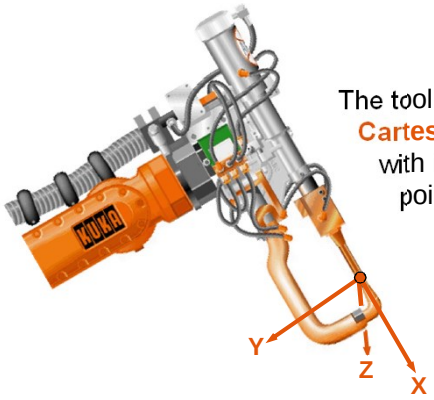
What happens during tool calibration?



The tool receives a user-defined **Cartesian coordinate system** with its origin at a reference point specified by the user.

What are the advantages of tool calibration?

Orientation

CP velocity

Tool direction

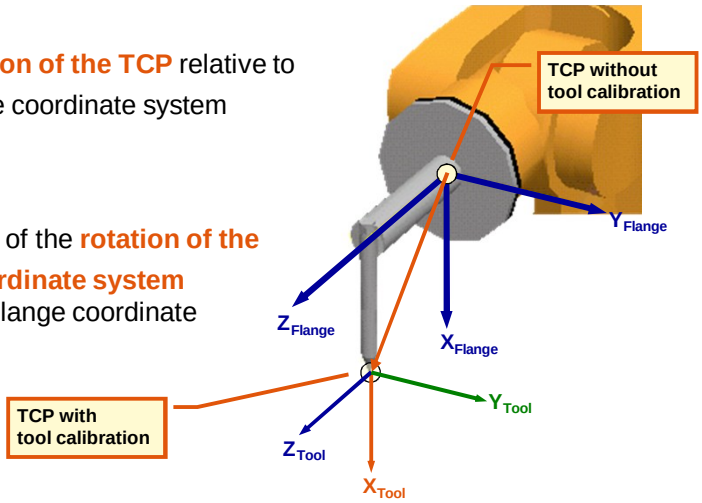
General procedure for tool calibration

1st step:

Calculation of the TCP relative to the flange coordinate system

2nd step:

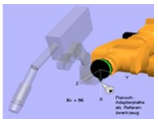
Definition of the **rotation of the Tool coordinate system** from the flange coordinate system



Tool calibration methods

1. TCP calibration

or

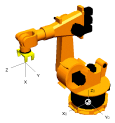


X Y Z - 4-Point

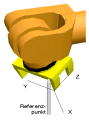
X Y Z - Reference

2. Orientation calibration

or



or

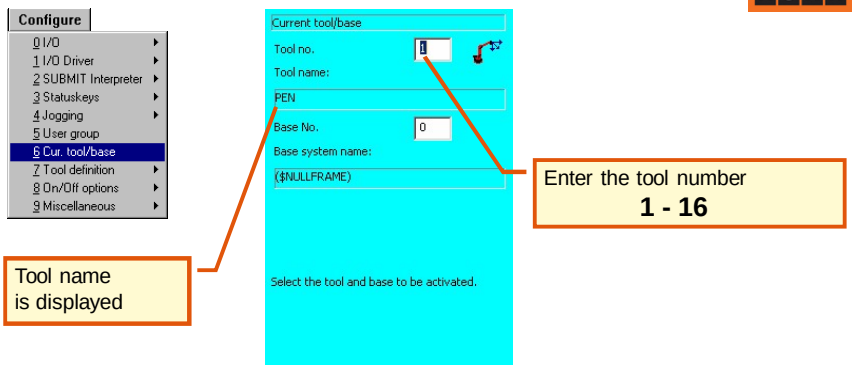


A B C - World 5D

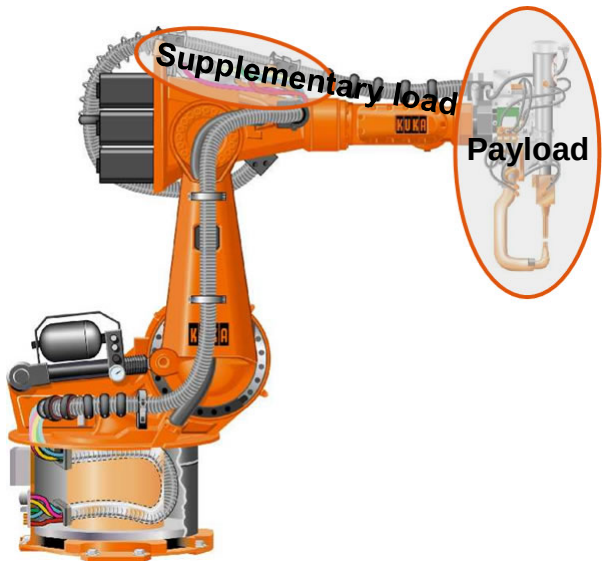
A B C - World 6D

A B C - 2-Point

Activating the tool

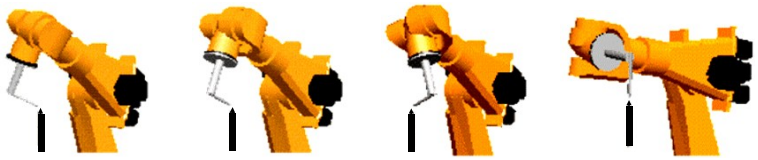


Load data



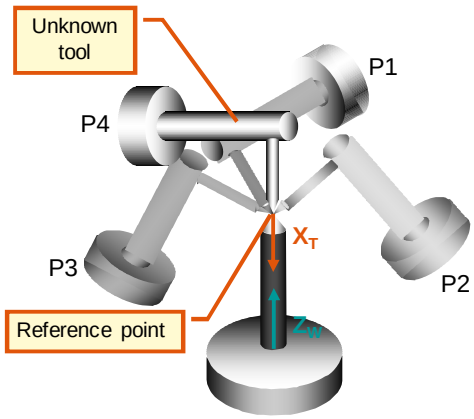
The X Y Z - 4 Point method

In the XYZ - 4-point method, the TCP of the tool is moved to a reference point from four different directions.



The TCP of the tool is then calculated from the different flange positions and orientations.

Diagram of the X Y Z - 4 Point method

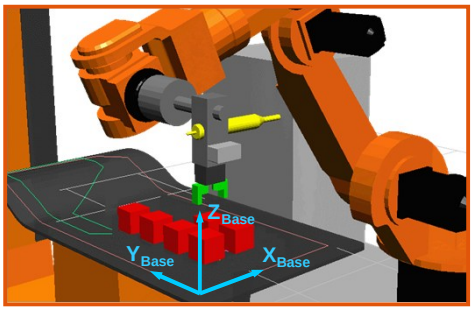


- Move the tool to the reference point with 4 different orientations (P1 to P4).
- **Tip:** Set the final orientation (P4) so that $+X_T$ runs in the direction of $-Z_W$.
- **Important:** The orientations of the tool positions (flange positions) must differ sufficiently from one another.

Reduce the velocity in the vicinity of the reference point in order to avoid a collision.

Base calibration

The work surface (pallet, clamping table, workpiece...) receives a user-defined Cartesian coordinate system with its origin at a reference point specified by the user.

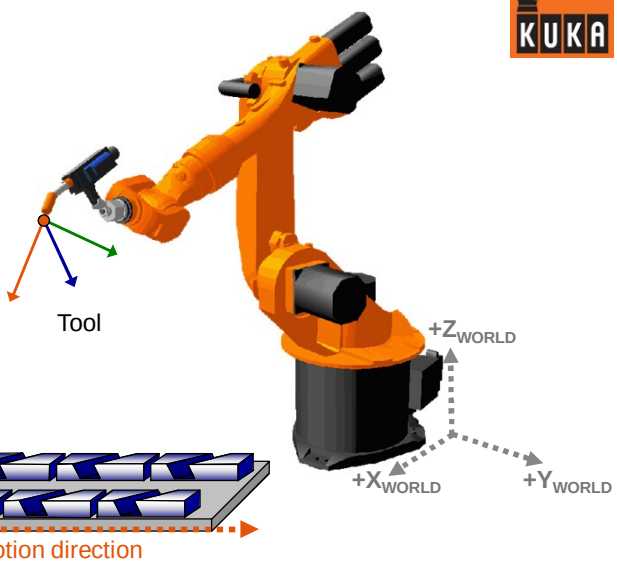


Purpose of base calibration

Jogging along the edges of the work surface or workpiece.

Jogging using the jog keys

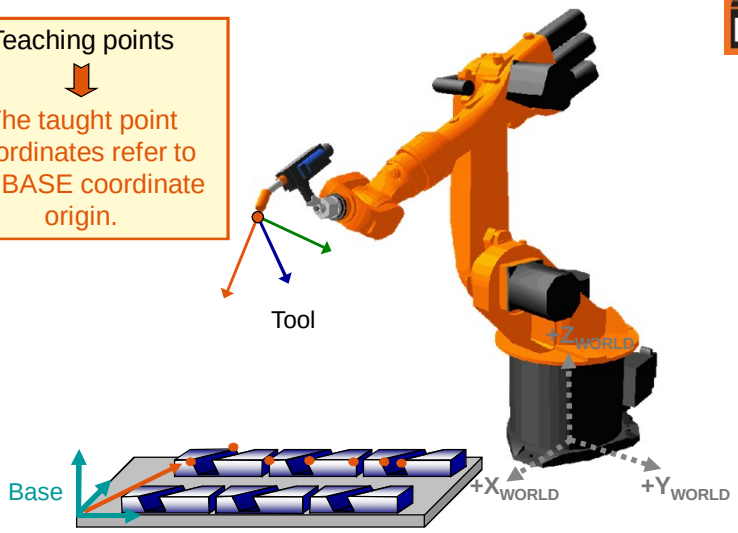
or using the 6D mouse



Purpose of base calibration

Teaching points

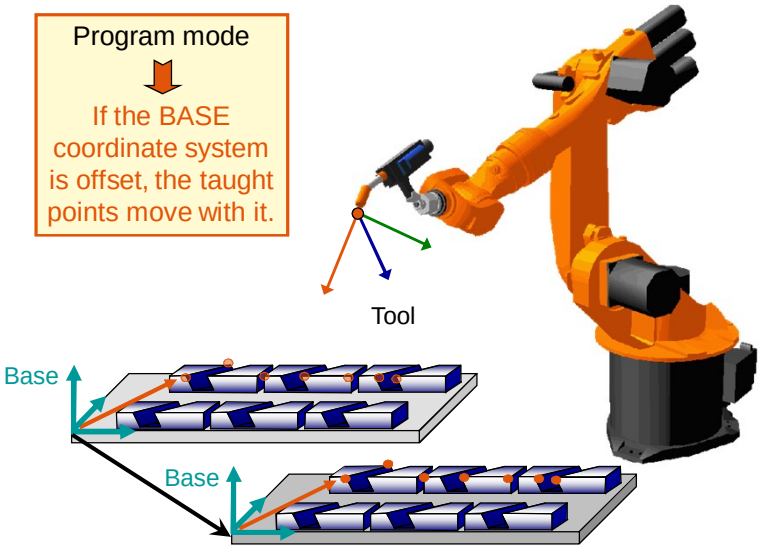
The taught point coordinates refer to the BASE coordinate origin.



