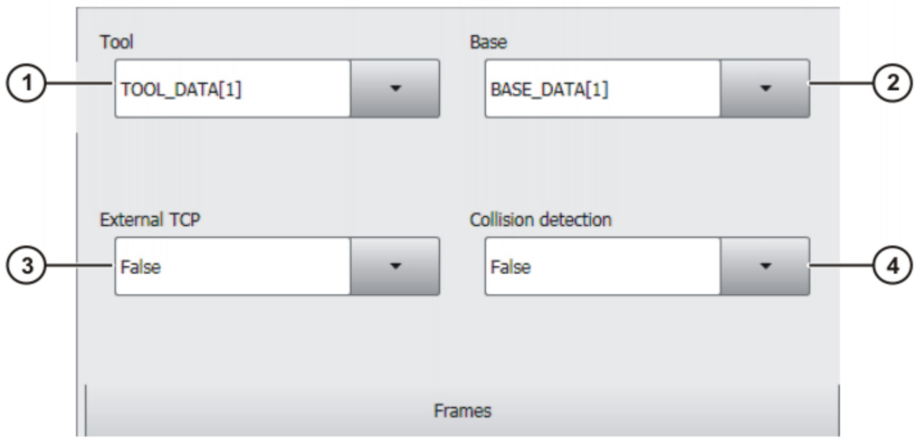

Tool: TOOL_DATA[1]
Base: BASE_DATA[1]
External TCP: False
Collision detection: False

Fig. 6-20: Option window “Frames”

Item	Description
1	Tool selection. If **True** in the box **External TCP**: workpiece selection. Range of values: [1] … [16]
2	Base selection. If **True** in the box **External TCP**: fixed tool selection. Range of values: [1] … [32]
3	Interpolation mode ■ **False**: The tool is mounted on the mounting flange. ■ **True**: the tool is a fixed tool.
4	■ **True**: For this motion, the robot controller calculates the axis torques. These are required for collision detection. ■ **False**: For this motion, the robot controller does not calculate the axis torques. Collision detection is thus not possible for this motion.

6. The acceleration can be reduced from the maximum value in the option window “Motion parameters”. If approximate positioning has been activated, the approximation distance can also be modified. Furthermore, the orientation control can also be modified.

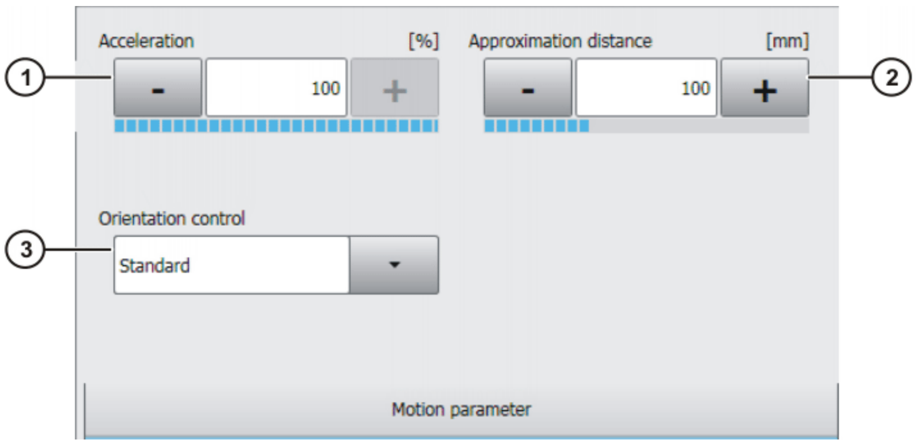


Fig. 6-21: **Option window “Motion parameter” (LIN, CIRC)**

Item	Description
1	Acceleration Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode.
2	Furthest distance before the end point at which approximate positioning can begin The maximum permissible value is half the distance between the start point and the end point. If a higher value is entered, this is ignored and the maximum value is used. This box is only displayed if **CONT** was selected in the inline form.
3	Orientation control selection. ■ **Standard** ■ **Wrist PTP** ■ **Constant** (>>> "Orientation control with CP motions" Page 111)

7. Save instruction with **Cmd Ok**. The current position of the TCP is taught as the end point.

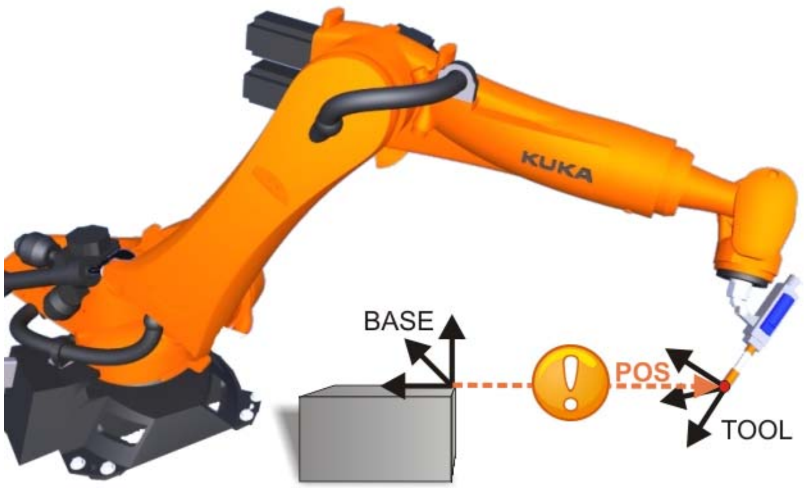


Fig. 6-22: Saving the point coordinates with “Cmd OK” and “Touchup”

6.5 Modifying motion commands

Modifying motion commands

There are different reasons for modifying existing motion commands:

Examples of reasons	Modification to be made
Position of the part to be gripped changes. The position of one of five boreholes changes during processing. A weld seam needs to be shortened.	Modification of the position data
Position of the pallet changes.	Modification of the frame data: base and/or tool
A position has been inadvertently taught with the wrong base.	Modification of the frame data: base and/or tool with update of position
Processing too slow: the cycle time must be improved.	Modification of the motion data: velocity, acceleration Modification of the motion type

Effects of modifying motion commands

Modifying position data

- Only the data set of the point is modified: the point receives new coordinates, as the values are updated with “Touchup”.

  The old point coordinates are overwritten and are subsequently no longer available!

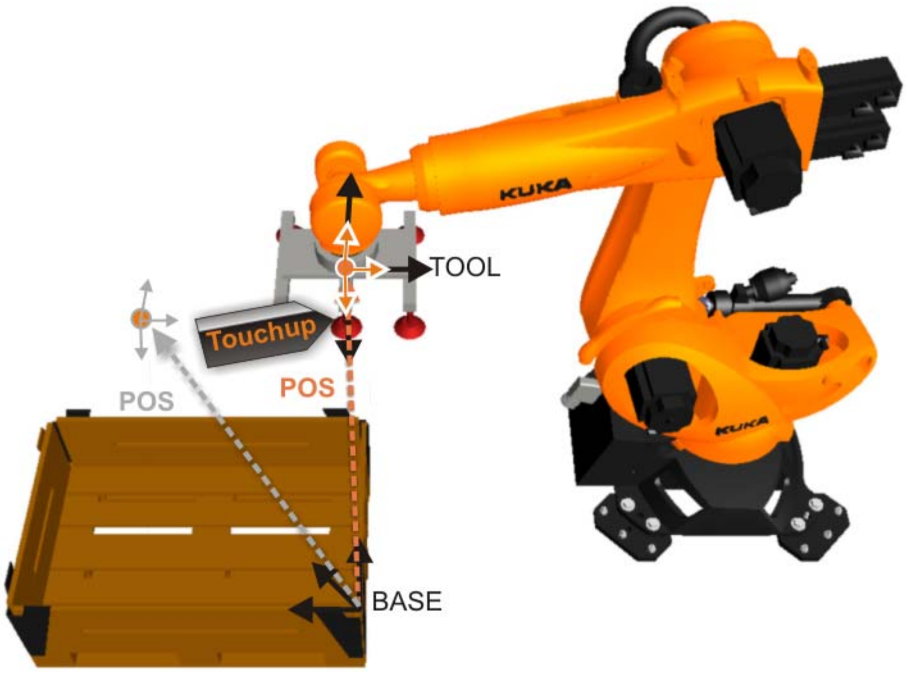


Fig. 6-23: Modification of the robot position with “Touchup”

Modifying frame data

- When modifying frame data (e.g. tool, base), the position is offset (cf. “vector translation”).
- The robot position changes!

  The old coordinates of the point remain saved and valid. Only the reference is changed (e.g. the base).
- The workspace may be exceeded! Certain robot positions are thus not accessible.
- If the robot position is to remain the same despite modification of the frame parameters, the coordinates must be updated by means of “Touchup” in the desired position following modification of the parameters (e.g. Base)!

⚠ WARNING	A user dialog also warns: “Caution: risk of collision when changing point-related frame parameters!”

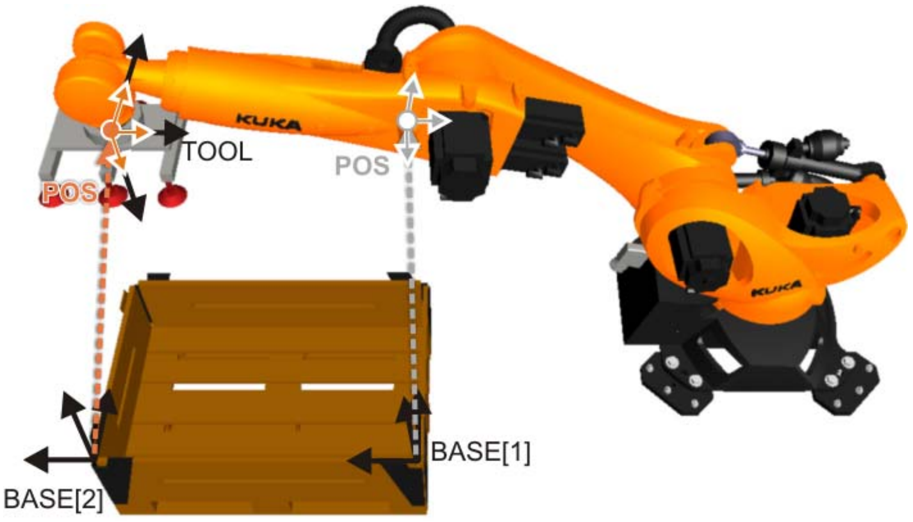


Fig. 6-24: Modifying frame data (example: base)

Modifying motion data

- Modifying the velocity or acceleration changes the motion profile. This can have an effect on the process, particularly in the case of CP applications:
    - Thickness of an adhesive bead.
    - Quality of a weld seam.

Modifying the motion type

- Modification of the motion type always results in a modification of the path planning! In unfavorable circumstances, this can result in collisions, as the path may change unpredictably.

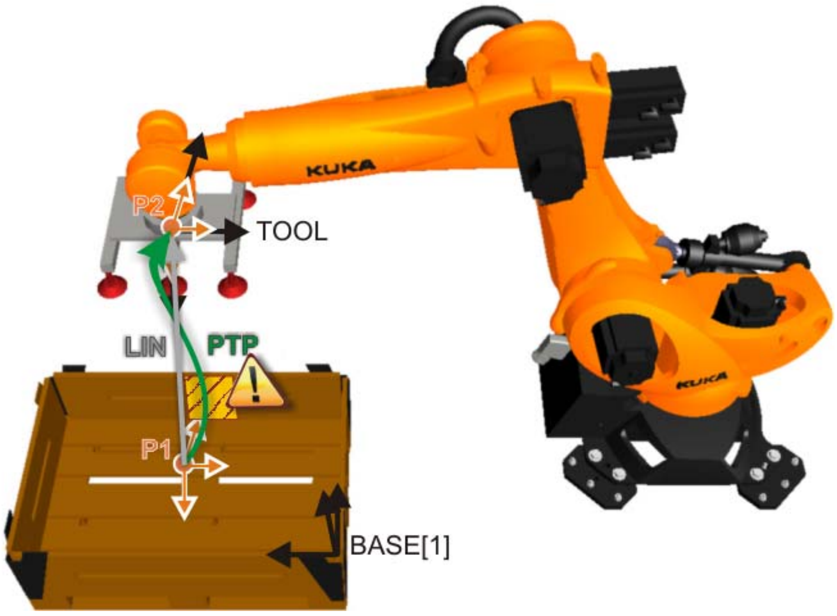


Fig. 6-25: Modifying the motion type

Safety instructions relating to the modification of motion commands

⚠ WARNING Every time motion commands are modified, the robot program must be tested at reduced velocity (T1 mode). Starting the robot program immediately at high velocity can result in damage to the robot system and the overall system, as unforeseeable motions may occur. There is a danger of life-threatening injuries to any person in the danger zone.

Modifying motion parameters – Frames

1. Position the cursor in the line containing the instruction that is to be changed.
2. Press **Change**. The inline form for this instruction is opened.
3. Open the option window “Frames”.
4. Set new “Tool” or “Base” or “External TCP”.
5. Confirm the user dialog “Caution: risk of collision when changing point-related frame parameters!” with **OK**.
6. If you wish to **retain** the **current robot position** with modified tool and/or base settings, it is essential to press the **Touch Up** key to recalculate and save the current position.
7. Save changes by pressing **Cmd Ok**.

⚠ WARNING If frame parameters are modified, the programs must be tested again to ensure there is no risk of a collision.

Modifying the position

Procedure for modifying the robot position:

1. Set T1 mode and position the cursor in the line containing the instruction that is to be changed.
2. Move the robot into the desired position.
3. Press **Change**. The inline form for this instruction is opened.
4. For PTP and LIN motions:
   - Press **Touchup** to accept the current position of the TCP as the new end point.

   For CIRC motions:
   - Press **Teach Aux** to accept the current position of the TCP as the new auxiliary point.
   - Or press **Teach End** 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 **Cmd Ok**.

Modifying motion parameters

This procedure can be used for the following modifications:

- Motion type
- Velocity
- Acceleration
- Approximate positioning
- Approximation distance

1. Position the cursor in the line containing the instruction that is to be changed.
2. Press **Change**. The inline form for this instruction is opened.
3. Modify parameters.
4. Save changes by pressing **Cmd Ok**.

⚠ WARNING If motion parameters are modified, the programs must be checked again for process reliability and to ensure there is no risk of a collision.

6.6 Exercise: CP motion and approximate positioning

Aim of the exercise

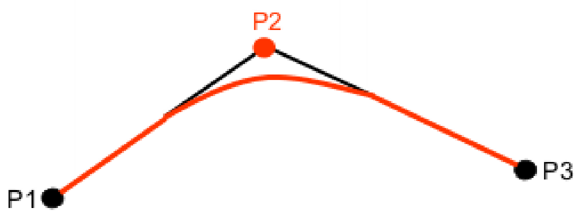
On successful completion of this exercise, you will be able to carry out the following activities:

- Create simple motion programs with the motion types PTP, LIN and CIRC
- Create motion programs with exact positioning points and approximate positioning
- Handle programs in the Navigator (copy, duplicate, rename, delete)

Preconditions

The following are preconditions for successful completion of this exercise:

- Basic principles of motion programming with the motion types PTP, LIN and CIRC
- Theoretical knowledge of approximation of motions



- Theoretical knowledge of the HOME position

Task, part A

Carry out the following tasks: program creation, component contour 1

1. Create a new program with the name **Component_contour1**
2. Teach the component contour 1 marked on the worktable, using the blue base and pen1 as the tool
   - The jog velocity on the worktable is 0.3 m/s
   - Make sure that the longitudinal axis of the tool is always perpendicular to the path contour (orientation control)
3. Test your program in the modes T1, T2 and Automatic. Observe the relevant safety instructions.