Purpose of base calibration

Program mode

If the BASE coordinate system is offset, the taught points move with it.

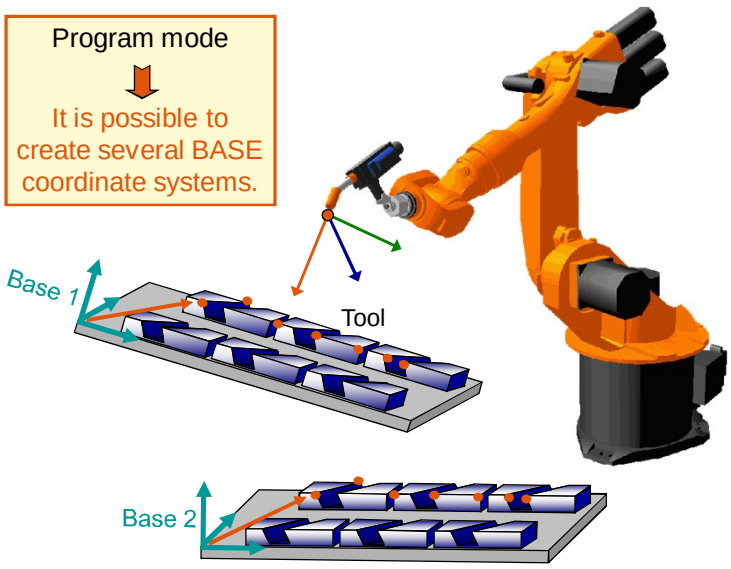


Purpose of base calibration

Program mode

⬇

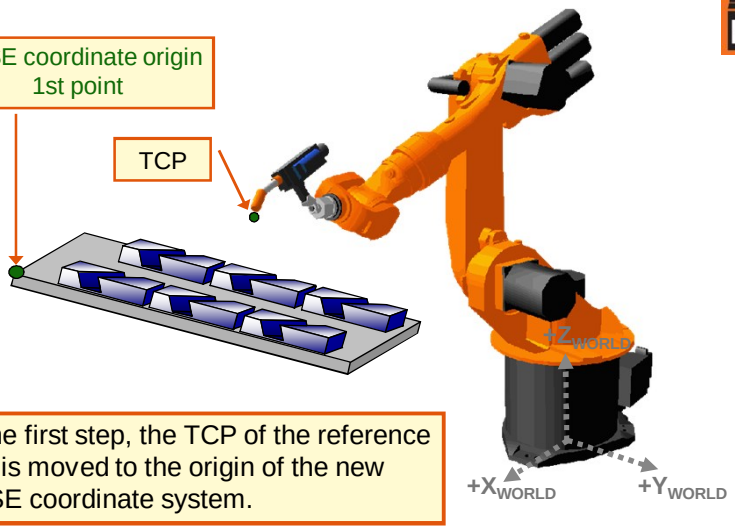
It is possible to create several BASE coordinate systems.



The "3-Point" method 1st step

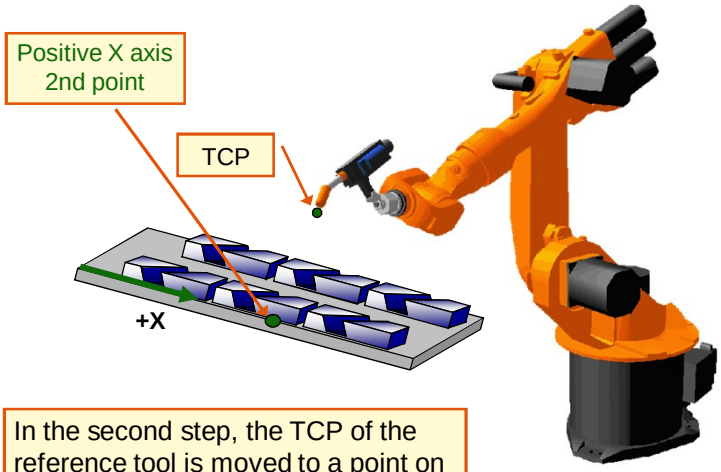
BASE coordinate origin 1st point

TCP



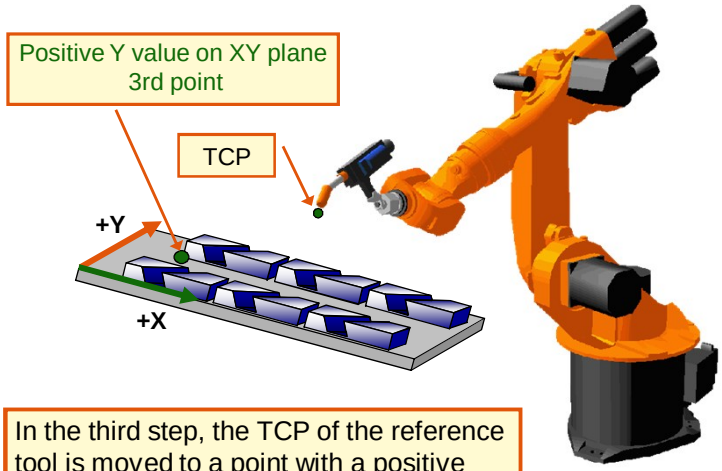
In the first step, the TCP of the reference tool is moved to the origin of the new BASE coordinate system.

The "3-Point" method *2nd step*



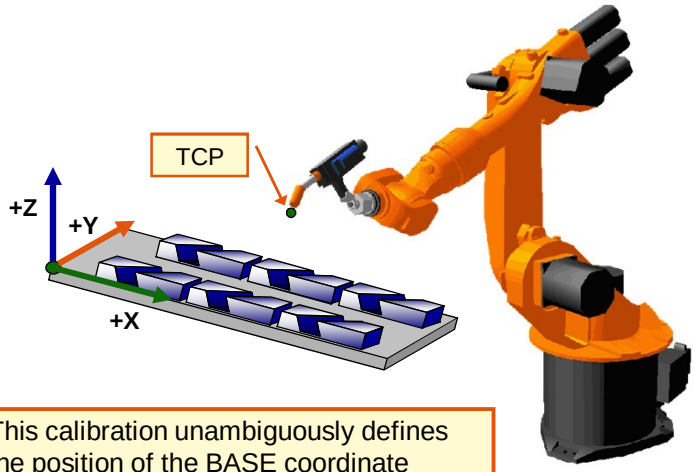
In the second step, the TCP of the reference tool is moved to a point on the positive X axis of the new BASE coordinate system.

The "3-Point" method *3rd step*



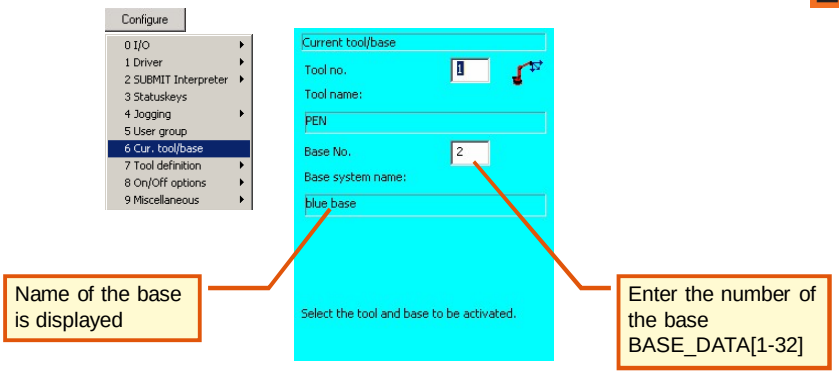
In the third step, the TCP of the reference tool is moved to a point with a positive Y value on the XY plane of the new BASE coordinate system.

Calibrated base



This calibration unambiguously defines the position of the BASE coordinate system.

Activating a base

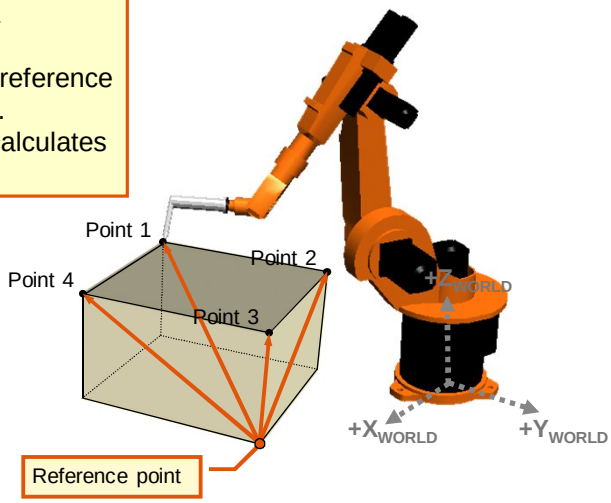


Name of the base is displayed

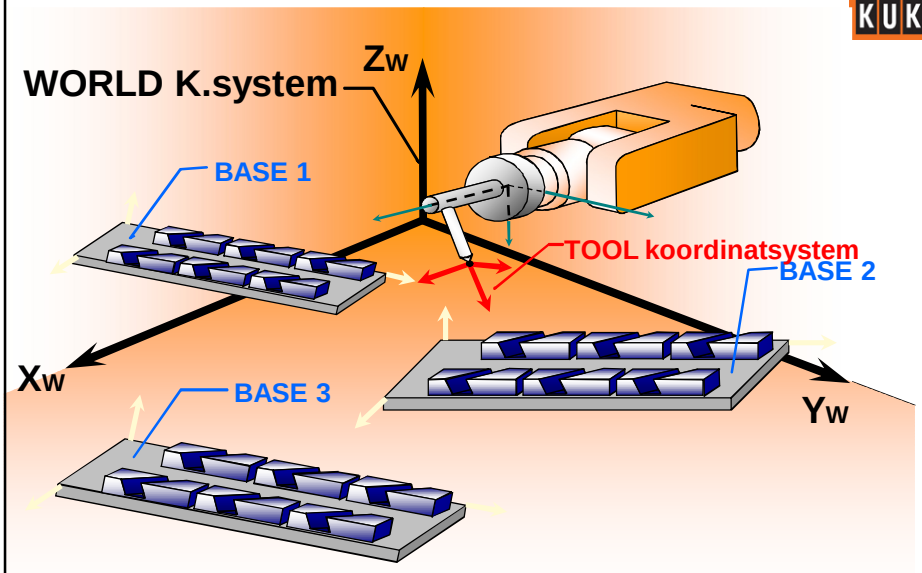
Enter the number of the base BASE_DATA[1-32]

Calculating the BASE coordinate system indirectly

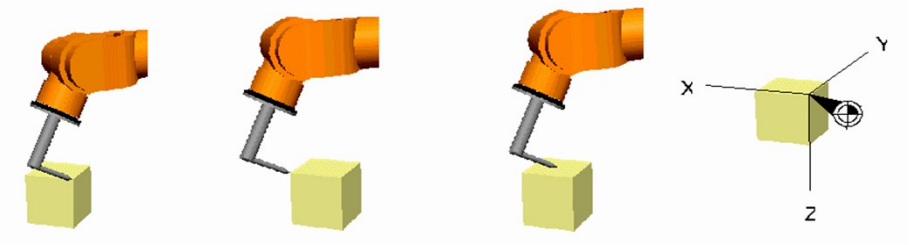
- In the controller, enter the coordinates of 4 points referring to the BASE (e.g. from CAD).
- Move the TCP of the reference tool to the four points.
- The robot controller calculates the BASE.



Base oppmåling



3-punkts metoden



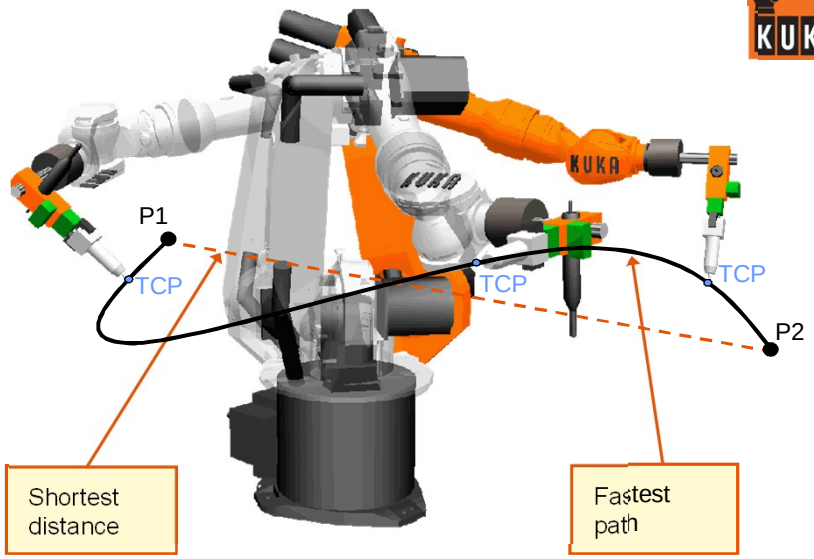
Origo (nullpunkt)

Punkt langs X-aksen

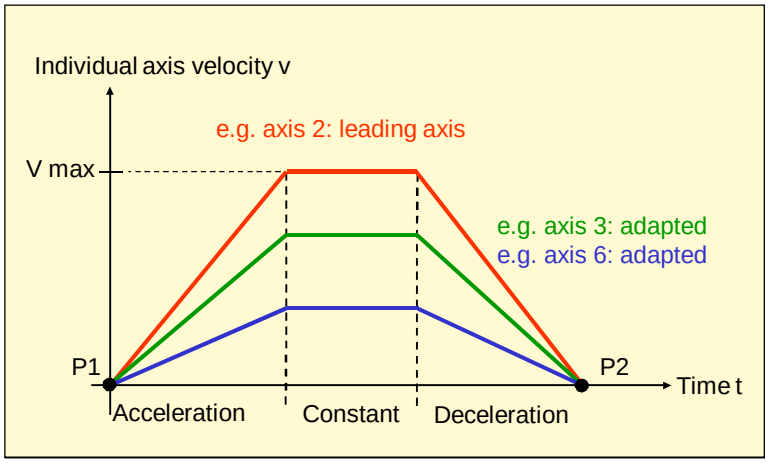
Punkt i XY-planet med positiv Y-verdi

Resultat

PTP - motion



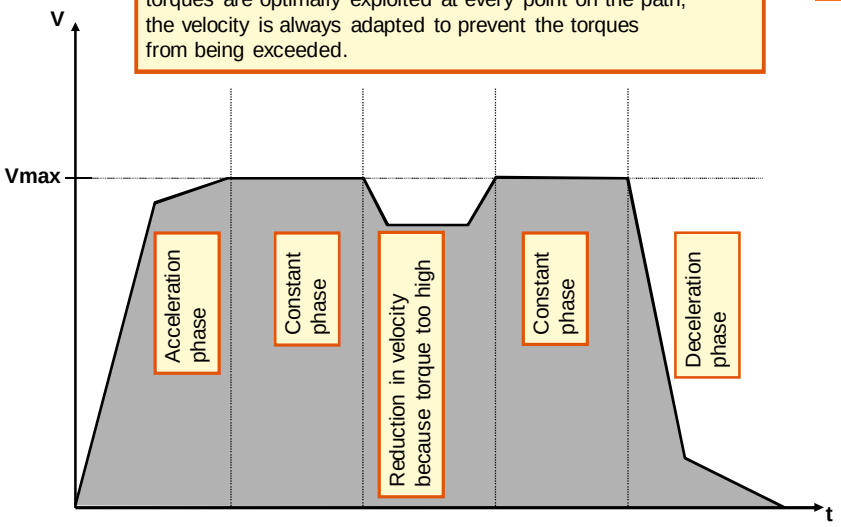
SYNCHRO - PTP



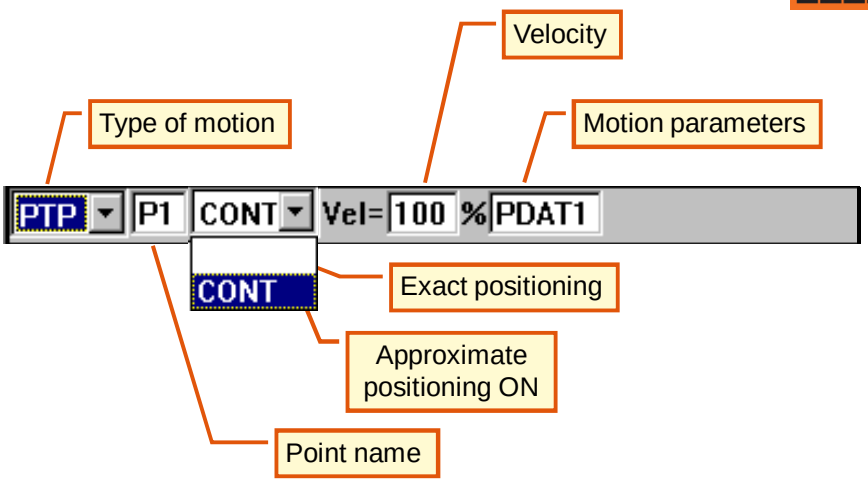
Leading axis is the name given to the axis that takes longest to reach the end point; the velocity specified in the inline form is disregarded.

PTP motion – higher motion profile

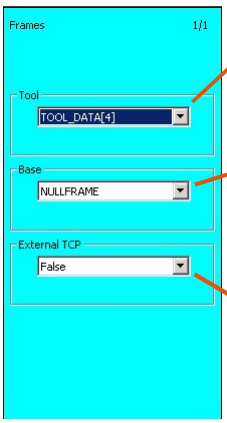
The higher motion profile is used by default, i.e. the permissible torques are optimally exploited at every point on the path; the velocity is always adapted to prevent the torques from being exceeded.



Programming a PTP motion



Programming a PTP motion

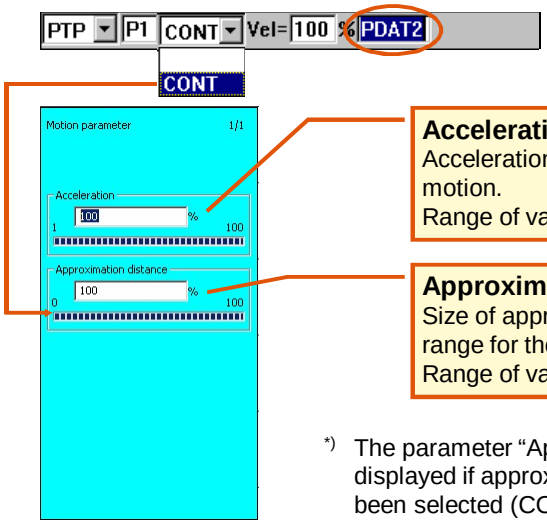


Tool
Tool selection
Tool_Data[1]..[16], Nullframe

Base
Workpiece coordinate system selection
Base_Data[1]..[32], Nullframe

External TCP
Robot guiding tool: False
Robot guiding workpiece: True

Programming a PTP motion



Acceleration
Acceleration used for the motion.
Range of values: 1..100%

Approximation distance*)
Size of approximate positioning range for the motion.
Range of values: 0..100%

*) The parameter “Approximation distance” is displayed if approximate positioning has been selected (CONT).

Status and Turn

The entries “S” and “T” serve to select a specific, unambiguously defined robot position where several different axis positions are possible for the same point in space.