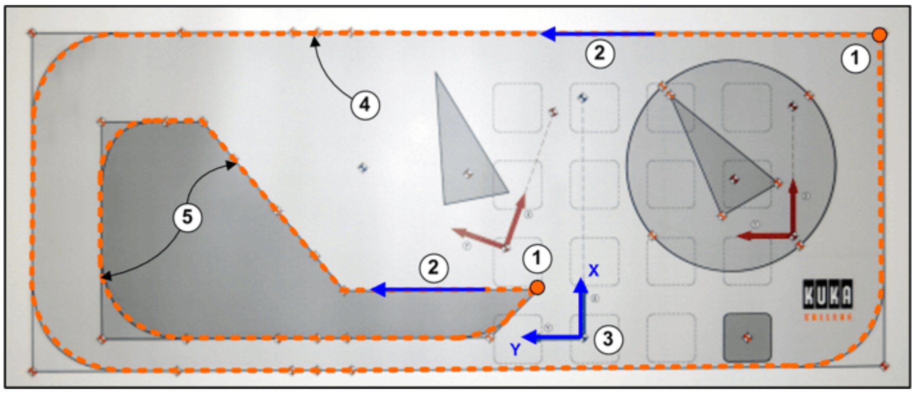


Fig. 6-26: CP motion and approximate positioning: component contour 1 and 2

1	Start points	2	Direction of motion
3	Reference base	4	Component contour 1
5	Component contour 2		

Task, part B

Carry out the following tasks: copying the program and approximate positioning

1. Create a duplicate of the program Component_contour1 with the name **Component1_CONT**.
2. Insert the approximate positioning instruction into the motion commands of the new program so that the robot follows the contour continuously
3. The corners of the contour are to be approximated using different approximation parameters.
4. Test your program in the modes T1, T2 and Automatic. Observe the relevant safety instructions.

Supplementary task

Carry out the following tasks: program creation, component contour 2

1. Create a second program with the name **Component_contour2** Use the same base and the same tool.
   - The jog velocity on the worktable is 0.3 m/s
   - Make sure that the longitudinal axis of the tool is always perpendicular to the path contour (orientation control)
2. Test your program in the modes T1, T2 and Automatic. Observe the relevant safety instructions.
3. Create a duplicate of the program Component_contour2 with the name **Component2_CONT**.
4. Insert the approximate positioning instruction into the motion commands of the new program so that the robot follows the contour continuously

5. Test your program in the modes T1, T2 and Automatic. Observe the relevant safety instructions.

Questions on the exercise

1. What are the characteristics of LIN and CIRC motions?

. . . .

. . . .

. . . .

2. How is the velocity specified for PTP, LIN and CIRC motions and what does this velocity refer to?

. . . .

. . . .

. . . .

3. How is the approximation distance specified for PTP, LIN and CIRC motions?

. . . .

. . . .

4. What must be taken into consideration when programming new `CONT` statements?

. . . .

. . . .

5. What must be taken into consideration when changing the HOME position?

. . . .

. . . .

6. What must be taken into consideration when correcting or modifying programmed points?

. . . .

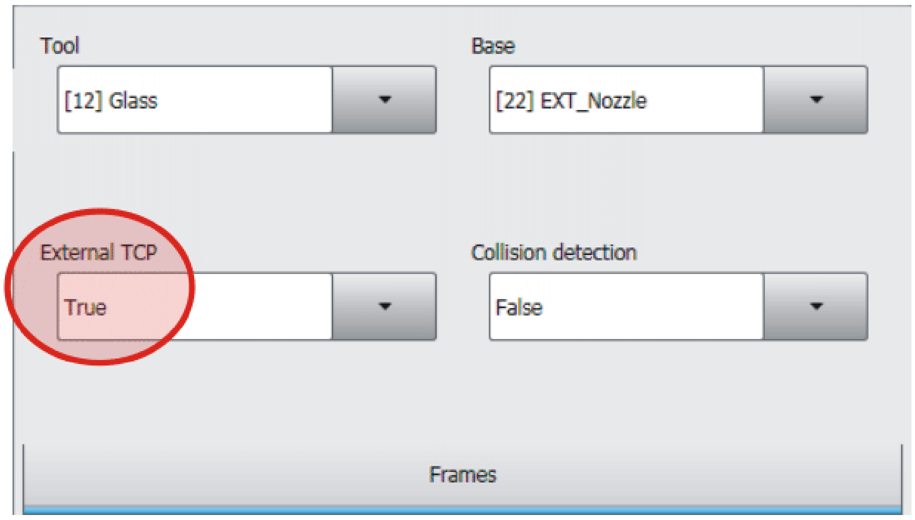
. . . .

6.7 Motion programming with external TCP

Motion programming with external TCP

In the case of motion programming with a fixed tool, the motion sequence differs from that of a standard motion in the following ways:

- Labeling in inline form: the entry **External TCP** in the option window **Frames** must be set to TRUE.



Tool: [12] Glass
Base: [22] EXT_Nozzle
External TCP: True
Collision detection: False

Fig. 6-27: Option window “Frames”: External TCP

- The **motion velocity** then refers to the external TCP.
- The **orientation** along the path then also refers to the external TCP.
- Both the correct base coordinate system (fixed tool/external TCP) and the correct tool coordinate system (moving workpiece) must be specified.

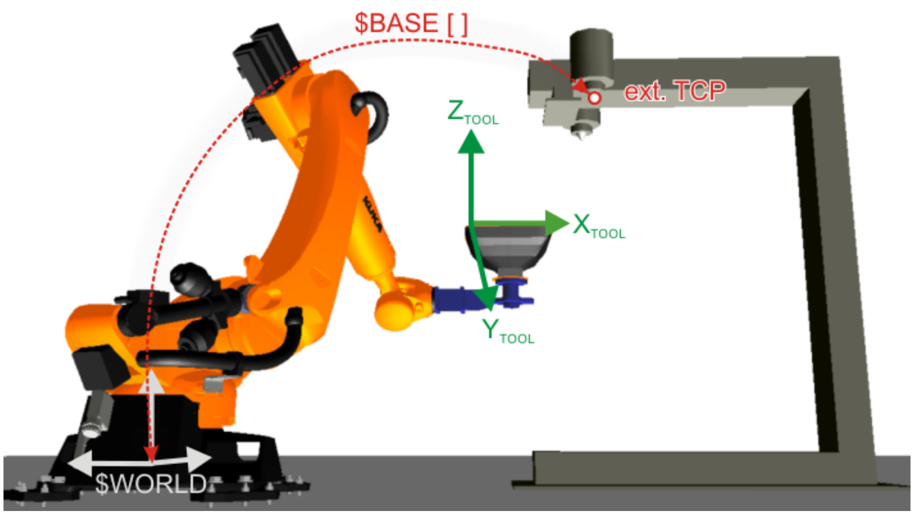


Fig. 6-28: Coordinate systems for fixed tool

6.8 Exercise: Motion programming with external TCP

Aim of the exercise

On successful completion of this exercise, you will be able to carry out the following activities:

- Program motions with a robot guiding a workpiece relative to a fixed tool

Preconditions

The following are preconditions for successful completion of this exercise:

- Knowledge of how to activate an external tool when programming motions

Task description

Carry out the following tasks: Program the contour for adhesive application

1. Manually clamp the panel in the gripper
2. Teach the contour on the plastic panel using the program name **Glue_panel**.
   - Do this using your calibrated external tool **Nozzle** and workpiece **Panel**.
   - Make sure that the longitudinal axis of the fixed tool is always perpendicular to the adhesive application contour
   - The jog velocity on the plastic panel is 0.2 m/s.
3. Test your program in accordance with the instructions.
4. Archive your program.

Questions on the exercise

1. What does the adhesive velocity you have programmed refer to?

. . . .

. . . .

2. How do you activate the external tool in your program?

. . . .

. . . .

7 Using logic functions in the robot program

7.1 Introduction to logic programming

Use of inputs and outputs in logic programming

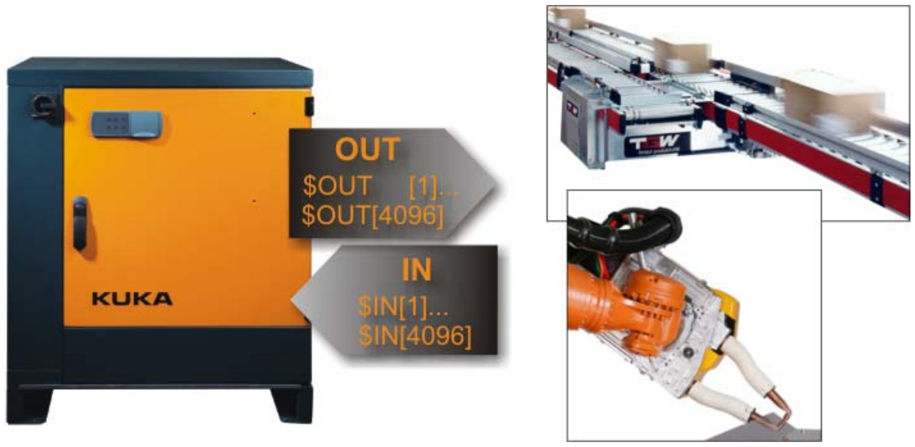


Fig. 7-1: Digital inputs and outputs

In order to implement **communication** *with the* **periphery** *of the robot controller,* **digital** *and* **analog inputs/outputs** *can be used.*

Explanation of terms

Term	Explanation	Example
Communication	Signal exchange via a serial interface	Polling a state (gripper open/closed)
Periphery	"Surroundings"	Tool (e.g. gripper, weld gun, etc.), sensors, material conveyor systems, etc.
Digital	Digital technology: value- and time-discrete signals	Sensor signal: part present: value 1 (TRUE), part not present: value 0 (FALSE)
Analog	Mapping of a physical variable	Temperature measurement
Inputs	The signals *arriving* in the controller via the field bus interface	Sensor signal: gripper is open / gripper is closed
Outputs	The signals *sent* by the controller to the periphery via the field bus interface	Command for switching a valve to close a finger gripper.

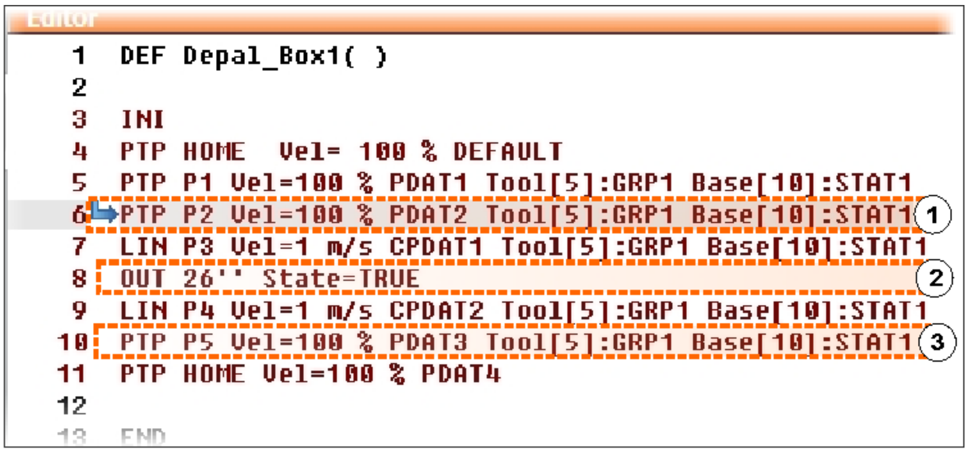
Input and output signals are used for logic statements in the programming of KUKA robots:

- **OUT** | Switches an output at a specific point in the program
- **WAIT FOR** | Signal-dependent wait function: the controller waits for a signal here:
  - Input **IN**
  - Output **OUT**
  - Time signal **TIMER**
  - Internal memory address in the controller (cyclical flag/1-bit memory) **FLAG** or **CYCFLAG** (if continuously and cyclically evaluated)
- **WAIT** | Time-dependent wait function: the controller waits a specified time at this point in the program.

7.2 Programming wait functions

Computer advance run

The computer advance run loads the motion blocks in the advance run (not visible for the operator) to allow the controller to carry out path planning in the case of approximate positioning commands. It is not only motion data that are processed in the advance run, however, but also arithmetical data and commands for controlling the periphery.



```
DEF Depal_Box1( )

INI
PTP HOME  Vel= 100 % DEFAULT
PTP P1 Vel=100 % PDAT1 Tool[5]:GRP1 Base[10]:STAT1
PTP P2 Vel=100 % PDAT2 Tool[5]:GRP1 Base[10]:STAT1
LIN P3 Vel=1 m/s CPDAT1 Tool[5]:GRP1 Base[10]:STAT1
OUT 26'' State=TRUE
LIN P4 Vel=1 m/s CPDAT2 Tool[5]:GRP1 Base[10]:STAT1
PTP P5 Vel=100 % PDAT3 Tool[5]:GRP1 Base[10]:STAT1
PTP HOME Vel=100 % PDAT4

END
```

Fig. 7-2: Computer advance run

1 Position of the main run pointer (gray bar)
2 Command set that triggers an advance run stop
3 Possible position of the advance run pointer (not visible)

Certain statements trigger an advance run stop. These include statements that influence the periphery, e.g. OUT statements (close gripper, open weld gun). If the advance run pointer is stopped, approximate positioning cannot be carried out.

Wait functions

Wait functions in a motion program can be programmed very simply using inline forms. A distinction is made between time-dependent wait functions and signal-dependent wait functions.

With **WAIT**, the robot motion is stopped for a programmed time. WAIT always triggers an advance run stop.

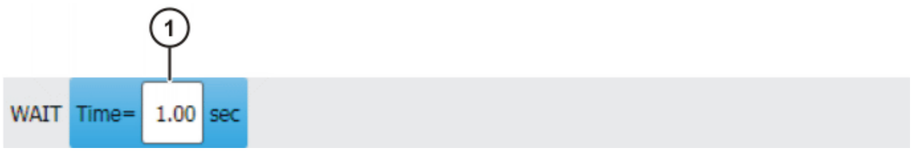


Fig. 7-3: Inline form “WAIT”

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

Example program:

```
PTP P1 Vel=100% PDAT1
PTP P2 Vel=100% PDAT2
WAIT Time=2 sec
PTP P3 Vel=100% PDAT3
```

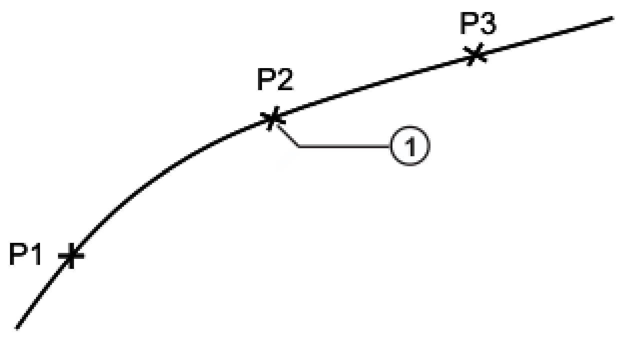


Fig. 7-4: Example motion for logic

Item	Comments
1	Motion is interrupted for 2 seconds at point P2

WAIT FOR 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.

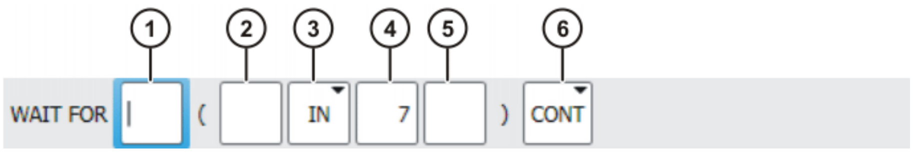


Fig. 7-5: Inline form “WAITFOR”

Item	Description
1	Add external logic operation. The operator is situated between the bracketed expressions. ■ **AND** ■ **OR** ■ **EXOR** Add NOT. ■ **NOT** ■ **[Empty box]** Enter the desired operator by means of the corresponding button.
2	Add internal logic operation. The operator is situated inside a bracketed expression. ■ **AND** ■ **OR** ■ **EXOR** Add NOT. ■ **NOT** ■ **[Empty box]** Enter the desired operator by means of the corresponding button.

Item	Description
3	Signal for which the system is waiting ▪ **IN** ▪ **OUT** ▪ **CYCFLAG** ▪ **TIMER** ▪ **FLAG**
4	Number of the signal ▪ **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 **Long text**. The name is freely selectable.
6	▪ **CONT**: Execution in the advance run ▪ **[Empty box]**: Execution with advance run stop

⚠ CAUTION If the entry CONT is used, it must be taken into consideration that the signal is polled in the advance run. A signal change after the advance run will not be detected.

Logic operations

If signal-dependent wait functions are used, so too are logic operations. Logic operations can be used for combined polling of different signals or states: dependencies can be created, for example, or specific states can be excluded.

The result of a function with a logic operator always provides a truth value, i.e. “True” (value 1) or “False” (value 0).

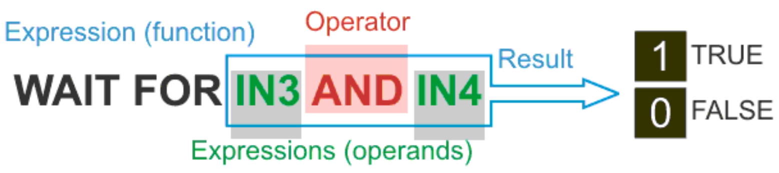


Fig. 7-6: Example and principle of a logic operation

Operators for logic operations are:

- **NOT** | This operand is used for negation, i.e. the value is inverted (“True” becomes “False”).
- **AND** | The result of the expression is true if both linked expressions are true.
- **OR** | The result of the expression is true if at least one of the linked expressions is true.
- **EXOR** | The result of the expression is true if both of the expressions linked by the operator have different truth values.

Processing with and without advance run (CONT)

Signal-dependent wait functions can be programmed with or without processing in the advance run. **Without advance run** means that the motion will always stop at the point and that the signal will be checked there: (1) (>>> Fig. 7-7). In other words, the point cannot be approximated.

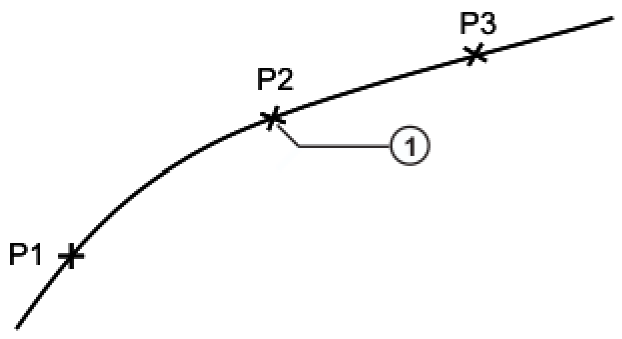


Fig. 7-7: Example motion for logic

```
PTP P1 Vel=100% PDAT1
PTP P2 CONT Vel=100% PDAT2
WAIT FOR IN 10 'door_signal'
PTP P3 Vel=100% PDAT3
```

Signal-dependent wait functions programmed **with advance run** allow approximate positioning to be carried out for the point before the command line. However, the current position of the advance run pointer is not unambiguous (default value: three motion blocks), so the precise moment at which the signal will be checked is not specified (1) (>>> Fig. 7-8). Furthermore, signal changes after the signal has been checked are not detected!

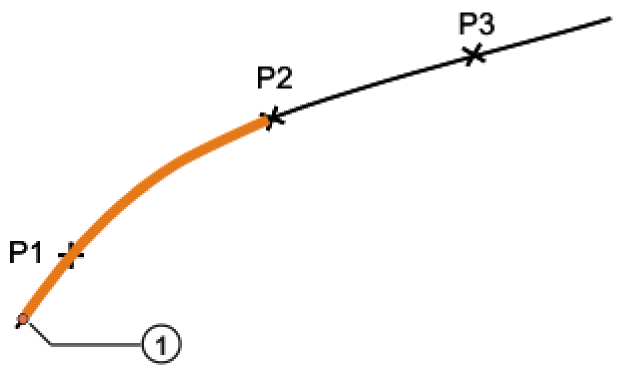


Fig. 7-8: Example motion for logic with advance run

```
PTP P1 Vel=100% PDAT1
PTP P2 CONT Vel=100% PDAT2
WAIT FOR IN 10 'door_signal' CONT
PTP P3 Vel=100% PDAT3
```

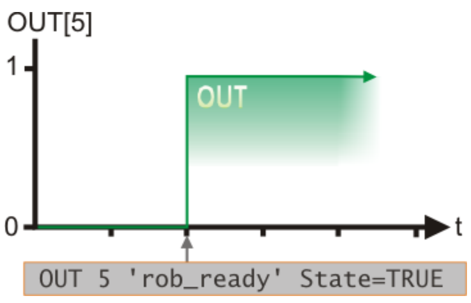
Procedure

1. Position the cursor in the line after which the logic instruction is to be inserted.
2. Select the menu sequence **Commands** > **Logic** > **WAIT FOR** or **WAIT**.
3. Set the parameters in the inline form.
4. Save instruction with **Cmd Ok**.

7.3 Programming simple switching functions

Simple switching function

A switching function can be used to send a digital signal to the periphery. For this, an output number defined beforehand and corresponding to the interface is used.



OUT 5 'rob_ready' State=TRUE

Fig. 7-9: Static switching function

The signal is set statically, i.e. it remains active until a different value is assigned to the output. The switching function is implemented in the program by means of an inline form:

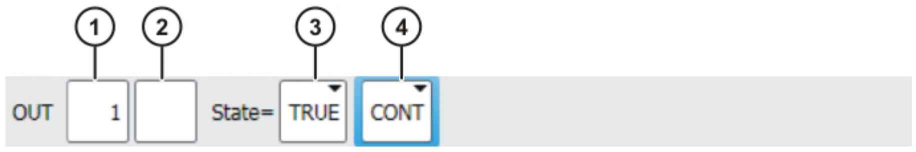


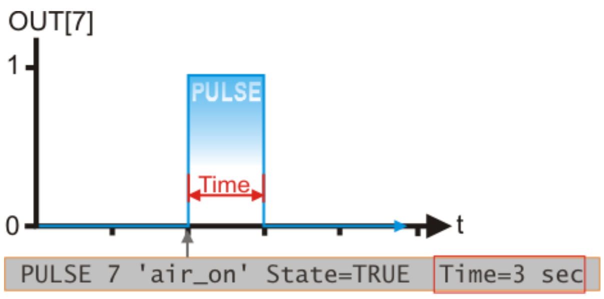
Fig. 7-10: Inline form “OUT”

Item	Description
1	Output number ■ **1 … 4096**
2	If a name exists for the output, this name is displayed. Only for the user group “Expert”: A name can be entered by pressing **Long text**. The name is freely selectable.
3	State to which the output is switched ■ **TRUE** ■ **FALSE**
4	■ **CONT**: Execution in the advance run ■ **[Empty box]**: Execution with advance run stop

⚠ CAUTION If the entry CONT is used, it must be taken into consideration that the signal is set in the advance run.

Pulsed switching functions

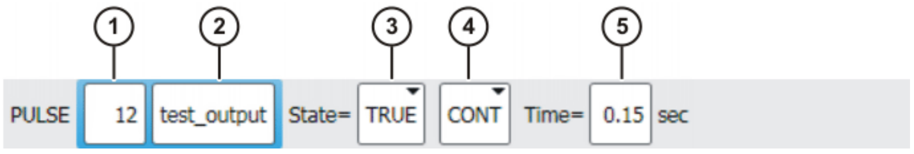
As in the case of simple switching functions, the value of an output is changed here, as well. With pulsing, however, the signal is withdrawn again after a defined time.



PULSE 7 'air_on' State=TRUE Time=3 sec

Fig. 7-11: Pulsed signal level

Once again, programming is carried out by means of an inline form in which a pulse of a defined length is set.



PULSE 12 test_output State= TRUE CONT Time= 0.15 sec

Fig. 7-12: Inline form “PULSE”

Item	Description
1	Output number ■ **1 … 4096**
2	If a name exists for the output, this name is displayed. Only for the user group “Expert”: A name can be entered by pressing **Long text**. The name is freely selectable.
3	State to which the output is switched ■ **TRUE**: “High” level ■ **FALSE**: “Low” level
4	■ **CONT**: Execution in the advance run ■ **[Empty box]**: Execution with advance run stop
5	Length of the pulse ■ **0.10 … 3.00 s**

Effects of CONT on switching functions

If the entry CONT is left out in the inline form OUT, an **advance run stop** is forced during the switching operation and exact positioning is carried out at the point before the switching command. Once the output has been set, the motion is resumed.

```
LIN P1 Vel=0.2 m/s CPDAT1
LIN P2 CONT Vel=0.2 m/s CPDAT2
LIN P3 CONT Vel=0.2 m/s CPDAT3
OUT 5 'rob_ready' State=TRUE
LIN P4 Vel=0.2 m/s CPDAT4
```

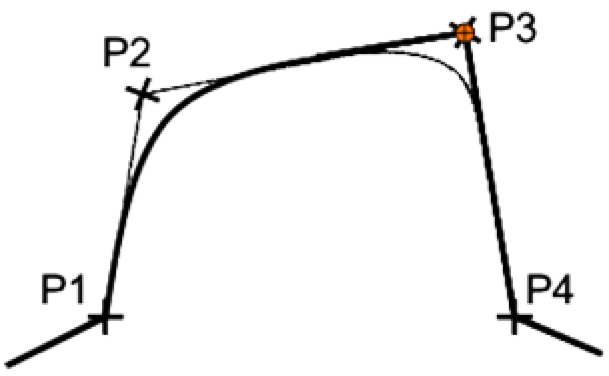


Fig. 7-13: Example motion with switching with advance run stop

Setting the entry CONT has the effect that the advance run pointer is not stopped (no advance run stop is triggered). This means that a motion before the switching command can be approximated. The signal is set in the **advance run**.

```
LIN P1 Vel=0.2 m/s CPDAT1
LIN P2 CONT Vel=0.2 m/s CPDAT2
LIN P3 CONT Vel=0.2 m/s CPDAT3
OUT 5 'rob_ready' State=TRUE CONT
LIN P4 Vel=0.2 m/s CPDAT4
```

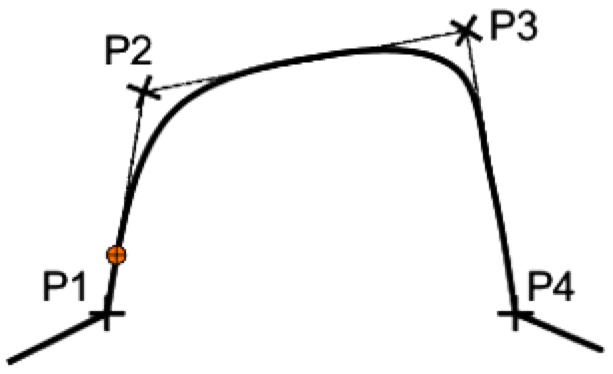


Fig. 7-14: Example motion with switching in the advance run

> ⚠ **CAUTION** The default value for the advance run pointer is three lines. The advance run may vary, however; i.e. it has to be taken into consideration that the switching point may not always occur at the same time!

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** or **PULSE**.
3. Set the parameters in the inline form.
4. Save instruction with **Cmd Ok**.

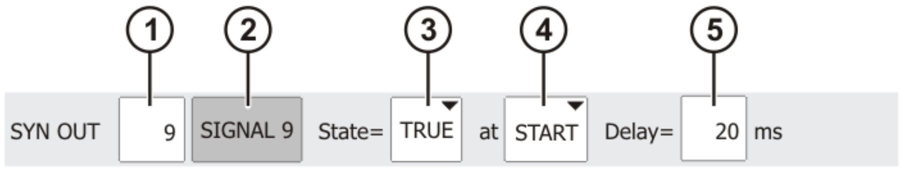
7.4 Programming time-distance functions

General

A time-distance function can be used to set an output at a specific point on the path without interrupting the robot motion. A distinction is made between “static” (SYN OUT) and “dynamic” (SYN Pulse) switching. In the case of SYN OUT 5 switching, the same signal is switched as with SYN PULSE 5. It is only the switching method that differs.

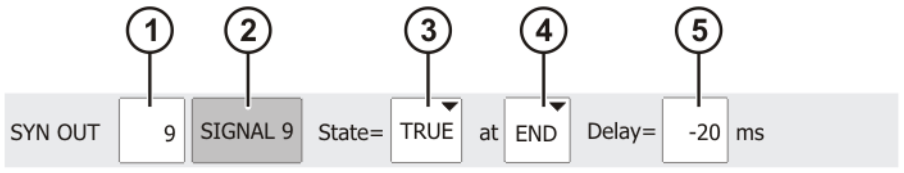
Option Start/End

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 **in time**. The reference motion block can be a LIN, CIRC or PTP motion.



SYN OUT 9 SIGNAL 9 State= TRUE at START Delay= 20 ms

Fig. 7-15: Inline form “SYN OUT”, option “START”



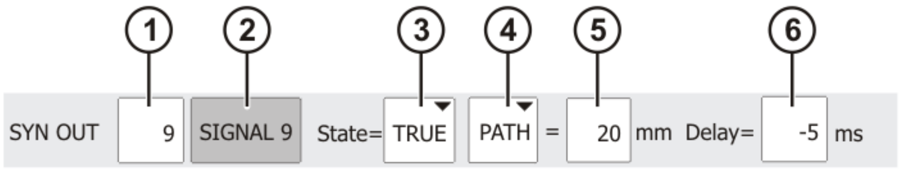
SYN OUT 9 SIGNAL 9 State= TRUE at END Delay= -20 ms

Fig. 7-16: Inline form “SYN OUT”, option “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 relative to the start point of the motion block. ■ **END**: Switching is carried out relative to the end point of the motion block.	START, END Option PATH:
5	Switching action delay **Note:** The time specification is absolute. The position of the switching point thus varies according to the velocity of the robot.	-1000 … +1000 ms

Option PATH

With the 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/or delayed or brought forward. The reference motion block can be a LIN or CIRC motion. It must **not** be a PTP motion.



SYN OUT 9 SIGNAL 9 State= TRUE PATH = 20 mm Delay= -5 ms

Fig. 7-17: 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	Point at which switching is carried out ■ **PATH**: Switching is carried out relative to the end point of the motion block.	START, END Option PATH:
5	Switching action offset **Note:** The specification of the location is relative to the end point of the motion block. The position of the switching point thus does not vary if the velocity of the robot changes.	-1000 … +1000 ms
6	Switching action delay **Note:** The delay is relative to the offset.	