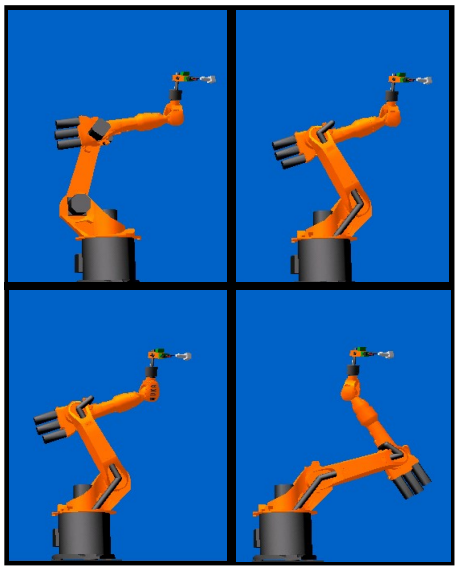
The specification of Status and Turn is only evaluated for PTP motions. For this reason, the first motion in a program must be a PTP motion.

```
DEFDAT MAIN_PROGRAM ( )

DECL POS XPOINT1={X 900, Y 0, Z 800, A 0, B 0, C 0, S 6, T 27}
DECL FDAT FPOINT1...
...

ENDDAT
```

STATUS

TURN

Status and Turn



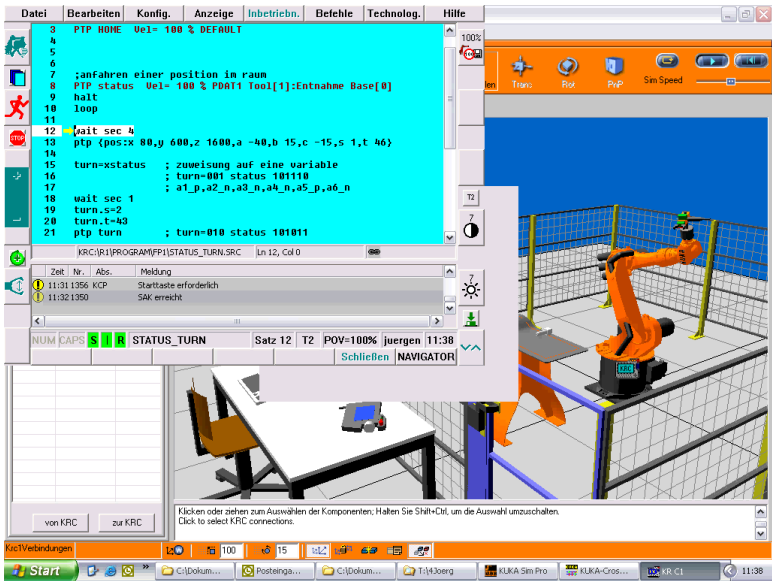
Status = 1
Turn = 46

Status = 2
Turn = 43

Status = 6
Turn = 59

Status = 4
Turn = 63

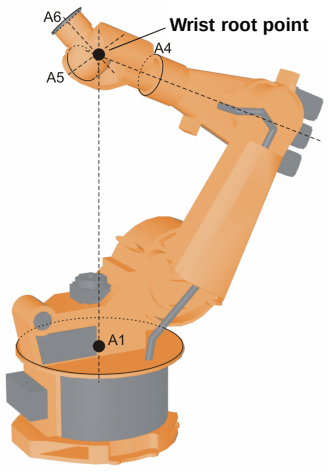
Status and Turn



Singularities

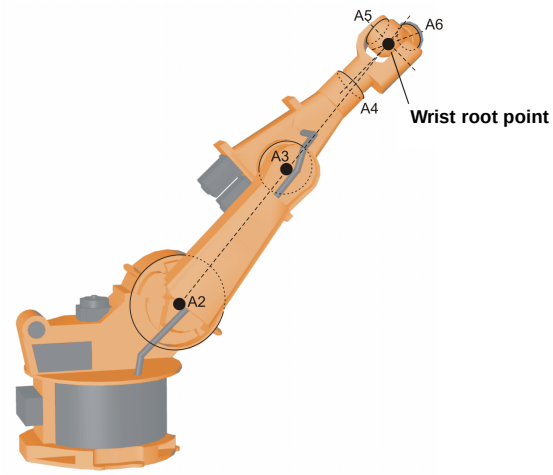
In a standard 6-axis kinematic system, a distinction is made between 3 different singularity positions. These are the overhead singularity, the extended position and the wrist axis singularity. One characteristic of a singularity is that unambiguous reverse transformation (conversion of Cartesian coordinates to axis-specific values) is not possible, even though Status and Turn are specified. Small Cartesian changes in the immediate vicinity of a singularity give rise to major changes in the axis angles.

Overhead singularity (α1 position)



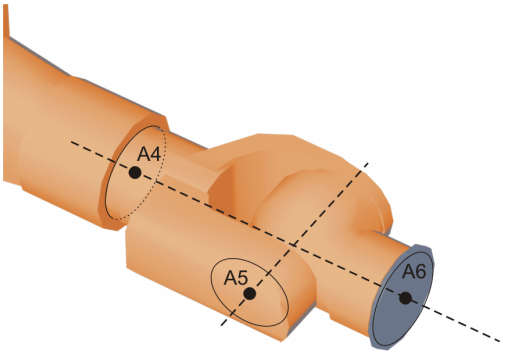
Here, the wrist root point, located at the intersection of axes A4, A5 and A6, is positioned directly on axis 1.

Extended position (α2 position)



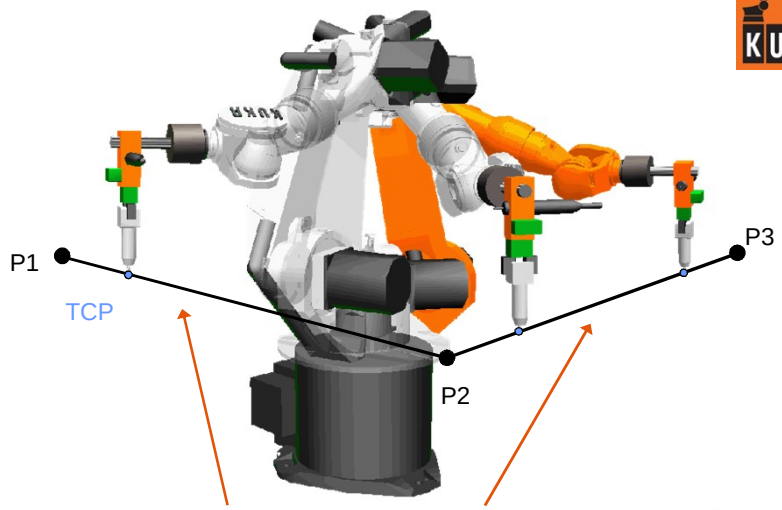
The extension of A2-A3 intersects the wrist root point.

Wrist axis singularity (α5 position)



In this case, axes 4 and 6 are parallel. It is not possible to determine the positions of these two axes unambiguously by means of reverse transformation as there is an infinite number of axis positions for A4 and A6.

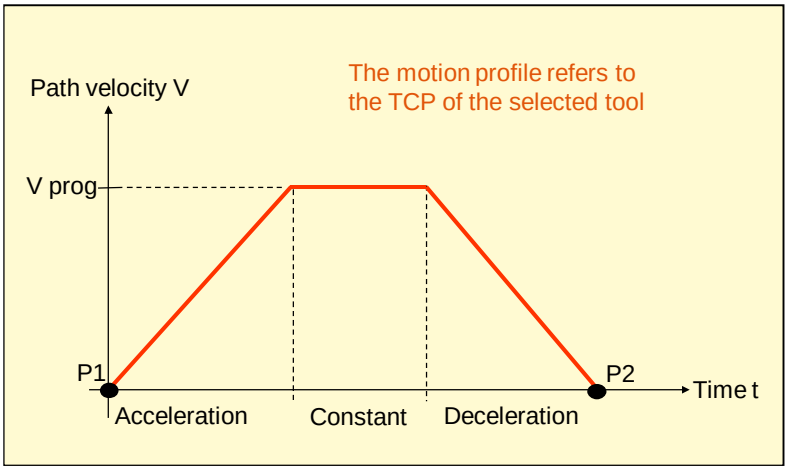
LIN - motion



The TCP is moved along a straight line to the end point

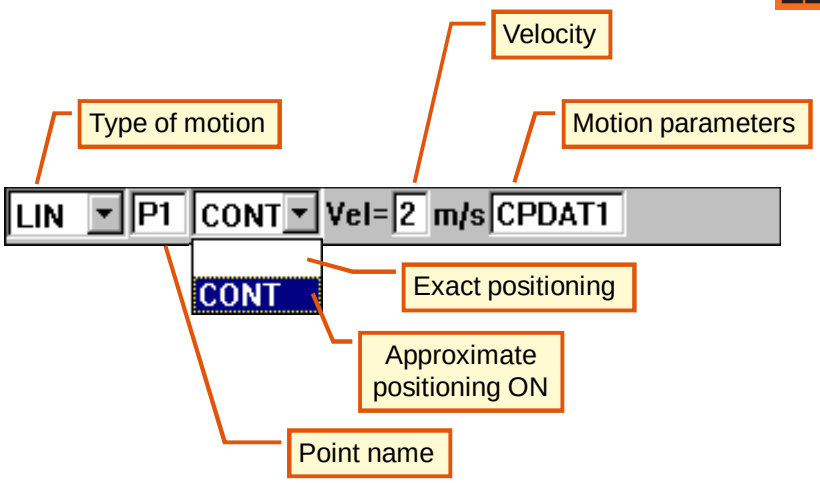
KUKA Roboter GmbH, Hery Park 3000, D-86368 Gersthofen, Tel.: +49 (0) 8 21/45 33-1906, Fax: +49 (0) 8 21/45 33-2340, http://www.kuka-roboter.de

Velocity profile

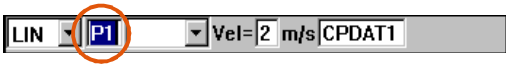


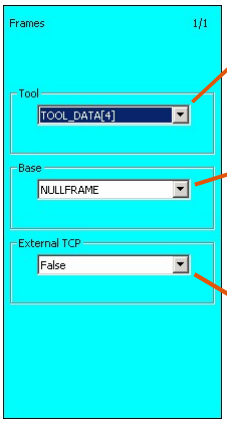
KUKA Roboter GmbH, Hery Park 3000, D-86368 Gersthofen, Tel.: +49 (0) 8 21/45 33-1906, Fax: +49 (0) 8 21/45 33-2340, http://www.kuka-roboter.de