Effect of the switching options Start/End

Example program 1: Option Start

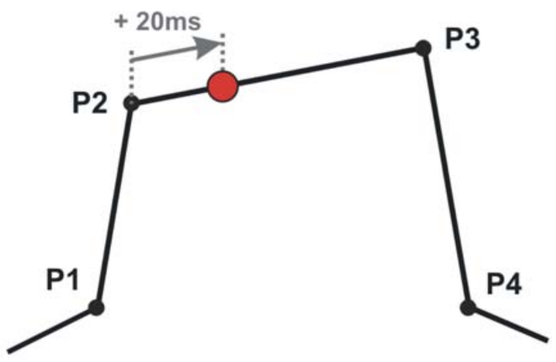


Fig. 7-18: SYN OUT Start with positive delay

```
LIN P1 VEL=0.3m/s CPDAT1
LIN P2 VEL=0.3m/s CPDAT2
;Schaltfunktion bezogen auf P2
SYN OUT 8 'SIGNAL 8' State= TRUE at Start Delay=20ms
LIN P3 VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

Example program 2: Option Start with CONT and positive delay

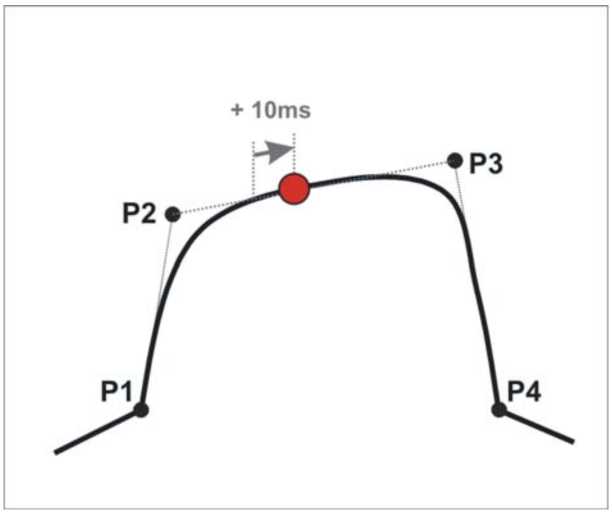


Fig. 7-19: SYN OUT Start with CONT and positive delay

```
LIN P1 VEL=0.3m/s CPDAT1
LIN P2 CONT VEL=0.3m/s CPDAT2
;Schaltfunktion bezogen auf P2
SYN OUT 8 'SIGNAL 8' State= TRUE at Start Delay=10ms
LIN P3 CONT VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

Example program 3: Option End with negative delay

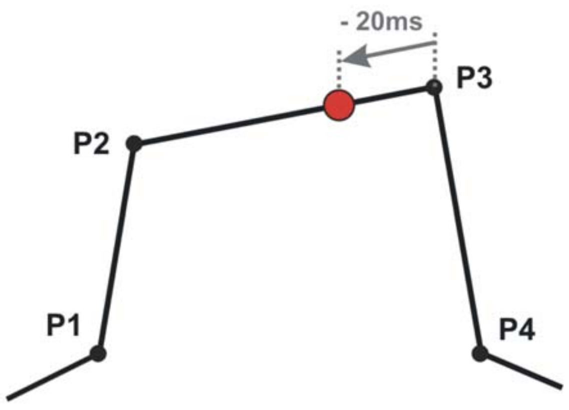


Fig. 7-20: SYN OUT END with negative delay

```
LIN P1 VEL=0.3m/s CPDAT1
LIN P2 VEL=0.3m/s CPDAT2
;Schaltfunktion bezogen auf P3
SYN OUT 9 'SIGNAL 9' Status= TRUE at End Delay=-20ms
LIN P3 VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

Example program 4: Option End with CONT and negative delay

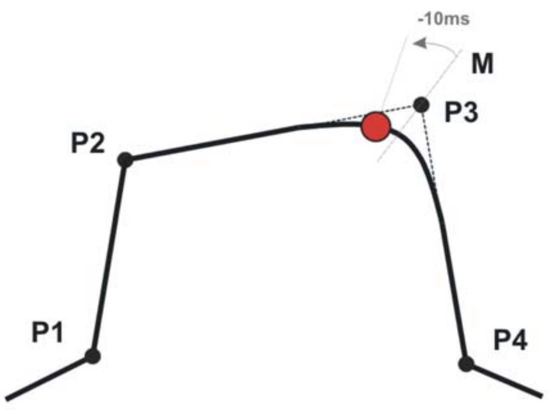


Fig. 7-21: SYN OUT with option END with negative delay

```
LIN P1 VEL=0.3m/s CPDAT1
LIN P2 VEL=0.3m/s CPDAT2
;Schaltfunktion bezogen auf P3
SYN OUT 9 'SIGNAL 9' Status= TRUE at End Delay=-10ms
LIN P3 VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

Example program 5: Option End with CONT and positive delay

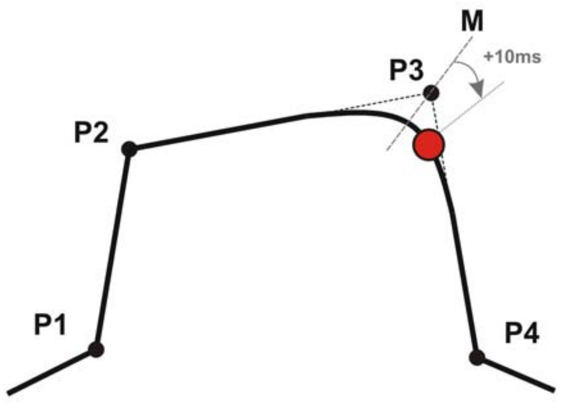


Fig. 7-22: SYN OUT with option END and positive delay

```
LIN P1 VEL=0.3m/s CPDAT1
LIN P2 VEL=0.3m/s CPDAT2
;Schaltfunktion bezogen auf P3
SYN OUT 9 'SIGNAL 9' Status= TRUE at End Delay=10ms
LIN P3 VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

Switching limits without CONT

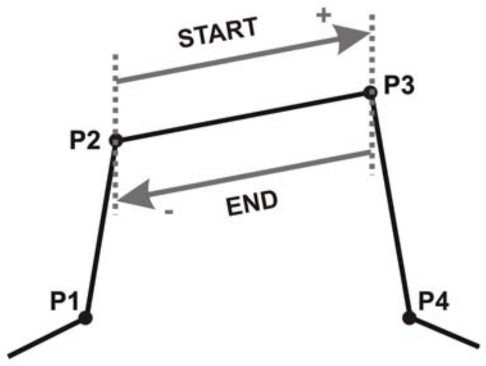


Fig. 7-23: Switching limits, option Start/End without CONT

Switching limits with CONT:

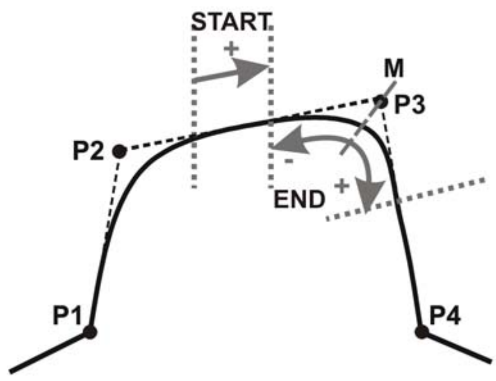


Fig. 7-24: Switching limits, option Start/End with CONT

Effect of the switching option PATH

Example program:

A milling tool has to be switched on the path. Machining of the workpiece is to begin 20 mm after P3. For the milling tool to have reached its full speed 20 mm (Path=20) after P3, it must already be switched on 5 ms beforehand (Delay=-5ms).

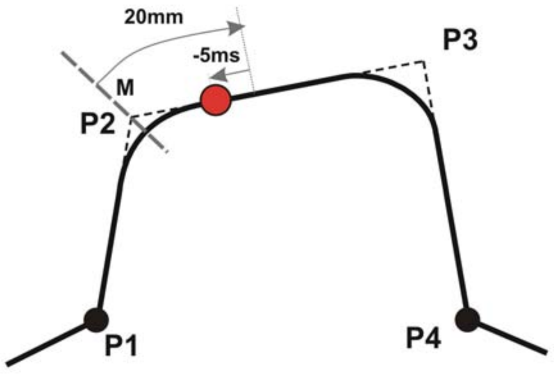


Fig. 7-25

```
LIN P1 VEL=0.3m/s CPDAT1
;Schaltfunktion bezogen auf P2
SYN OUT 9 'SIGNAL 9' Status= True Path=20 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
```

Switching limits

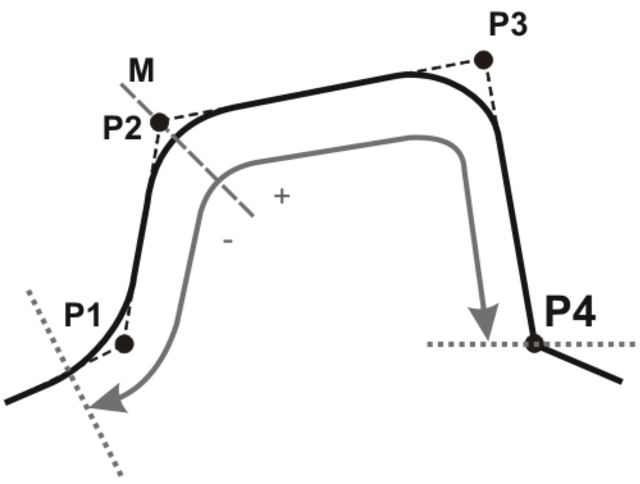


Fig. 7-26

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** or **SYN PULSE**.
3. Set the parameters in the inline form.
4. Save instruction with **Cmd Ok**.

7.5 Exercise: Logic statements and switching functions

Aim of the exercise

On successful completion of this exercise, you will be able to carry out the following activities:

- Program simple logic statements
- Execute simple switching functions
- Execute path-related switching functions
- Program signal-dependent wait functions

Preconditions

The following are preconditions for successful completion of this exercise:

- Theoretical knowledge of programming of simple logic statements
  - Knowledge of simple switching functions
  - Knowledge of simple pulse functions
  - Knowledge of path-related switching functions
  - Knowledge of path-related pulse functions
  - Knowledge of wait functions

Task description

Carry out the following tasks: logic programming component contour 1 with adhesive application

1. Create a duplicate of the program Component1_CONT with the name **Adhesive_contour**
2. Add the following logic functions to the program:
   - The PLC is to issue an enabling signal (input 11) before the robot leaves the HOME position.
   - The adhesive nozzle must be activated 0.5 seconds before it reaches the component (output 13).
   - A signal lamp is to be activated at the transition from the flat part of the table to the curved part of the component and deactivated again at the transition from the curved part back to the flat part of the table (output 12).
   - The adhesive nozzle must be deactivated again 0.75 seconds before it leaves the component (output 13).
   - A “finished” signal is to be sent to the PLC 50 mm before the end of work on the component. This signal (output 11) for the PLC is to be active for 2 seconds.
3. Test your program in accordance with the instructions.

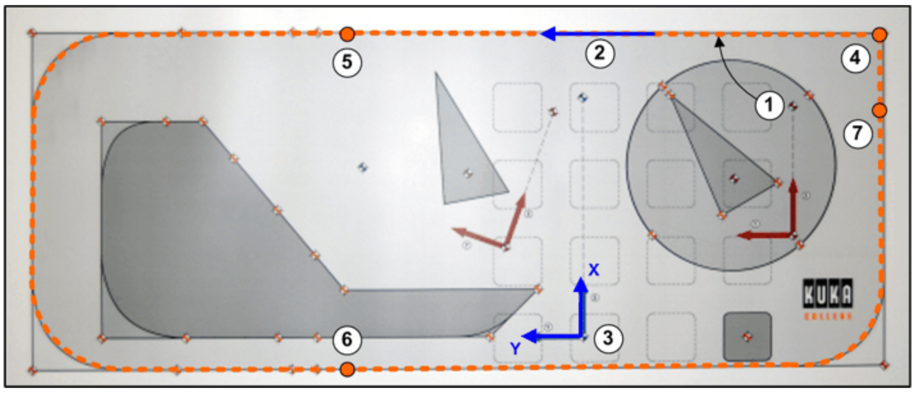


Fig. 7-27: Inputs and outputs: adhesive application

1	Component contour 1	2	Direction of motion
3	Reference base	4	Component start and end point

5	Transition from flat to curve	6	Transition from curve to flat
7	Point before end of component		

Questions on the exercise

1. What is the difference between the `OUT` and `OUT CONT` statements? What must be taken into consideration when using them?

. . . .

. . . .

2. What is the difference between the `PULSE` and `OUT` statements?

. . . .

. . . .

3. When are `SYN OUT` statements used?

. . . .

. . . .

4. What are the restrictions when using `SYN OUT` Path statements in motion programming?

. . . .

. . . .

5. What is the danger of using the `WAIT FOR` statement with a `CONT` statement?

. . . .

. . . .

8 Working with variables

8.1 Displaying and modifying variable values

Overview of variables

Variables are placeholders for values that arise during a computing process. Variables are labeled with their memory location, type, name and content.

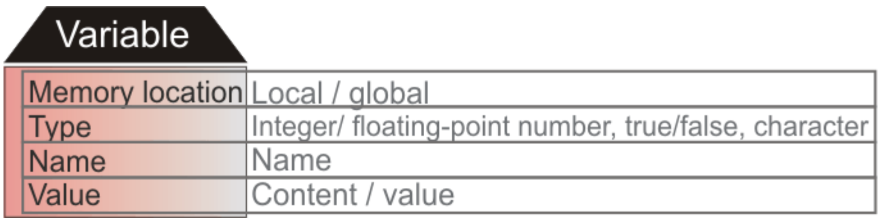


Fig. 8-1: Labeling of variables

The memory location of a variable is very important for its validity. A global variable is created in the system files and is valid in all programs. A local variable is created in the application program and is thus only valid (and readable) in the running program.

Examples of variables used:

Variable	Memory location	Type	Name	Value
Current tool	global | KUKA system variable	Integer	$ACT_TOOL	5
Current base:	global | KUKA system variable	Integer	$ACT_BASE	12
Part counter	local | application program	Integer	zaehler	3
Negative angle value for the software limit switch of axis 2	global | machine.dat	Floating-point number	$SoftN_End[2]	-104.5
Error state	global | e.g. in config.dat	True/False value	stoerung	True

Availability and validity of variables in the display

The memory location of a variable is very important for the display option of a variable:

- **global** | If the variable is global, it can be displayed at any time. In such a case, the variable must be stored as a global variable in a system file (e.g. config.dat, machine.dat) or in a local data list.
- **local** | In the case of local variables, a distinction is made between local in the program file (.src) and local in the local data list (*.dat). If the variable is declared in the .src file, it only exists during the runtime of the program. This is referred to as a "runtime variable". If a variable is declared as local in the .dat file, it is only known in the corresponding program file, but its value remains active after the program has been deselected.

Displaying and modifying the value of a variable

1. In the main menu, select **Display** > **Variable** > **Single**.
   The **Variable display - 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.

4. Press Enter.

   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 Enter.

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

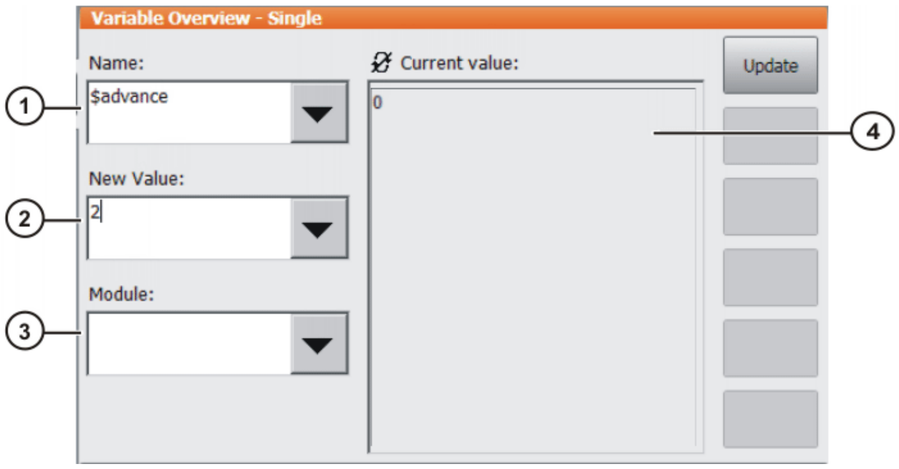


Fig. 8-2: Variable Overview - Single window

Item	Description
1	Name of the variable to be modified.
2	New value to be assigned to the variable.
3	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	This box has two states: ■ : The displayed value is not refreshed automatically. ■ : The displayed value is refreshed automatically. Switching between the states: ■ Press **Refresh**.

8.2 Displaying robot states

Internal system values

Much information about the status of the robot can be obtained by displaying internal system values. These values can be displayed at any time.

Internal system values are referred to as “system variables”.
A *variable* is a reserved memory location. This memory location (or placeholder for values) always has a name and a specific address in the memory of the computer.

System variables available for monitoring robot states include the following:

- Timers
- Flags

- Counter
- Input and output signals (IN/OUT) are also managed as system variables.

System variable	Examples of use of the display function
$TIMER[1..64]	Checking wait times of the robot (cooling of components, wait time due to process duration, etc.).
$FLAG[1..1024] $CYCFLAG[1..256]	Flags that have been used in the program can also be monitored outside the program (global). Cyclical flags are also evaluated continuously.
I[1..20]	Counter that counts the processing steps.
$IN[1..4096]	Checks whether a gripper is open or closed (the sensors of the gripper signal the status via an input signal).
$OUT[1..4096]	Checking a gripper command (an output signal transfers a command to the actuators of the gripper).

Features of system variables

KUKA system variables always begin with the “$” sign. System variables can always be displayed, as they are always valid. The global data lists serve as memory locations.

System information display

Procedure for displaying flags, counters and timers:

- Select **Monitor** > **Variable** in the main menu.

  The various system variables are available for selection:

  - **Cyclical flags**
  - **Flags**
  - **Counters**
  - **Timers**

Procedure for displaying inputs and outputs:

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

8.3 Exercise: Displaying system variables

Aim of the exercise

On successful completion of this exercise, you will be able to carry out the following activities:

- Open the variable display
- Display system variables

Preconditions

The following are preconditions for successful completion of this exercise:

- Theoretical knowledge about displaying system variables
- Theoretical knowledge of system variables

Task description

Carry out the following tasks:

1. Open the variable display.
2. Display the current home position (variable name: XHOME).
3. Display the current robot position (variable name: $pos_act).
4. Determine the position of the software limit switches for axes 1-3 of your robot (variable name: $softn_end[Axis] and $softp_end[Axis]).
5. Determine the value of the advance run pointer (variable name: $advance).

9 Using technology packages

9.1 Gripper operation with KUKA.GripperTech

Technology package KUKA.GripperTech

KUKA.Gripper&SpotTech is an add-on technology package. It simplifies the use of a gripper in terms of:

The following technology keys are required for operating the gripper:

Technology key	Description
1	Select gripper. The number of the gripper is displayed. ■ Pressing the upper key counts upwards. ■ Pressing the lower key counts downwards.
	Toggle between the gripper states (e.g. open or close). The current state is not displayed. The possible states depend on the configured gripper type. In the case of weld guns, the possible states depend on the configuration of the manual gun control.

Procedure for gripper operation

NOTICE Before a gripper can be operated using the technology keys, the technology keys must first be activated!
In the main menu, select **Configure** > **Status keys** > **GripperTech**.

⚠ WARNING **Warning!**
When using the gripper system there is a risk of crushing and cutting. Anyone using the gripper must ensure that no part of the body can be crushed by the gripper.

1. Select the gripper using the technology key.

   1

2. Activate operating mode T1 or T2.
3. Press enabling switch.
4. Control the gripper using the technology key.

9.2 Gripper programming with KUKA.GripperTech

Programming gripper commands

The technology package KUKA.GripperTech allows the programming of gripper commands directly in the selected program using ready-made inline forms. Two commands are available for this:

- **SET Gripper** | Command for opening/closing the gripper in the program
- **CHECK Gripper** | Command for checking the gripper state

Gripper programming functions

Gripper command during motion

- It is generally possible to program the gripper command so that it is executed relative to the start or end point.
- To do so, simply activate the entry **CONT** in the inline form and specify the duration of the delay in ms (**Delay**).

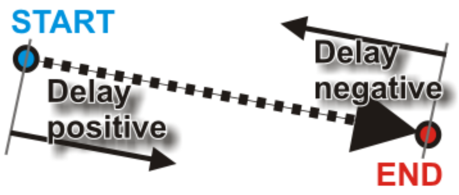
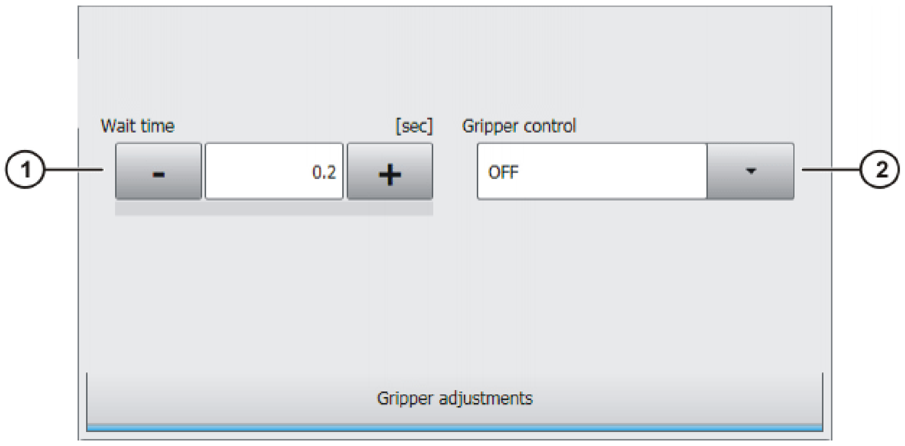


Fig. 9-1: Schematic diagram of delay

⚠ WARNING Gripper commands to be processed during the motion must be selected carefully, as careless use can result in injuries and material damage due to flying parts or collisions!

Gripper settings for exact positioning



Wait time [sec]: 0.2
Gripper control: OFF
Gripper adjustments

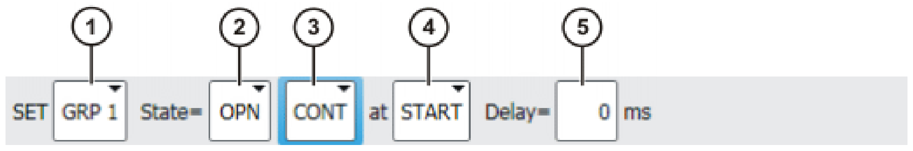
Fig. 9-2: Gripper adjustments

- Use gripper monitoring:
  - If gripper monitoring is activated with **ON**, the parameterized sensor systems are polled.
  - If there is no feedback from the sensors, a timeout error occurs and the sensor can be simulated in test mode.
  - If gripper monitoring is deactivated with OFF, the system waits for the parameterized wait time before the program is resumed.

Procedure for gripper programming

Procedure

1. Select the menu sequence **Commands** > **GripperTech** > **Gripper**.
2. Set the parameters in the inline form.
3. Save with **Cmd Ok**.



SET GRP 1 State= OPN CONT at START Delay= 0 ms

Fig. 9-3: Inline form for gripper with approximate positioning

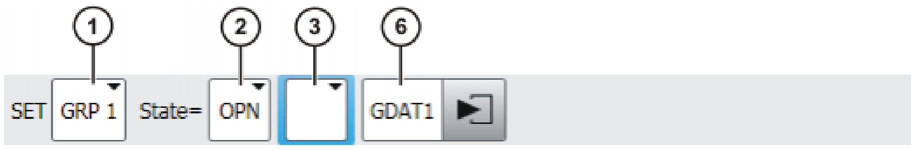


Fig. 9-4: Inline form for gripper without approximate positioning

Item	Description
1	Select gripper. ■ All configured grippers are available for selection.
2	Select the switching state of the gripper. ■ The number depends on the gripper type. ■ The designation depends on the configuration.
3	Execution in the advance run. ■ **CONT**: Execution in the advance run. ■ [*blank*]: Execution with advance run stop.
4	Box only available if **CONT** selected. ■ **START**: The gripper action is executed at the start point of the motion. ■ **END**: The gripper action is executed at the end point of the motion.
5	Box only available if **CONT** selected. Define a wait time, relative to the start or end point of the motion, for execution of the gripper action. ■ -200 ... 200 ms
6	Data set with gripper parameters

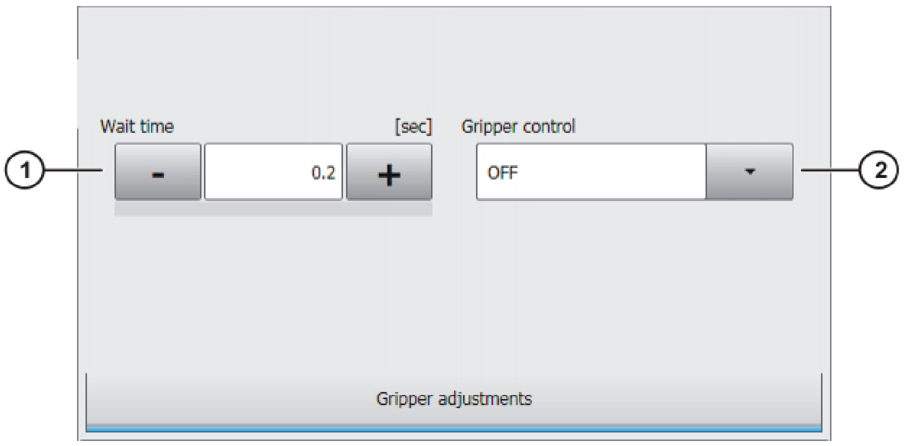


Fig. 9-5: Gripper adjustments

Item	Description
1	Wait time after which the programmed motion is resumed. ■ 0 … 10 s
2	Gripper control ■ **OFF** (default), **ON**

Procedure for programming gripper monitoring

Procedure

1. Select the menu sequence **Commands** > **GripperTech** > **Check Gripper**.
2. Set the parameters in the inline form.
3. Save with **Cmd Ok**.

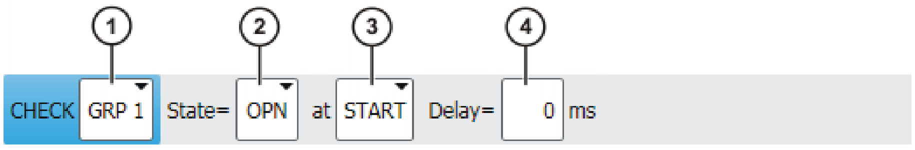


Fig. 9-6: Inline form "Check Gripper"

Item	Description
1	Select gripper. ■ All configured grippers are available for selection.
2	Select the switching state of the gripper. ■ The number depends on the gripper type. ■ The designation depends on the configuration.
3	Select the time at which the sensor interrogation is executed. ■ **START**: The sensor interrogation is executed at the start point of the motion. ■ **END**: The sensor interrogation is executed at the end point of the motion.
4	Define a wait time, relative to the start or end point of the motion, for execution of the sensor interrogation.

9.3 KUKA.GripperTech configuration

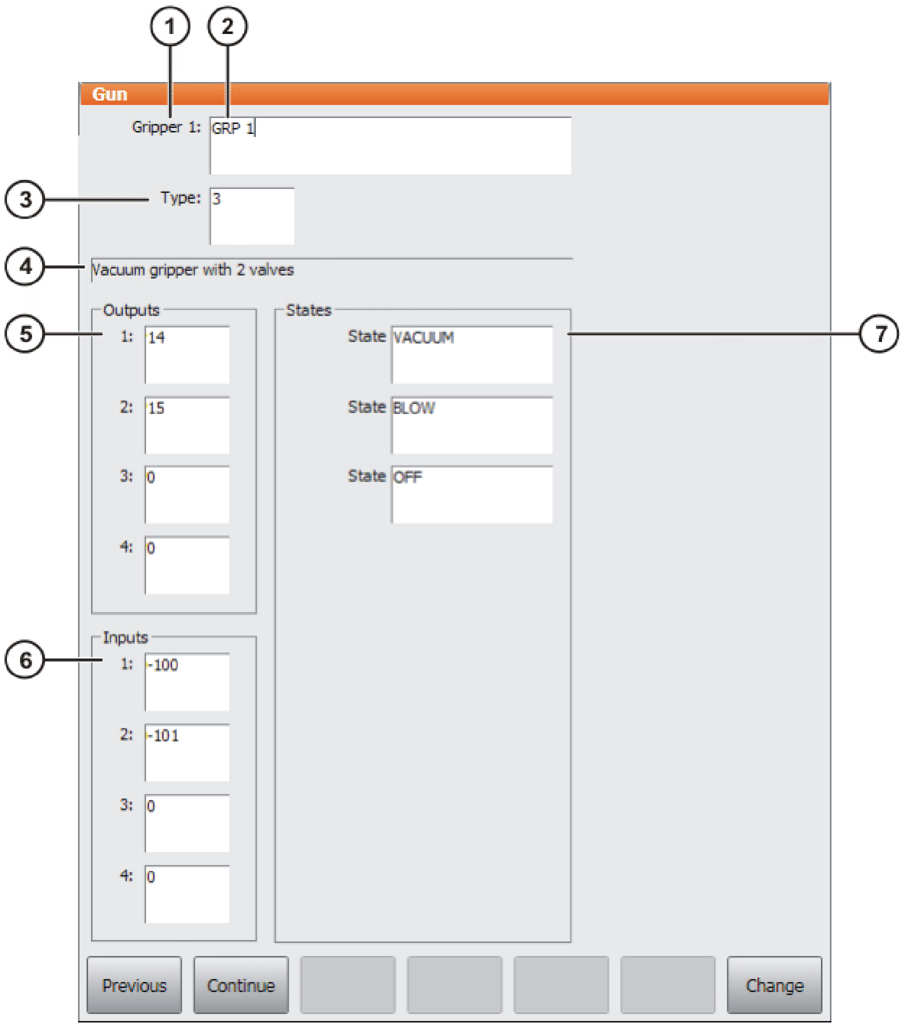
Configuration options and gripper types

KUKA.GripperTech allows gripper configuration by the user. For this, five pre-defined gripper types are available for selection. Additionally, user-defined grippers can also be configured.

NOTICE Up to 16 different grippers can be configured on the controller.

Gripper types

Type	OUT	IN	States	Example
Type 1	2	4	2	Simple gripper with the functions OPEN and CLOSE
Type 2	2	2	3	Slide with center position
Type 3	2	2	3	Vacuum gripper with the functions SUCTION, RELEASE and OFF
Type 4	3	2	3	The same as type 3 but with three control outputs
Type 5	2	4	2	The same as type 1 but with a pulse signal instead of a continuous signal
Not assigned	Freely configurable			



Gripper 1: GRP 1

Type: 3

Vacuum gripper with 2 valves

Outputs

1: 14

2: 15

3: 0

4: 0

Inputs

1: -100

2: -101

3: 0

4: 0

States

State VACUUM

State BLOW

State OFF

Fig. 9-7: Example: predefined gripper

Item	Description
1	Number of the gripper ■ 1 … 16
2	Name of the gripper The name is displayed in the inline form. The default name can be changed. ■ 1 … 24 characters
3	Type ■ For predefined grippers: 1 … 5
4	Designation of the gripper type (not updated until saved) The designation cannot be changed.
5	Assignment of the output numbers Outputs that are not required can be assigned the value “0”. In this way, they are immediately identifiable as unused. If they are none-theless assigned a number, this has no effect.

Item	Description
6	Assignment of the input numbers Inputs that are not required can be assigned the value “0”. In this way, they are immediately identifiable as unused. If they are nonetheless assigned a number, this has no effect.
7	Switching states The default names can be changed. The names are displayed in the inline forms if the corresponding gripper is selected there.

Free gripper types

A freely programmable gripper type has been integrated in order to cover all user requirements. Any number of completely freely definable grippers can be configured by means of entries in the files $CONFIG.DAT, USERGRP.DAT and USER_GRP.SRC.

NOTICE More detailed information about the configuration of grippers can be found in the operating instructions for KUKA System Technology KUKA.Gripper&SpotTech 3.0.

Procedure for gripper configuration

Configuration with predefined gripper type

1. In the main menu, select **Configure** > **I/O** > **Gripper**. A window opens.
2. Select the desired gripper number with **Next** or **Previous**.
3. If desired, change the default name of the gripper.
4. Assign one of the types 1 to 5 to the gripper.
5. Assign the inputs and outputs.
6. If desired, change the default names of the states.
7. Save the configuration with **Change**.

9.4 Exercise: Gripper programming – plastic panel

Aim of the exercise

On successful completion of this exercise, you will be able to carry out the following activities:

- Program statements for controlling and monitoring grippers and weld guns (KUKA.Gripper & SpotTech)
- Activate and work with the technology-specific status keys

Preconditions

The following are preconditions for successful completion of this exercise:

- Theoretical knowledge of the KUKA.Gripper & SpotTech technology package

Task description

Carry out the following tasks: fetch and set down plastic panel

1. Create a new program with the name **Fetch_panel**, using the gripper tool and the blue base.
2. Teach the “Fetch_panel” process with the setdown and pick-up positions illustrated in the figure (>>> Fig. 9-8).
3. Test your program in the modes T1, T2 and Automatic. Observe the relevant safety instructions.
4. Create a second program with the name **Set_down_panel**, using the necessary base and the corresponding tool.
5. Teach the procedure “Set_down_panel”.
6. Test your program in the modes T1, T2 and Automatic. Observe the relevant safety instructions.
7. Archive your programs.

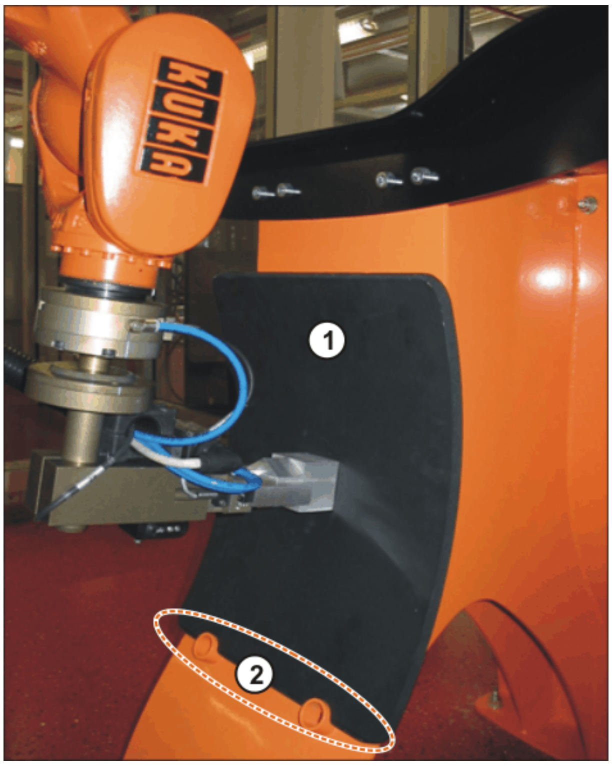


Fig. 9-8: Panel with setdown position

1	Panel	2	Setdown position

Questions on the exercise

1. When should safety instruction be carried out? (min. 4 answers)

. . . .

. . . .

. . . .

. . . .

2. What is the release device on a KUKA robot?

. . . .

. . . .

. . . .

9.5 Exercise: Gripper programming – pen

Aim of the exercise

On successful completion of this exercise, you will be able to carry out the following activities:

- Program statements for controlling and monitoring grippers and weld guns (KUKA.Gripper & SpotTech)
- Activate and work with the technology-specific status keys

Preconditions

The following are preconditions for successful completion of this exercise:

- Theoretical knowledge of the KUKA.Gripper & SpotTech technology package

Task description

Carry out the following tasks: fetch and set down pen 1

1. Create two new programs with the names **Fetch_pen1** and **Set_down_pen1** (duplicate).
2. During programming, make use of the advantages of jogging in the tool direction
3. Make sure that the jog velocity when fetching pens from the pen magazine and setting them down in the pen magazine does not exceed 0.3 m/s
4. Before fetching a pen, carry out a safety query with regard to the gripper position

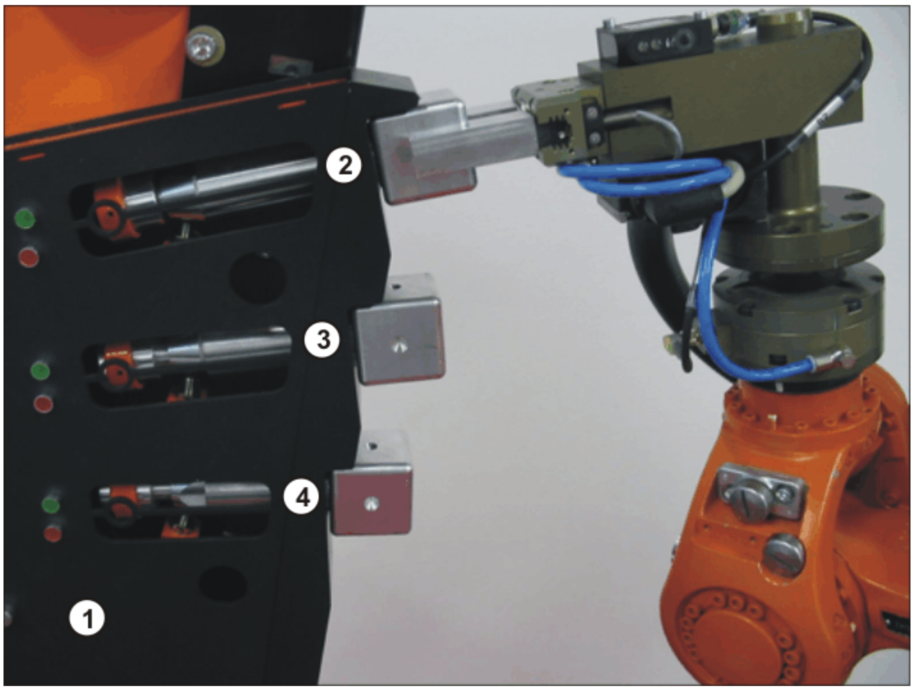


Fig. 9-9: Pen magazine

1	Pen magazine	2	Pen 1
3	Pen 2	4	Pen 3

Questions on the exercise

1. What is the difference between a wait time and “Gripper monitoring ON/ OFF”?

. . . .

. . . .

3. You receive the notification message *Approximation not possible*. What are the possible causes of this?

. . . .

. . . .

. . . .

4. How many standard KUKA gripper types are there?

. . . .

. . . .

10 Successful programming in KRL

10.1 Structure and creation of robot programs

Program execution control

In addition to dedicated motion commands and communication commands (switching functions and wait functions), robot programs also include a large number of routines that can be used for program execution control. This includes:

- **Loops** | Loops are control structures. They go on repeating a block of instructions until a break condition is fulfilled.
  - Endless loops
  - Counting loops
  - Rejecting and non-rejecting loops
- **Branches** | Branches can be used if program sections are only to be executed under certain conditions.
  - Conditional branches
  - Multiple branches

Endless loop

An endless loop repeats the instruction block infinitely. The loop can be exited, however, by means of a premature termination (using the **EXIT** function).

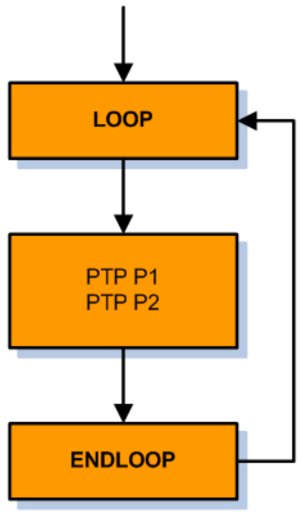


Fig. 10-1: Program flowchart: endless loop

Examples of a LOOP instruction:

- without **EXIT**

  The motion commands to P1 and P2 are executed permanently.