Programming a LIN motion



Programming a LIN motion



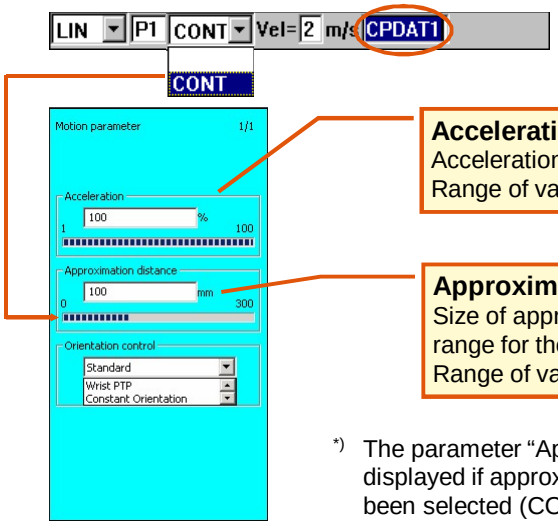


Tool
Tool selection
Tool_Data[1]..[16], Nullframe

Base
Workpiece coordinate system selection
Base_Data[1]..[32], Nullframe

External TCP
Robot guiding tool: False
Robot guiding workpiece: True

Programming a LIN motion

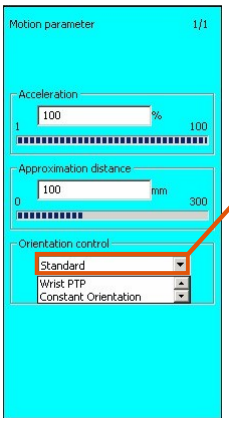


Acceleration
Acceleration used for the motion.
Range of values: 1...100%

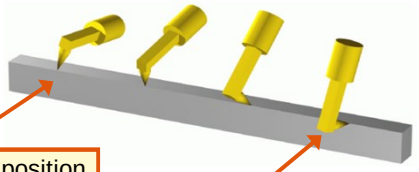
Approximation distance*)
Size of approximate positioning range for the motion.
Range of values: 0...300 mm

*) The parameter “Approximation distance” is displayed if approximate positioning has been selected (CONT).

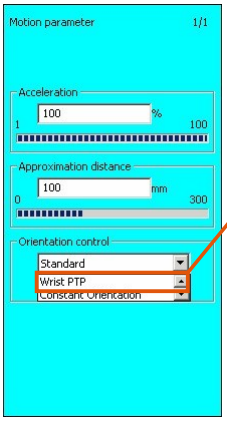
Programming a LIN motion



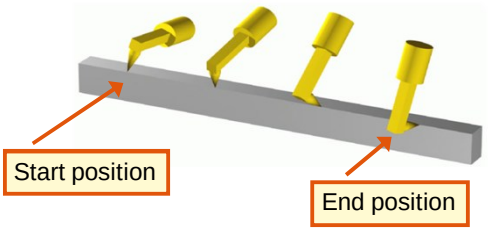
Orientation control - Standard



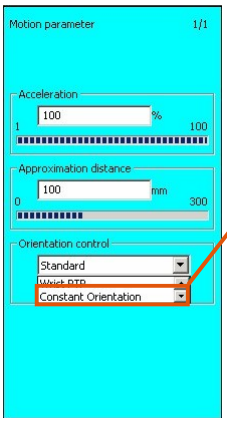
During the path motion, the orientation of the tool changes continuously from the start position to the end position. This is achieved by rotating and pivoting the tool direction.



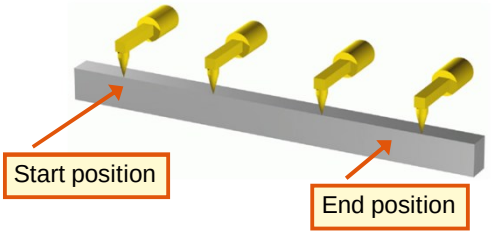
Orientation control - Wrist PTP



During the path motion, the orientation of the tool changes continuously from the start position to the end position. This is done by linear transformation (axis-specific motion) of the wrist axis angles. The problem of the wrist singularity can be avoided using this option as there is no orientation control by rotating and pivoting the tool direction.



Orientation control - Constant

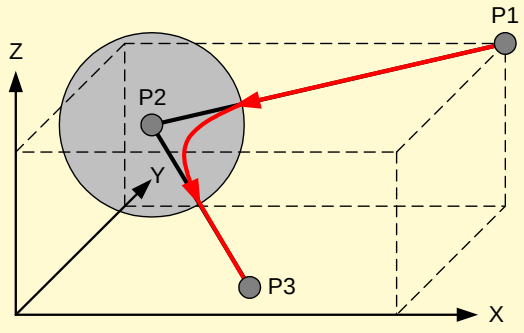


The orientation remains constant during the CP motion. The programmed orientation is disregarded for the end point and that of the start point is used.

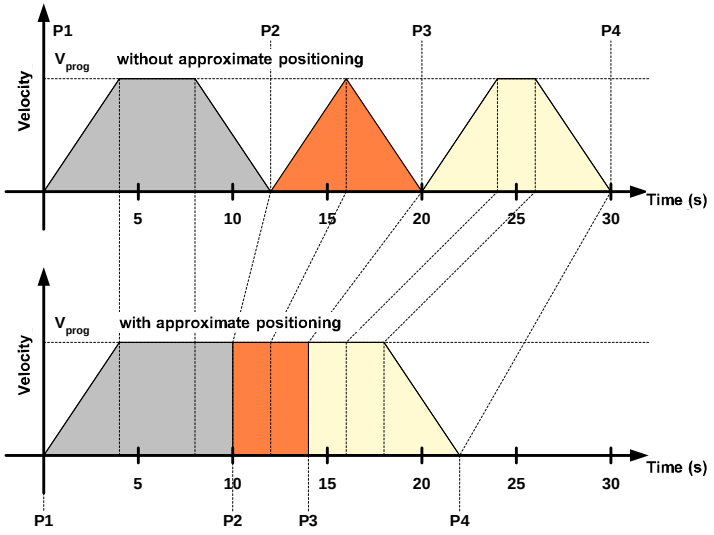
Approximation of motions

During approximate positioning, the robot does not move exactly to each programmed position, nor is it braked completely.

Advantage:
- reduced wear
- improved cycle times

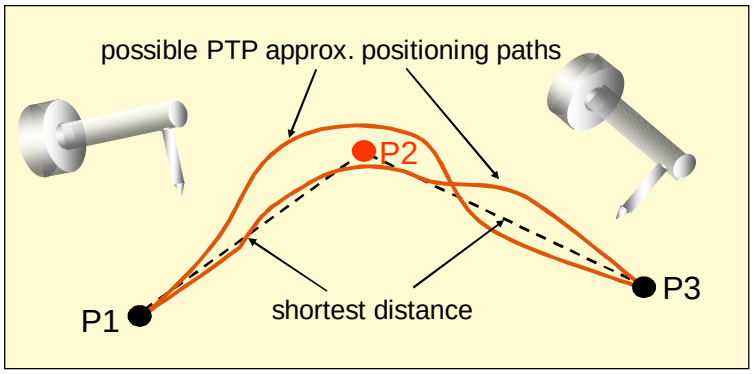


Cycle time improvement by means of approximated motions

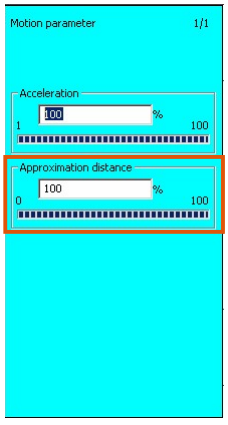


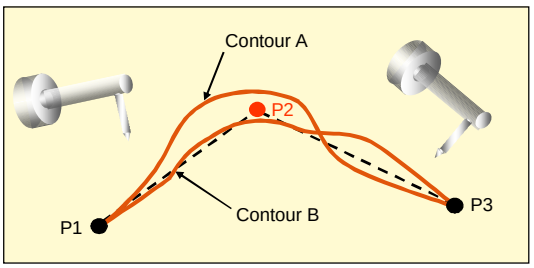
PTP motion with approximate positioning

PTP motion <u>with</u> approx. positioning → P2 is an approx. positioning point



PTP motion with approximate positioning

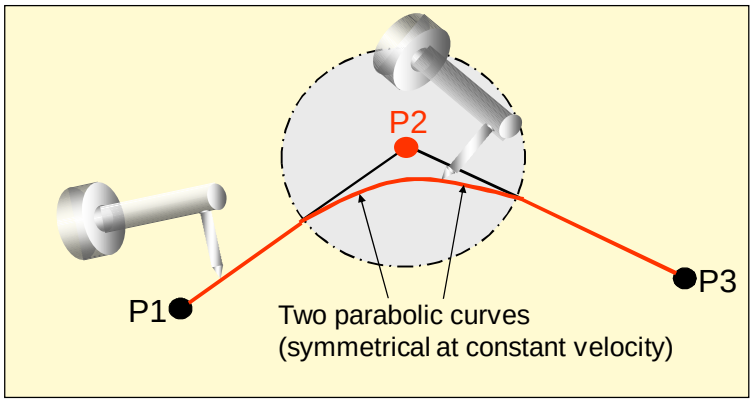




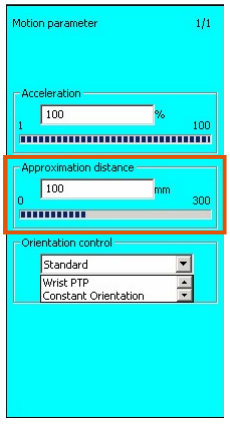
The value of “Approximation distance” specifies the size of the approximate positioning range. The path **cannot** be set, nor is it predictable.

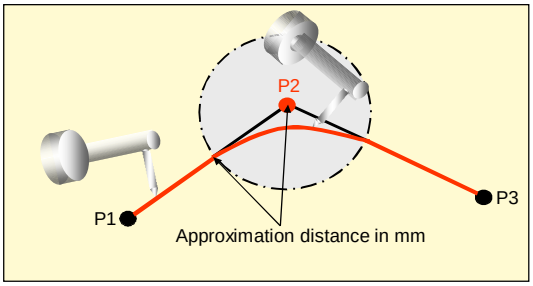
LIN motion with approximate positioning

LIN motion <u>with</u> approx. positioning → P2 is an approx. positioning point



LIN motion with approximate positioning

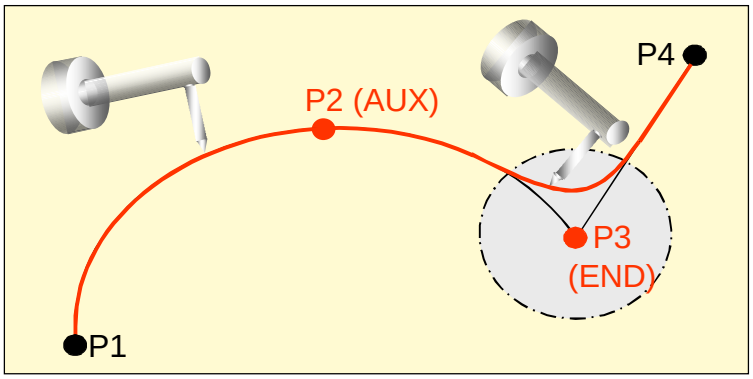




The value entered for “Approximation distance” specifies the distance from the end point and the point at which the approximation motion commences. The resulting path is **not** an arc. The same applies to the following CIRC command.