```
LOOP
   PTP P1 Vel=100% PDAT1
   PTP P2 Vel=100% PDAT2
ENDLOOP
```

- with **EXIT**

  The motion commands to P1 and P2 are executed until input 30 is switched to TRUE.

```
LOOP
   PTP P1 Vel=100% PDAT1
   PTP P2 Vel=100% PDAT2
   IF $IN[30]==TRUE THEN
      EXIT
   ENDIF
ENDLOOP
```

Counting loop

The counting loop (FOR loop) can be used to repeat instructions a defined number of times. The number of repetitions is controlled by means of a counting variable.

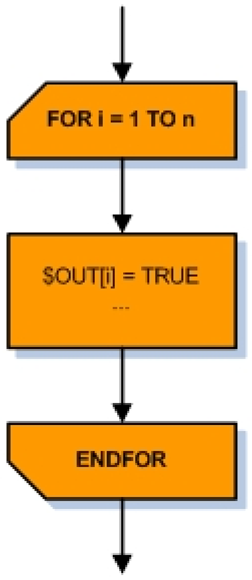


Fig. 10-2: Program flowchart: FOR loop

Example of a FOR loop: Outputs 1 to 5 are consecutively switched to TRUE. The integer variable “i” is used to count the number of times the loop is repeated.

```
INT i
...
FOR i=1 TO 5
   $OUT[i] = TRUE
ENDFOR
```

Rejecting loop

A WHILE loop is a *rejecting* or *pre-test* loop that checks the break condition before the instruction section of the loop is executed.

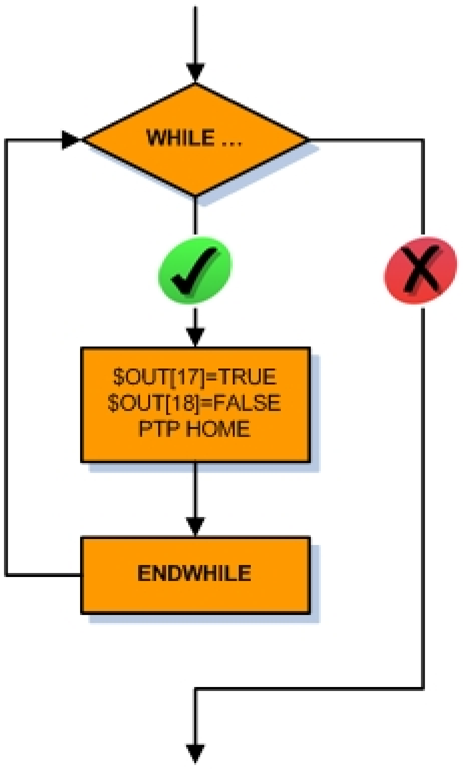


Fig. 10-3: Program flowchart WHILE

Example of a WHILE loop: Output 17 is switched to TRUE, output 18 is switched to FALSE and the robot is moved to the home position, but only if the condition (input 22 is TRUE) is met at the start of the loop.

```
WHILE $IN[22]==TRUE
   $OUT[17]=TRUE
   $OUT[18]=FALSE
   PTP HOME
ENDWHILE
```

Non-rejecting loop

A REPEAT loop is a *non-rejecting* or *post-test* loop that does not check the break condition until after the instruction section of the loop has been executed for the first time.

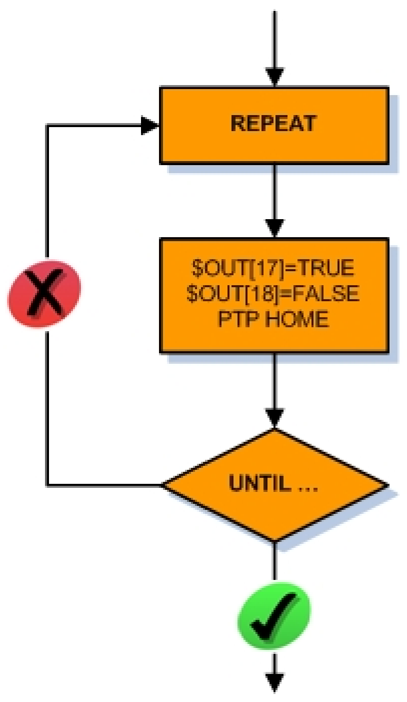


Fig. 10-4: Program flowchart REPEAT

Example of a REPEAT loop: Output 17 is switched to TRUE, output 18 is switched to FALSE and the robot is moved to the home position. Only then is the condition checked.

```
REPEAT
   $OUT[17]=TRUE
   $OUT[18]=FALSE
   PTP HOME
UNTIL $IN[22]==TRUE
```

Conditional branch

A *conditional branch* (IF statement) consists of a condition and two instruction sections. If the condition is fulfilled, the first instruction can be executed. If the condition is not met, the second instruction is executed.

There are also alternatives to IF statements:

- The second instruction section can be omitted: IF statement without ELSE. This means that if the condition is not met, the program is resumed directly after the branch.
- Several IF statements can be nested in each other (*multiple branch*): the statements are executed in sequence until a condition is met.

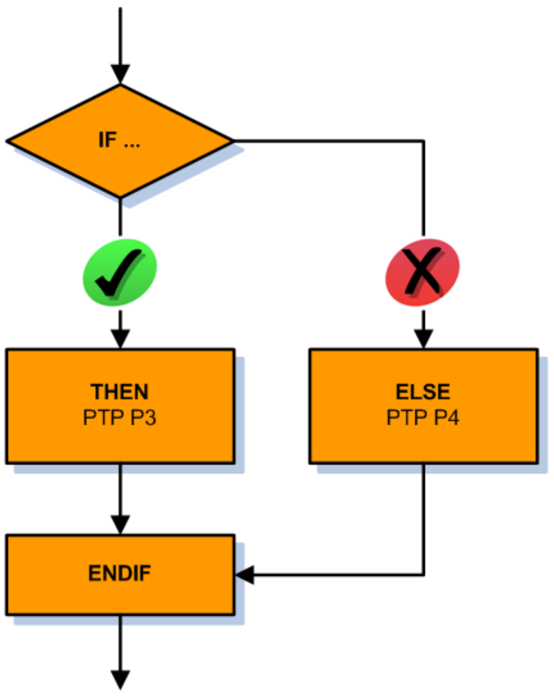


Fig. 10-5: Program flowchart: IF branch

Example of an IF statement: If the condition is met (input 30 must be TRUE), the robot moves to point P3, otherwise to point P4.

```
...
IF $IN[30]==TRUE THEN
   PTP P3
ELSE
   PTP P4
ENDIF
```

Switch statement

A SWITCH branch is a *switch statement* or *multiple branch*. First of all, an expression is evaluated. The value of the expression is then compared with the value of one of the case sections (CASE). If they match, the instructions of the corresponding case are executed.

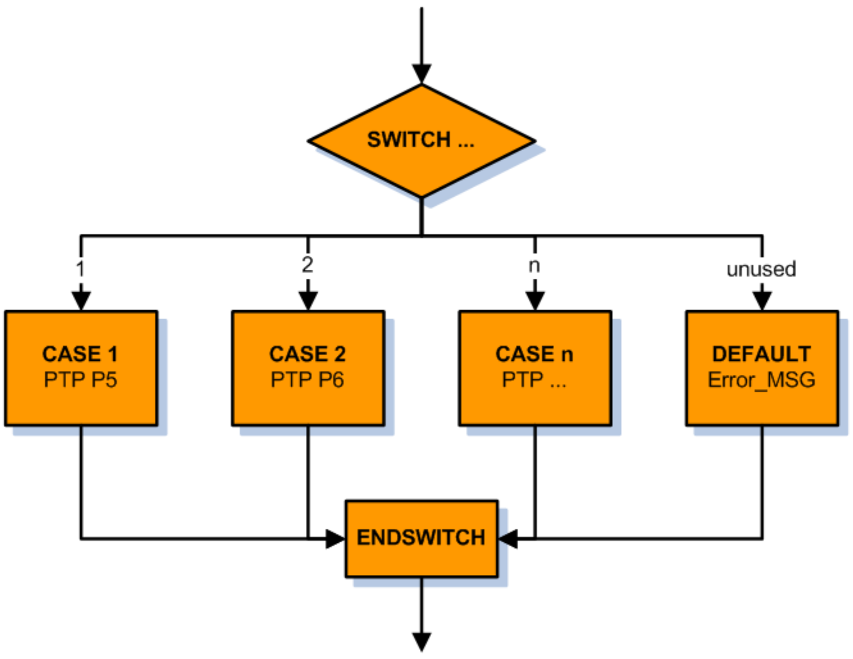


Fig. 10-6: Program flowchart: SWITCH – CASE statements

First of all, the value of the integer variable with the name “status” is checked. If the value of the variable is 1, CASE 1 is executed: the robot moves to point P5. If the value of the variable is 2, CASE 2 is executed: the robot moves to point P6. If the value of the variable is not contained in one of the cases (here: anything other than 1 or 2), the DEFAULT branch is executed: an error message.

```
INT status
...
SWITCH status
   CASE 1
     PTP P5
   CASE 2
     PTP P6
   ...
   DEFAULT
     ERROR_MSG
ENDSWITCH
```

10.2 Structuring robot programs

Robot program structuring options

The structure of a robot program is an important factor for its usability. The more structured a program is, the more comprehensible, effective, legible and economical it will be. The following techniques can be used for structuring a program:

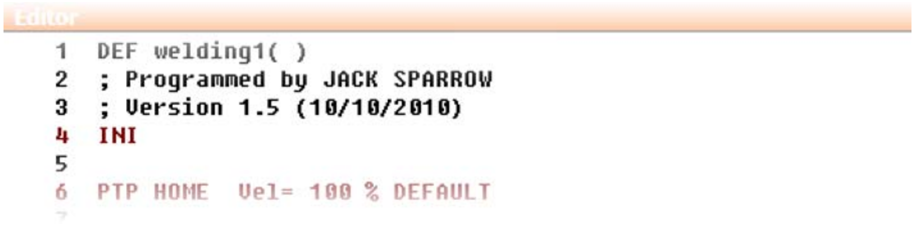
- **Commenting** | Comment and stamp
- **Indentation** | Spaces
- **Hiding** | Folds
- **Module technique** | Subprograms

Comments and stamps

The addition of a comment makes it possible to store a text in the robot program, destined solely for the reader of the program. The text is thus not loaded by the robot interpreter. The text is only there to make the program more legible.

Comments can be used in many different ways in robot programs:

- **Information** about the program text | Author, version, creation date

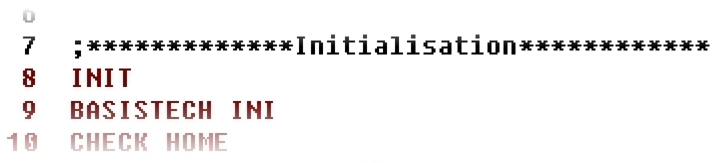


```
DEF welding1( )
; Programmed by JACK SPARROW
; Version 1.5 (10/10/2010)
INI

PTP HOME  Vel= 100 % DEFAULT
```

Fig. 10-7: Example of a comment: Information

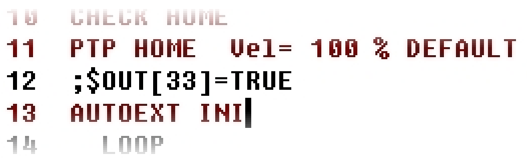
- **Structuring** the program text | Particularly using graphic means (special characters: #, *, ~,)



```
;*************Initialisation************
INIT
BASISTECH INI
CHECK HOME
```

Fig. 10-8: Example of a comment: Structure

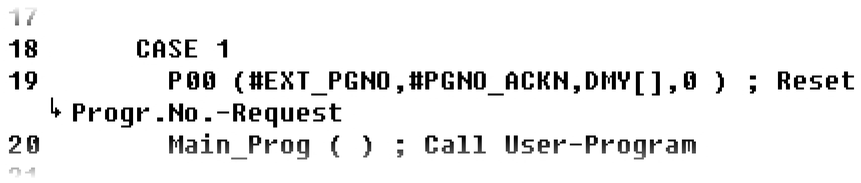
- **Commenting out** (Expert level) | Starting a program line with a semicolon turns it into a comment, i.e. the text is recognized as a comment and is ignored during program execution.



```
CHECK HOME
PTP HOME  Vel= 100 % DEFAULT
;$OUT[33]=TRUE
AUTOEXT INI
  LOOP
```

Fig. 10-9: Example of a comment: Commenting out

- **Explanations** relating to individual lines and **notes about work to be done** | Labeling of inadequate program sections



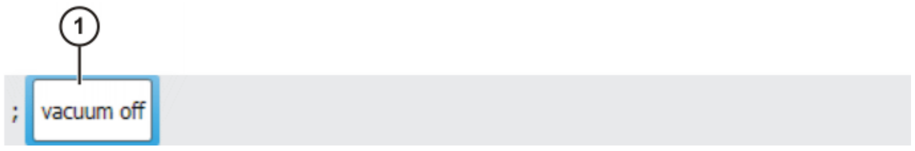
```
    CASE 1
      P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 ) ; Reset
↳ Progr.No.-Request
      Main_Prog ( ) ; Call User-Program
```

Fig. 10-10: Example of a comment: Explanations

NOTICE Comments are only of any use if they are kept up to date. It is vital to ensure that the comments are updated following subsequent modification of the corresponding instructions!

Comments can be inserted in three different ways:

- By inserting a semicolon (Expert level) | If a semicolon (" ; ") is inserted, the following part of the line is turned into a comment.
- Insertion of the inline form "Comment"

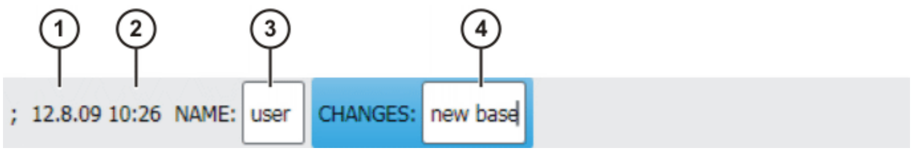


; vacuum off

Fig. 10-11: Inline form "Comment"

Item	Description
1	Any text

- Insertion of the inline form “Stamp” | An additional time stamp is inserted. In addition, the name of the editor can also be inserted.



; 12.8.09 10:26 NAME: user CHANGES: new base

Fig. 10-12: Inline form “Stamp”

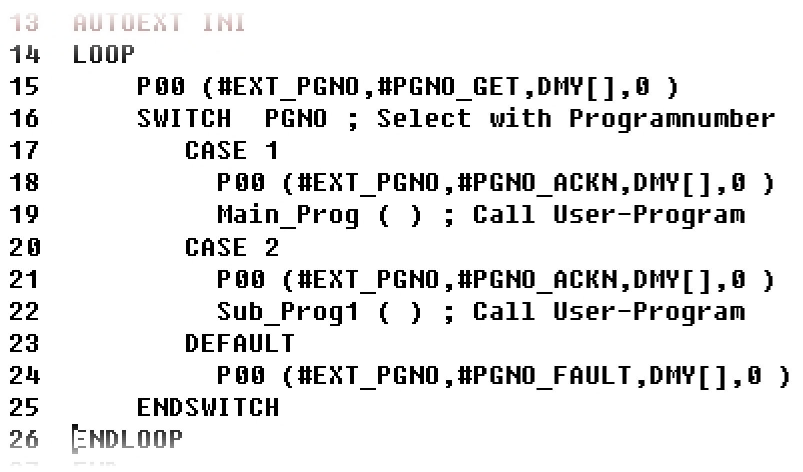
Item	Description
1	System date (cannot be edited)
2	System time (cannot be edited)
3	Name or ID of the user
4	Any text

Procedure for inserting comments and stamps

1. Select the line after which the comment or stamp is to be inserted.
2. Select the menu sequence **Commands** > **Comment** > **Normal** or **Stamp**.
3. Enter the desired data. If a comment or stamp has already been entered previously, the inline form still contains the same entries.
   - In the case of a comment, the box can be cleared using **New text** ready for entry of a new text.
   - In the case of a stamp, the system time can also be updated using **New time** and the **NAME** box can be cleared using **New name**.
4. Save with **Cmd Ok**.

Indenting program lines

Indenting program lines is an effective way of increasing the legibility of a robot program. It highlights program modules that belong together.



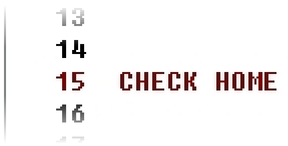
```
AUTOEXT INI
LOOP
    P00 (#EXT_PGNO,#PGNO_GET,DMY[],0 )
    SWITCH  PGNO ; Select with Programnumber
       CASE 1
         P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
         Main_Prog ( ) ; Call User-Program
       CASE 2
         P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
         Sub_Prog1 ( ) ; Call User-Program
       DEFAULT
         P00 (#EXT_PGNO,#PGNO_FAULT,DMY[],0 )
    ENDSWITCH
ENDLOOP
```

Fig. 10-13: Indenting program lines

NOTICE The effect of indentation is purely optical. Indented program sections are processed during program execution in exactly the same way as program sections that are not indented.

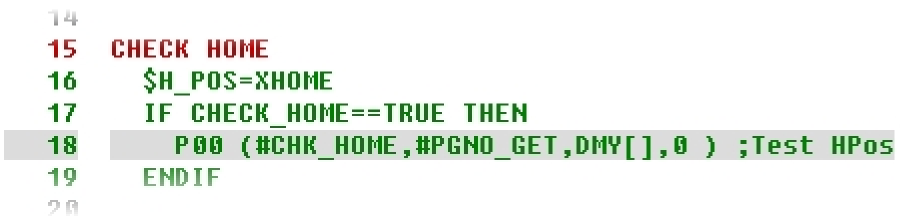
Hiding program lines by means of folds

The KUKA Robot Language offers the possibility of grouping program lines together and hiding them in *folds*. This makes program lines invisible for the user, making the program easier to read. Folds can be opened and edited in the user group Expert.



```
CHECK HOME
```

Fig. 10-14: Closed fold



```
CHECK HOME
  $H_POS=XHOME
  IF CHECK_HOME==TRUE THEN
    P00 (#CHK_HOME,#PGNO_GET,DMY[],0 ) ;Test HPos
  ENDIF
```

Fig. 10-15: Opened 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

10.3 Linking robot programs

Subprograms

The use of subprograms makes it possible to structure robot programs in a modular manner, thus giving them a more efficient layout. The aim is not to write all commands into a program, but to outsource certain sequences, calculations or processes to separate programs.

There are numerous advantages to using subprograms:

- The main program receives a clear structure and is easier to read, as the program length is reduced.
- Subprograms can be developed independently of one another: the programming effort can be shared and error sources are minimized.
- Subprograms can be used repeatedly.

A basic distinction is made between two different types of subprogram:

- Global subprograms

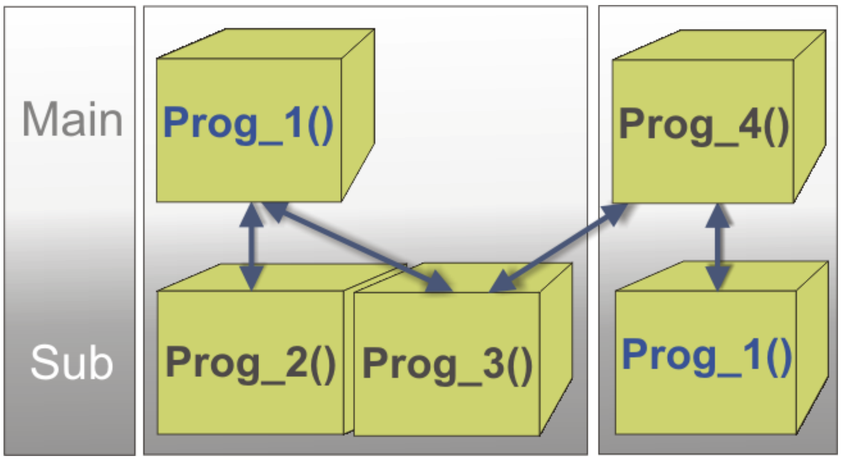


Fig. 10-16: Example diagram for global subprograms

A global subprogram is an independent robot program that is called from a different robot program. The branching of the programs can be requirement-specific, i.e. the program can function as a main program on one occasion and as a subprogram on a different occasion.

- Local subprograms

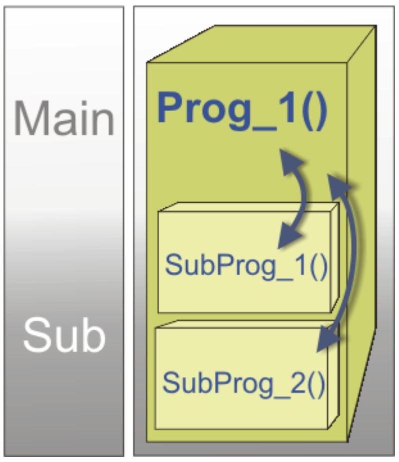
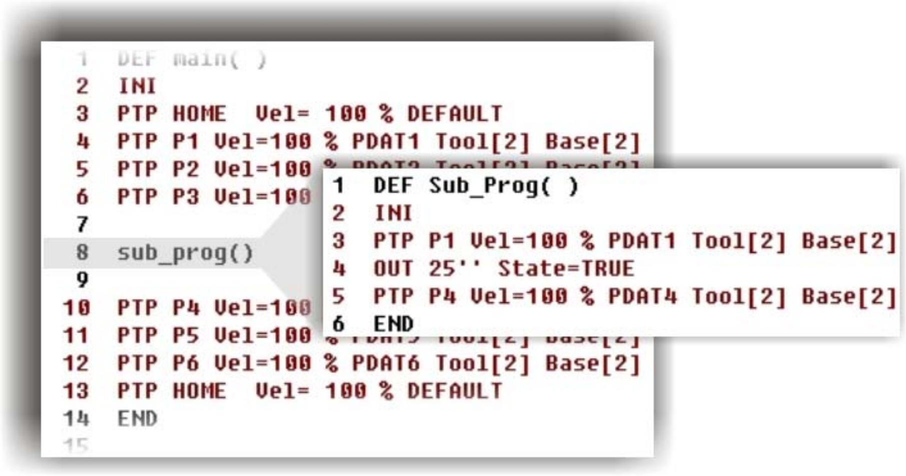


Fig. 10-17: Schematic diagram: local subprograms

Local subprograms are programs that are integrated into a main program, i.e. the commands are contained in the same SRC file. Point coordinates of the subprogram are thus saved in the same DAT file.

Subprogram call sequence

Every program begins with a DEF line and ends with an END line. If a subprogram is called in a main program, the subprogram is executed by default from DEF to END. Once the END line is reached, the program pointer jumps back to the program (main program) from which the subprogram was called.



```
DEF main( )
INI
PTP HOME  Vel= 100 % DEFAULT
PTP P1 Vel=100 % PDAT1 Tool[2] Base[2]
PTP P2 Vel=100 % PDAT2 Tool[2] Base[2]
PTP P3 Vel=100 % PDAT3 Tool[2] Base[2]

sub_prog()

PTP P4 Vel=100 % PDAT4 Tool[2] Base[2]
PTP P5 Vel=100 % PDAT5 Tool[2] Base[2]
PTP P6 Vel=100 % PDAT6 Tool[2] Base[2]
PTP HOME  Vel= 100 % DEFAULT
END
```

```
DEF Sub_Prog( )
INI
PTP P1 Vel=100 % PDAT1 Tool[2] Base[2]
OUT 25'' State=TRUE
PTP P4 Vel=100 % PDAT4 Tool[2] Base[2]
END
```

Fig. 10-18: Subprogram call sequence

NOTICE In order to exit a subprogram prematurely (i.e. before the END line), the command RETURN can be programmed in the subprogram. When this program line is read, the subprogram is terminated prematurely.

Procedure for calling a subprogram

In order to be able to program a subprogram call, the user group Expert must be selected. The **syntax** for a subprogram call is:

```
Name( )
```

1. Select **Configuration** > **User group** in the main menu. The current user group is displayed.
2. Press **Login...** to switch to a different user group. Select the user group **Expert**.
3. Enter the password **kuka** and confirm with **Login**.
4. Load the desired main program into the editor with **Open**.