CIRC motion with approximate positioning

CIRC motion with approx. positioning → P3 is an approx. positioning point



Computer advance run

What is the computer advance run?

The **main run pointer** (white bar), which can be seen on the graphical user interface when the program is running, always indicates the block that is currently being processed. The **advance run pointer**, on the other hand, is not visible and **precedes** the **main run pointer** by three motion blocks (default setting).

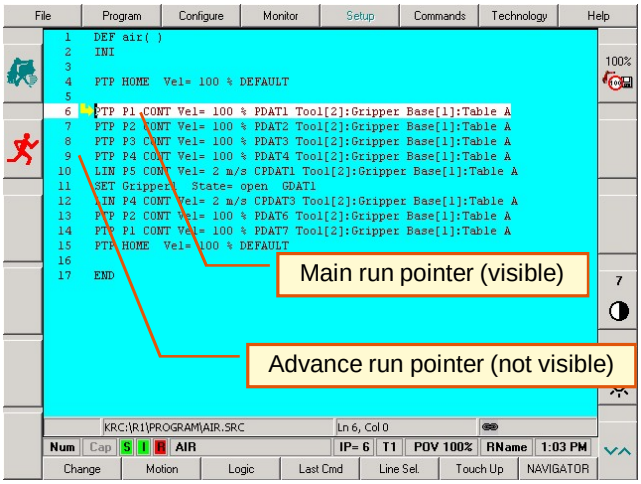
What is the function of the advance run pointer?

In order to be able to calculate the path, e.g. of an approximation motion, it is necessary to read the path planning data using the advance run pointer. It is not only motion data that are processed, however, but also arithmetical data and commands for controlling the periphery.

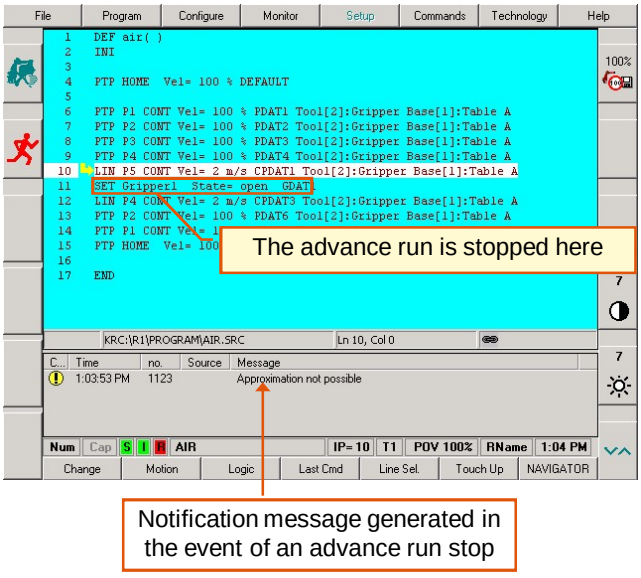
How is the advance run pointer influenced?

Instructions and data that influence the periphery (e.g. input/output instructions) trigger an advance run stop. If the advance run pointer is stopped, approximate positioning cannot be carried out.

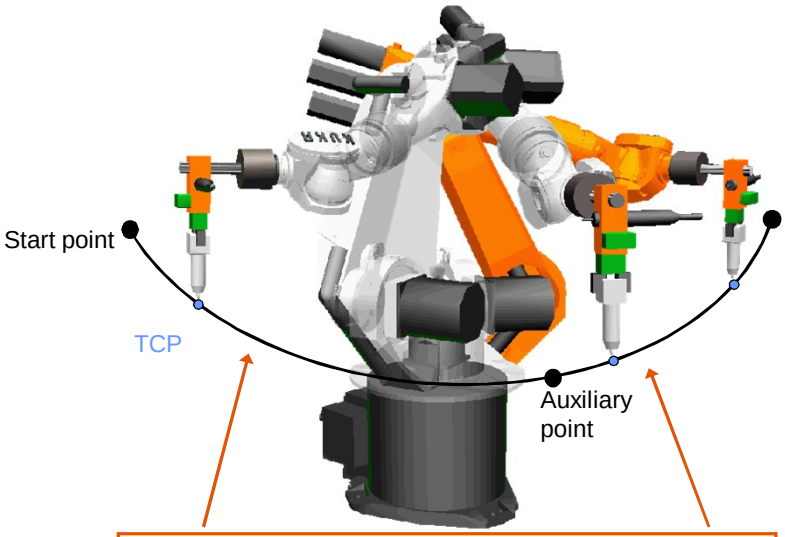
Computer advance run



Computer advance run



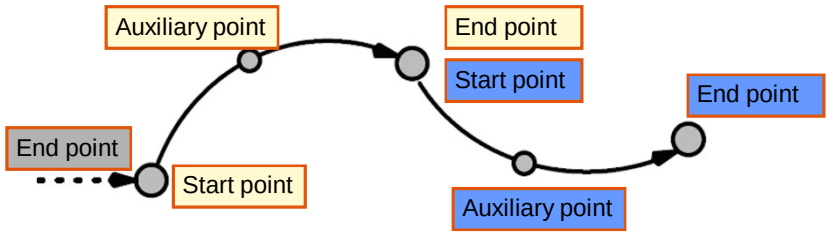
CIRC - motion



The TCP is moved along an arc to the end point

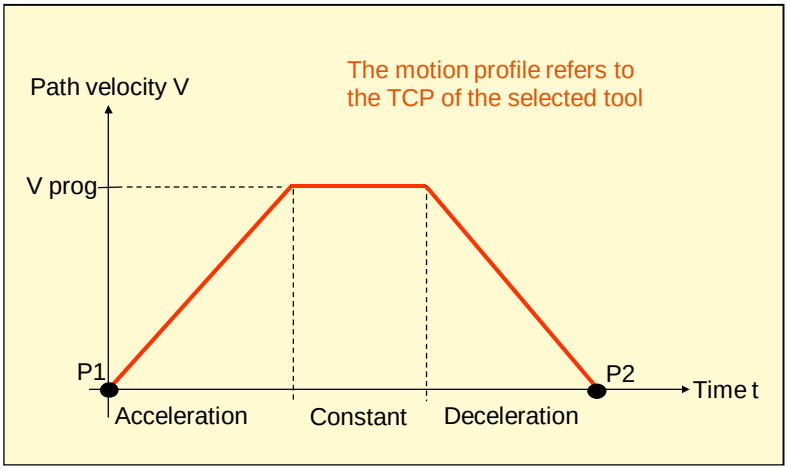
CIRC motion

Here, the TCP or workpiece reference point moves to the end point along an arc. The path is defined using start, auxiliary and end points. The end point of a motion instruction serves as the start point for the subsequent motion.

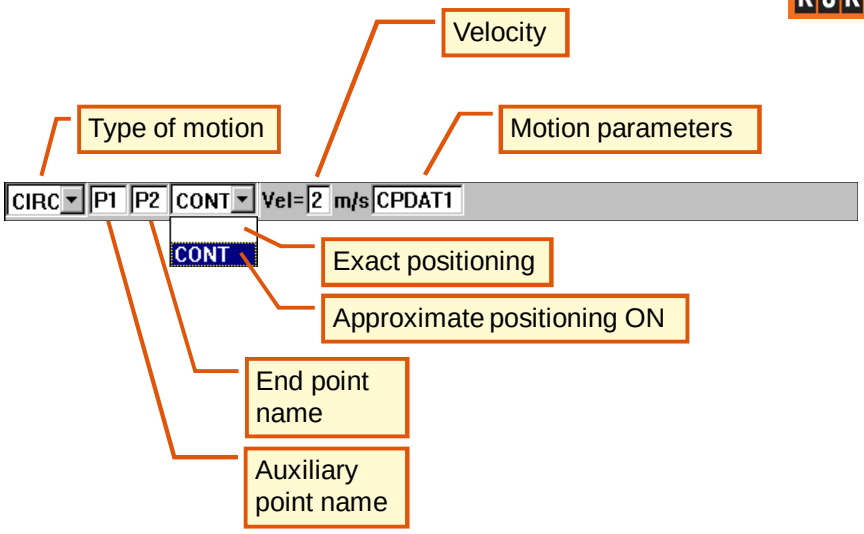


The orientation of the TCP is not taken into consideration at the auxiliary point and is not relevant for the teaching of coordinates.

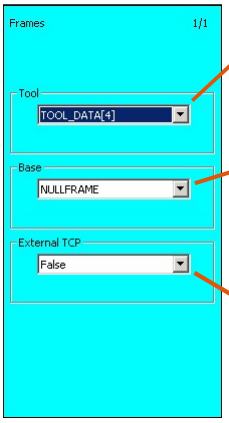
Velocity profile



Programming a CIRC motion



Programming a CIRC motion

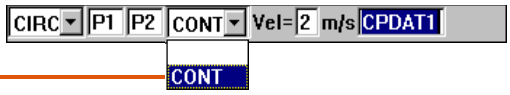


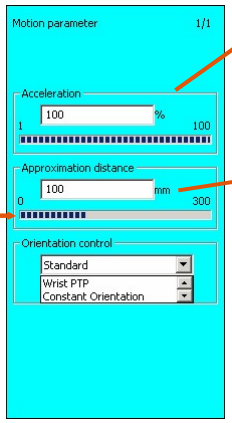
Tool
Tool selection
Tool_Data[1]..[16], Nullframe

Base
Workpiece coordinate system selection
Base_Data[1]..[32], Nullframe

External TCP
Robot guiding tool: False
Robot guiding workpiece: True

Programming a CIRC motion

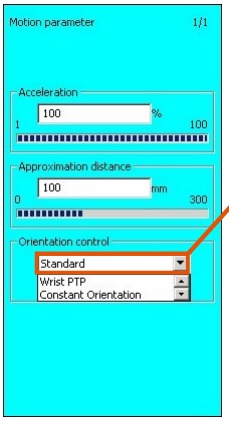




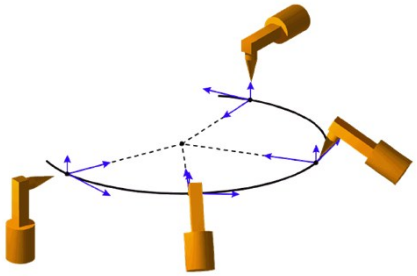
Acceleration
Acceleration used for the motion.
Range of values: 1...100%

Approximation distance*)
Size of approximate positioning range for the motion.
Range of values: 0...300 mm

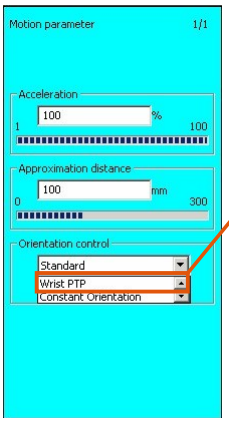
*) The parameter “Approximation distance” is only displayed if approximate positioning has been selected (CONT).



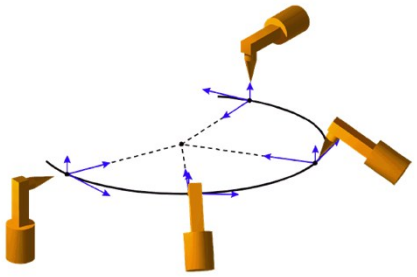
Orientation control - Standard



During the path motion, the orientation of the tool changes continuously from the start position to the end position. This is achieved by rotating and pivoting the tool direction.

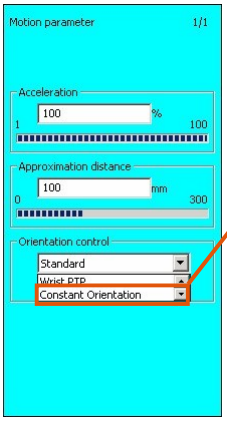


Orientation control - Wrist PTP

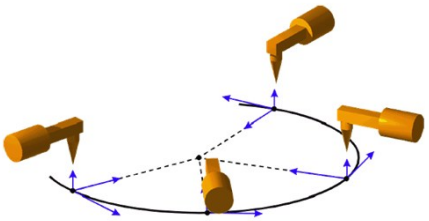


During the path motion, the orientation of the tool changes continuously from the start position to the end position. This is done by linear transformation (axis-specific motion) of the wrist axis angles. The problem of the wrist singularity can be avoided using this option as there is no orientation control by rotating and pivoting the tool direction.

Programming a CIRC motion



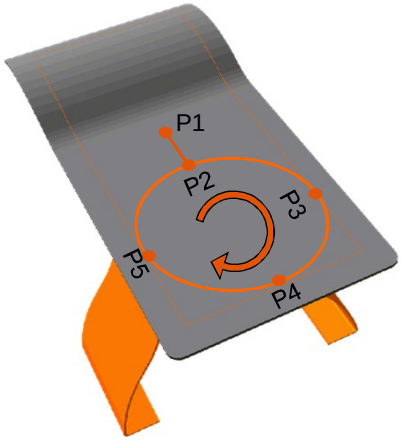
Orientation control - Constant



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

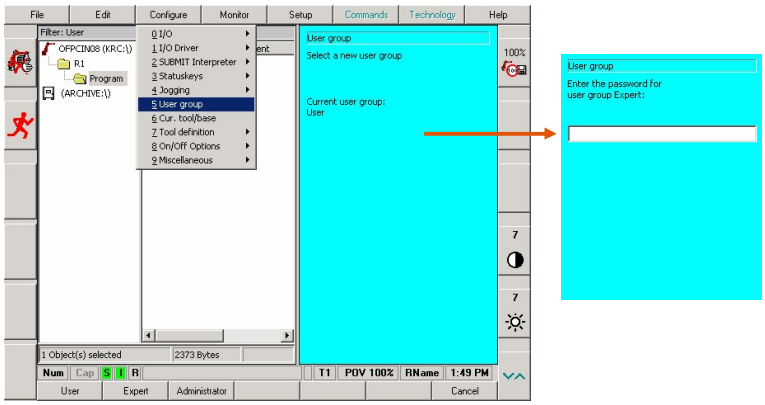
The 360° full circle

The full circle should be made up of at least two segments.




```
INI
PTP HOME
...
LIN P1
LIN P2
CIRC P3 P4 ;→ P3 is AUX; P4 is END
CIRC P5 P2 ;→ P5 is AUX; P2 is END
LIN P1
...
PTP HOME
END
```

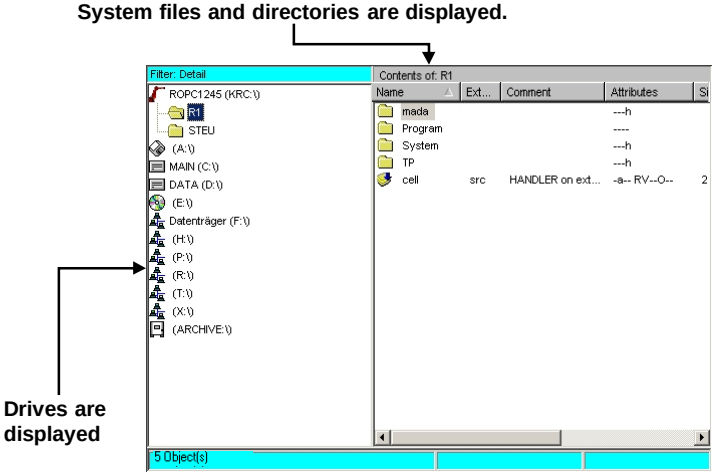
Changing user group



When the softkey “Expert” is pressed, the user is prompted to enter a password. The preset password is “kuka”.

Navigator (expert)

System files and directories are displayed.



Drives are displayed

Additional symbols in the Navigator (expert)

Drives

Symbol	Type	Default path
	Hard drive	e.g. Kukadisk (C:\) or Kukadata (D:\)
	CD-ROM	E:\
	Mapped network drive	e.g. F:\, G:\, etc.

Directories and files

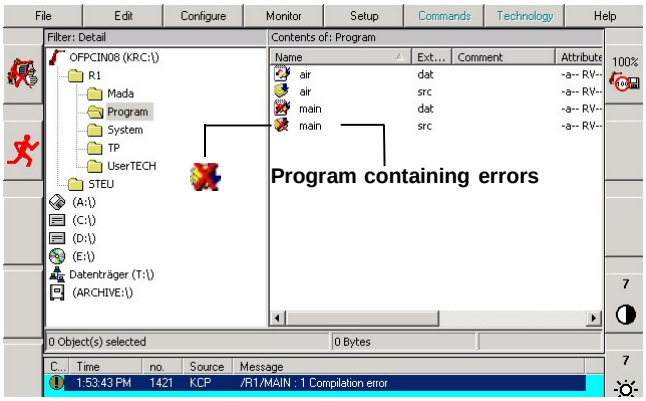
Symbol	Type	Meaning
	SRC file	Program file
	SRC file	Subprogram
	SRC file contains errors	Program with errors that cannot be interpreted by the compiler.
	DAT file	Data list
	DAT file contains errors	Data list with errors that cannot be interpreted by the compiler.
	ASCII file	Text file
	Other files	Binary files

Creating a new module (expert)

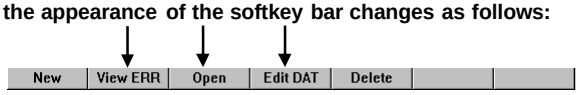
A KRL program can be made up of SRC and DAT files.
- SRC - contains program code
- DAT - contains specific program data

Cell	Skeleton program for control via a PLC
Expert	SRC and DAT file without a skeleton program
Expert Submit	SUB file without a skeleton program
Function	SRC file without a skeleton program
Module	SRC and DAT file with a skeleton program
Submit	SUB file with a skeleton program

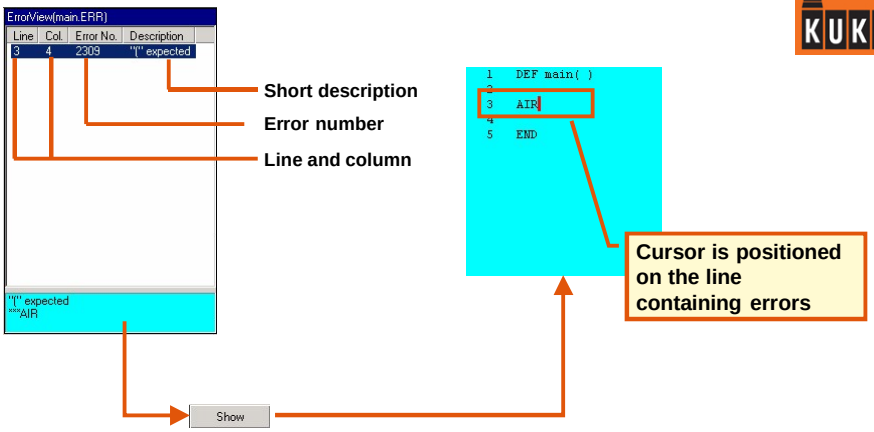
Error display



If the focus is moved to a file marked as containing errors, the appearance of the softkey bar changes as follows:

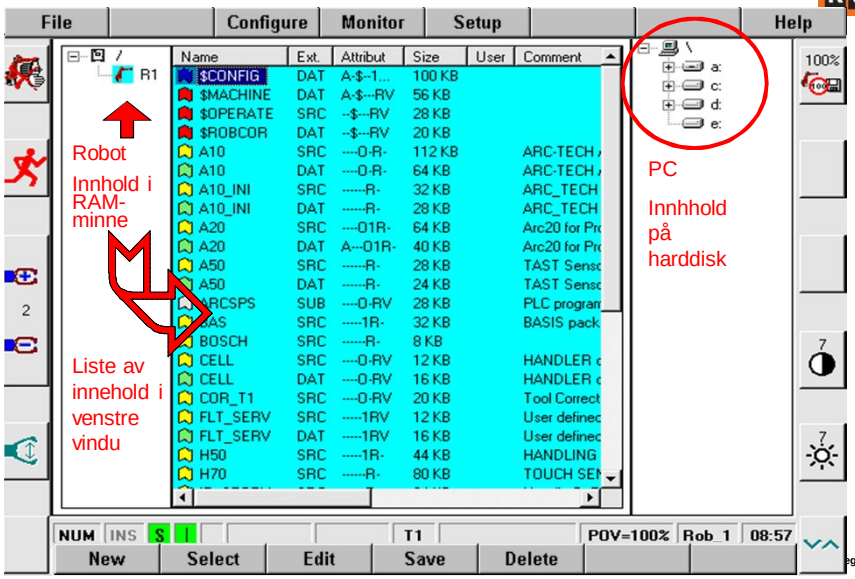


Error list



So that the line numbers in the error list correspond to those in the editor, the options “All FOLDs op” and “Detail view” must be activated.