```
INI

PTP HOME Vel= 100% DEFAULT

PTP HOME Vel= 100% DEFAULT
```

5. Position the cursor in the desired line.
6. Enter the subprogram name with both brackets, e.g. **myprog( )**

```
INI

PTP HOME Vel= 100% DEFAULT
myprog( )
PTP HOME Vel= 100% DEFAULT
```

7. Close the editor by means of the Close icon and save the changes.

10.4 Exercise: Programming in KRL

Aim of the exercise

On successful completion of this exercise, you will be able to carry out the following activities:

- Program endless loops
- Program subprogram calls
- Adapt CELL.SRC for Automatic External mode

Preconditions

The following are preconditions for successful completion of this exercise:

- Knowledge of using the Navigator at Expert level
- Basic knowledge of programming at Expert level (KRL)
- Knowledge of subprogram and loop programming
- Knowledge of the structure of CELL.SRC

Task description

Carry out the following tasks:

1. Create a new module at Expert level with the name **Procedure**. All other programs will be called as subprograms from this central program.
2. The exact program sequence is indicated in the program flowchart. (>>> Fig. 10-19)
3. Test your new program **Procedure** in T1, T2 and Automatic modes. Observe the relevant safety instructions.
4. Extend CELL.SRC as follows:
   - When program number 1 is sent, the pen is fetched from the magazine.
   - When program number 2 is sent, the contour on the table is executed.
   - When program number 3 is sent, the pen is returned to the magazine.
5. Test your CELL.SRC program.

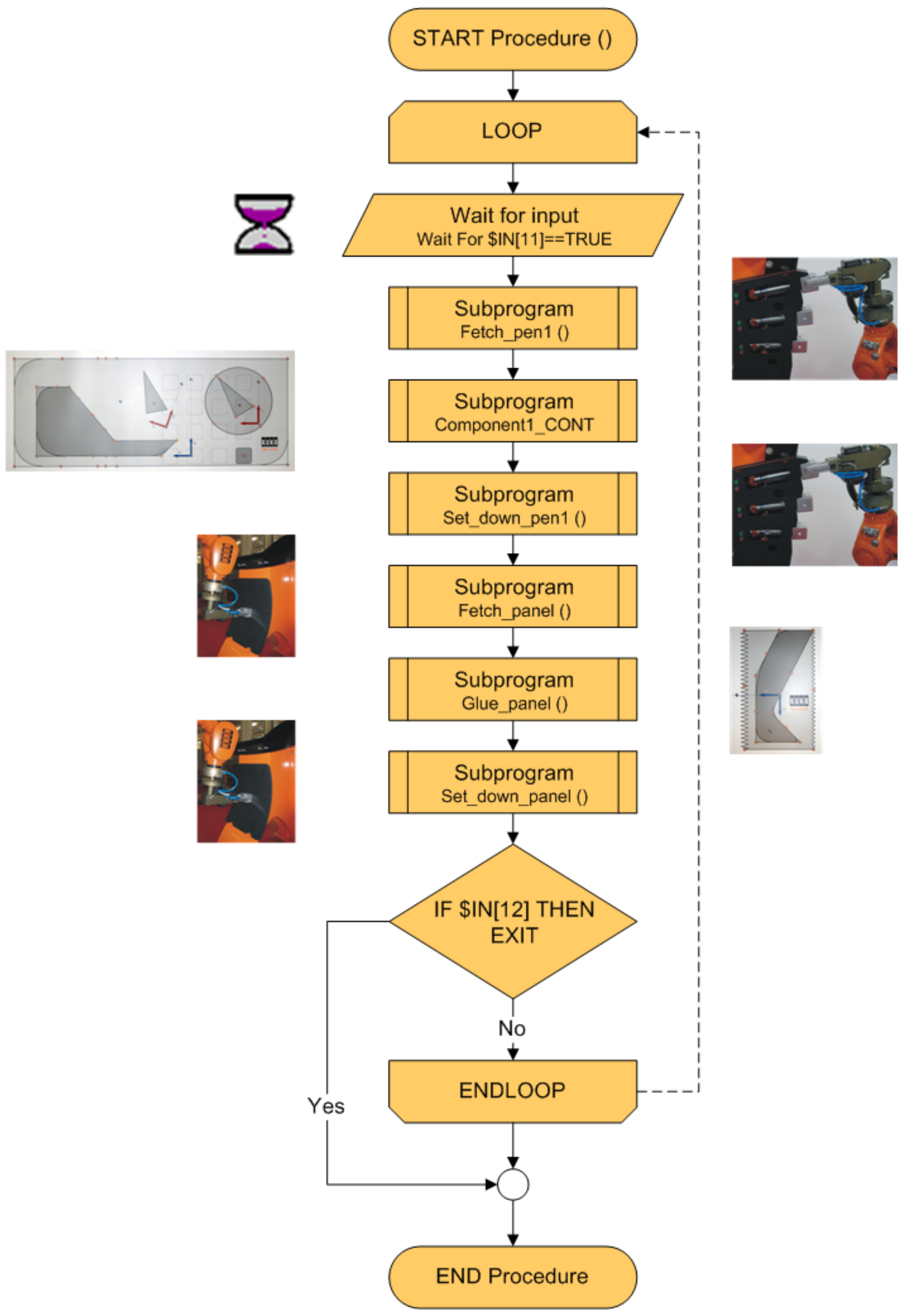


Fig. 10-19: Program flowchart: program Procedure

Questions on the exercise

1. What do the KUKA filename extensions `SRC` and `DAT` mean?

. . . .

. . . .

2. What statement can you use to leave an endless loop?

. . . .

. . . .

3. What syntax is required for a SWITCH statement?

. . . .

. . . .

. . . .

11 Working with a higher-level controller

11.1 Preparation for program start from PLC

Robot in system group

If robot processes are to be controlled centrally (by a host computer or PLC), this is carried out using the Automatic External interface.

Fig. 11-1: PLC connection

System structure principle

The following is required for successful communication between the KR C4 and a PLC:

- **Automatic External mode**: Operating mode in which a host computer or PLC assumes control of the robot system.
- **CELL.SRC**: Organization program for selecting robot programs from outside.
- Signal exchange between PLC and robot: **Automatic External interface** for configuration of the input and output signals:
  - Control signals to the robot (inputs): start and stop signal, program number, error acknowledgement
  - Robot status (outputs): status of drives, position, errors, etc.

Safety instructions – external program start

Once the CELL program has been selected, a BCO run must be carried out.

⚠ WARNING A BCO run is executed as a PTP motion from the actual position to the target position if the selected motion block contains the motion command PTP. If the selected motion block contains LIN or CIRC, the BCO run is executed as a LIN motion. Observe the motion to avoid collisions. The velocity is automatically reduced during the BCO run.

If the BCO run is successful, no further BCO run is performed in the case of the external start.

⚠ 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.

Procedure – external program start

Preconditions

- T1 or T2 operating mode

- Inputs/outputs for Automatic External and the program CELL.SRC are configured.

1. Select the program CELL.SRC in the Navigator. The CELL program is always located in the directory KRC:\R1.
2. Set program override to 100%. (This is the recommended setting. A different value can be set if required.)

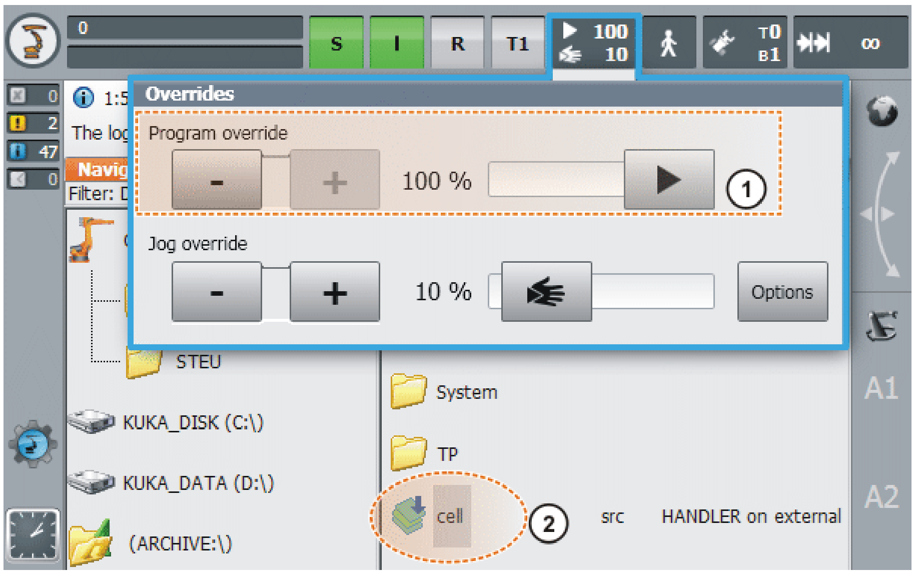


Fig. 11-2: Cell selection and jog override setting

1	Jog override setting
2	Cell.src selection

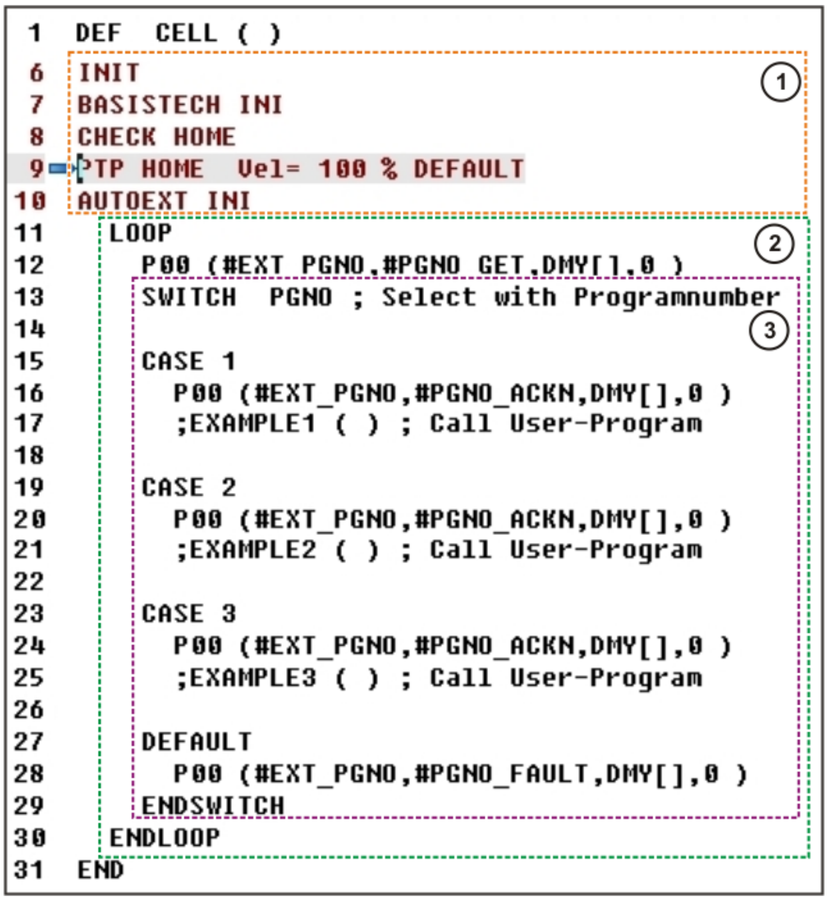
3. Carry out a BCO run:
   Hold down the enabling switch. Then press the Start key and hold it down until the message “Programmed path reached (BCO)” is displayed in the message window.
4. Select “Automatic External” mode.
5. Start the program from a higher-level controller (PLC).

11.2 Adapting the PLC interface (Cell.src)

Cell.src organization program

The organization program Cell.src is used to manage the program numbers transferred by the PLC. It is located in the folder “R1”. Like any other program, the Cell program can be customized, but the basic structure of the program must be retained.

Structure and functionality of the Cell program



```
DEF  CELL ( )
INIT
BASISTECH INI
CHECK HOME
PTP HOME  Vel= 100 % DEFAULT
AUTOEXT INI
  LOOP
    P00 (#EXT_PGNO,#PGNO_GET,DMY[],0 )
    SWITCH  PGNO ; Select with Programnumber

    CASE 1
      P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
      ;EXAMPLE1 ( ) ; Call User-Program

    CASE 2
      P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
      ;EXAMPLE2 ( ) ; Call User-Program

    CASE 3
      P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
      ;EXAMPLE3 ( ) ; Call User-Program

    DEFAULT
      P00 (#EXT_PGNO,#PGNO_FAULT,DMY[],0 )
    ENDSWITCH
  ENDLOOP
END
```

Fig. 11-3: Cell program

1 Initialization and home position

- Initialization of the basic parameters
- Check of the robot position after the home position
- Initialization of the Automatic External interface

2

Endless loop:

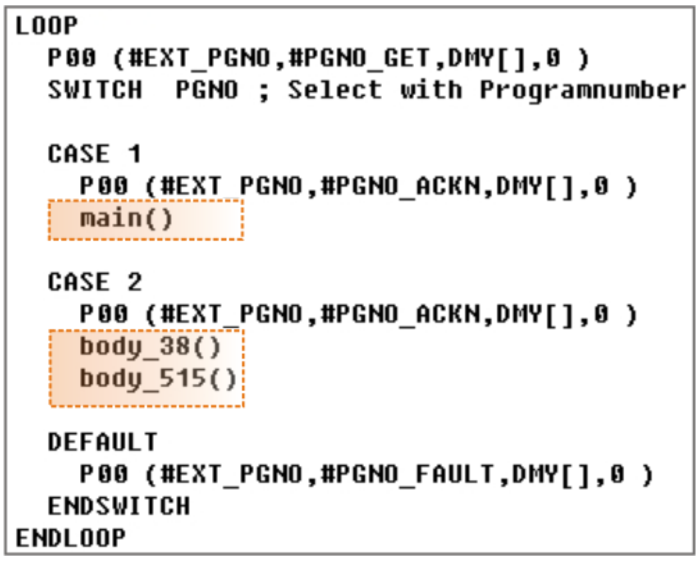
- Polling of program numbers by the "P00" module
- Entry into the selection loop with the program number determined.

3 Program number selection loop

- Depending on the program number (stored in the variable “PG-NO”), the program jumps to the corresponding branch (“CASE”).
- The robot program entered in the branch is then executed.
- Invalid program numbers cause the program to jump to the default branch.
- Once executed, the loop is repeated.

Procedure

1. Switch to the user group “Expert”.
2. Open CELL.SRC.
3. In the “CASE” sections, replace the name “EXAMPLE” with the name of the program that is to be called via the respective program number. Delete the semicolon in front of the name.



```
LOOP
  P00 (#EXT_PGNO,#PGNO_GET,DMY[],0 )
  SWITCH  PGNO ; Select with Programnumber

  CASE 1
    P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
    main()

  CASE 2
    P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
    body_38()
    body_515()

  DEFAULT
    P00 (#EXT_PGNO,#PGNO_FAULT,DMY[],0 )
  ENDSWITCH
ENDLOOP
```

Fig. 11-4: Example of an adapted Cell program

4. Close the program and save the changes.

Index