Filhåndtering på Ekspertnivå

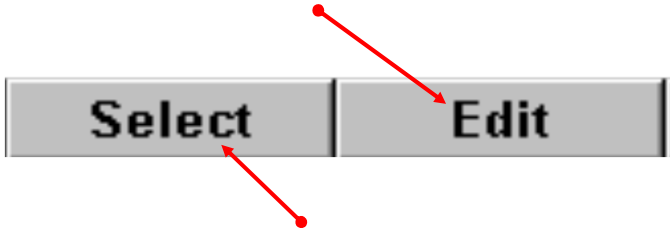


Filtyper i KRC1

.SRC	• Program eksikvering
	På brukernivå ser man bare filtypen SRC
.DAT	• Datalagring for program med samme navn (data, deklarasjoner og initsieringer)
.SUB	•Bakgrunnsprogram (Kontroller, PLS-fil)
.ERR	• Fil for å vise innkorrekt syntaks (Skapes automatisk ved stengning av fil med felaktig syntaks)

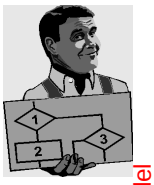
To muligheter å editere programmer

Tillegg og forandring av ekspertkommandoer uten bruk av Inlineformularer
Programeksikvering ikke mulig



Samme funksjoner som i anvendarnivå Tillegg og forandring med Inlineformularer Programeksikvering mulig

Programstruktur



Initsieringsdel

```
DEF PROG1()

;----- Declaration section -----
INT J

;----- Instruction section -----
$VEL_AXIS[1]=100   ;Definition of axis velocities
$VEL_AXIS[2]=100
$VEL_AXIS[3]=100
$VEL_AXIS[4]=100
$VEL_AXIS[5]=100
$VEL_AXIS[6]=100

$ACC_AXIS[1]=100   ;Definition of axis accelerations
$ACC_AXIS[2]=100
$ACC_AXIS[3]=100
$ACC_AXIS[4]=100
$ACC_AXIS[5]=100
$ACC_AXIS[6]=100

PTP {A1 0,A2 -90,A3 90,A4 0,A5 0,A6 0}

FOR J=1 TO 5
   PTP {A1 45}
   PTP {A2 -70,A3 50}
   PTP {A1 0,A2 -90,A3 90}
ENDFOR

PTP {A1 0,A2 -90,A3 90,A4 0,A5 0,A6 0}
END
```

Meldinger ved kompileringsfeil

1 SRC 62 ***J==J+1 &2309 (6)

Posisjon på raden

Feilnummer, se dokumentation

Tekst i raden som inneholder feilen

Radnummer (med FOLDs åppne men uten "begrenset visning", &ACCESS eller &COMMENT-raden)

Filtype for feilen: DAT, SRC eller SUB

Numering av feilmelding

Programkjøringsmodi

Mode	Beskrivelse
ISTEP 	Incrementel Step (individuell instruksjon) Programmet kjører med en rad om gangen, m.a.o. med STOPP etter hver (til og med tom) rad.
MSTEP 	Motion Step (bevegelsesinstruksjon) Programmet kjører en bevegelsesinstruksjon om gangen, m.a.o. med STOPP før hver bevegelsesinstruksjon.
GO 	Hele programmet kjøres i ett strekk uten STOPP mellom instruksjonene.

Eksempel på en betingelses innhold

Override & Driftsmodi:

```
IF (($OV_PRO==100) AND ($MODE_OP< >#T1) AND ($PRO_MODE1==#GO)) THEN
WELD_ON=TRUE
ENDIF
```

Tenningen av sveisen skurs kun på om følgende betingelser er oppfyllt:

Programmkjøringshastigheten er 100%

Operatørsmodusen får ikke være i T1 (redusert hastighet)

programmet får ikke kjøres stegvis

Folder (FOLDs)

Brukssområde:

- **Gruppere programseksjoner**
- **Navngi og gjemme fuksjoner**

Eksempel:

Gripper initialization **stengt fold**

åpen fold

```
;FOLD Gripper initialization
$OUT[7]=TRUE
$OUT[8]=FALSE
$OUT[9]=FALSE
$OUT[19]=TRUE
;ENDFOLD
```

Fargenøkkel

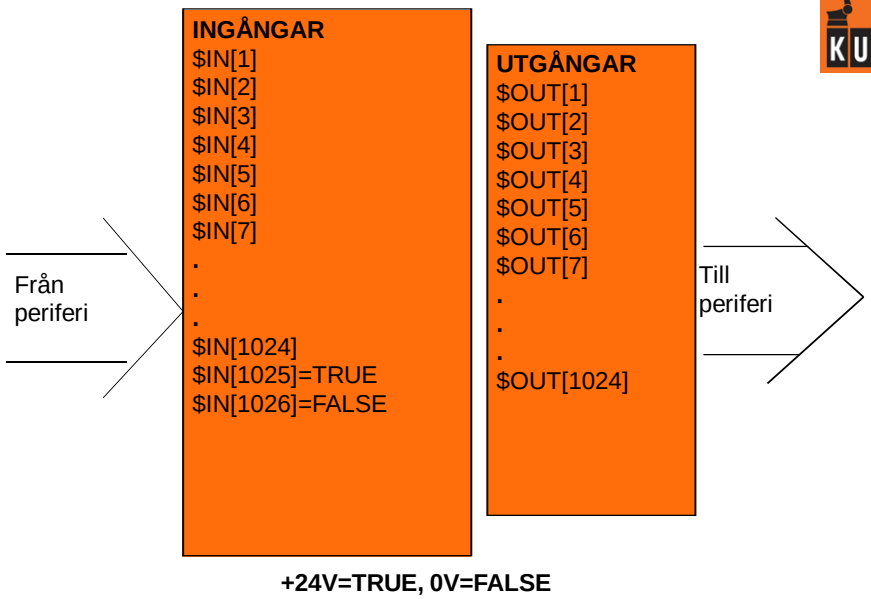
Editor farger:

- Stengt fold
- Åpen fold
- underfold
- Programkode
- Programkode i folder

Eksempel:

```
;FOLD Gripper initialization
;FOLD Preset outputs
$OUT[7]=TRUE
$OUT[8]=FALSE
;ENDFOLD
$OUT[9]=FALSE
$OUT[19]=TRUE
;FOLD Switch vacuum on
$OUT[3]=FALSE
;ENDFOLD
;ENDFOLD
```

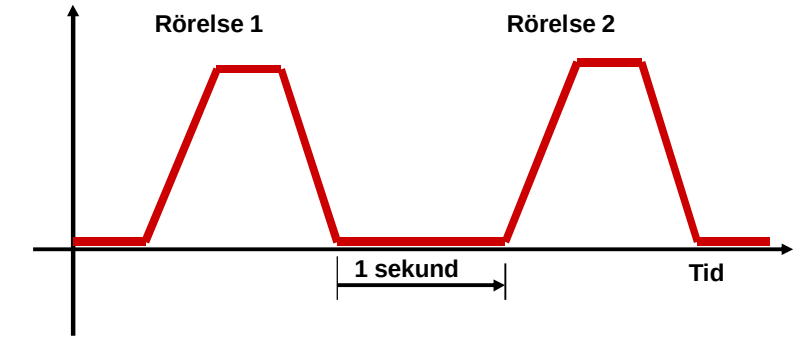

Styrskåp Ingångar / Utgångar



+24V=TRUE, 0V=FALSE

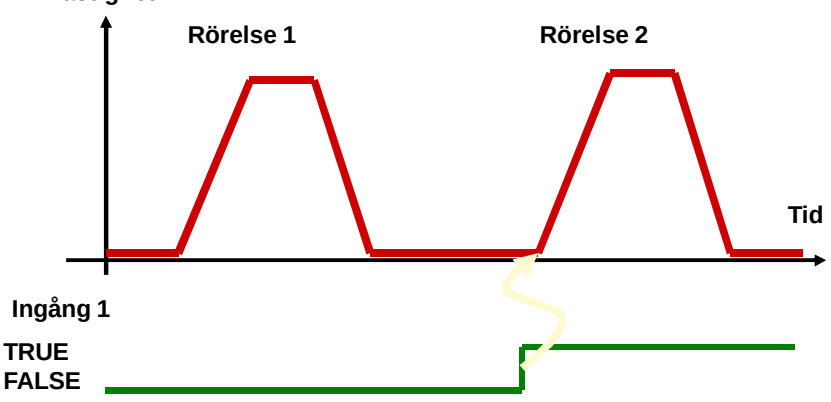
Wait instruktionen

```
LIN P1 PDAT1
WAIT SEC 1.0
LIN P2 PDAT2
```



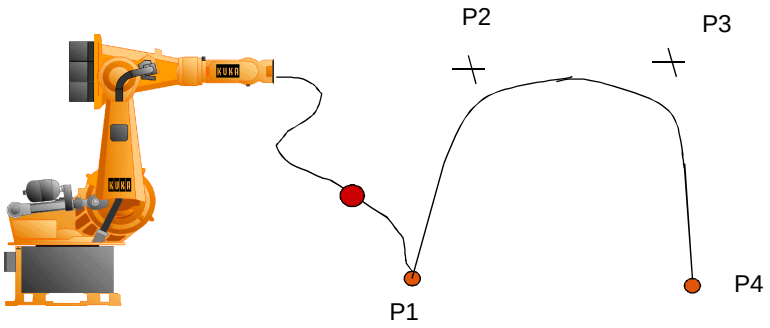
Vänta på en ingång (WAIT FOR IN)

```
LIN P1 PDAT1
WAIT FOR IN 1 :symNameState= TRUE
LIN P2 PDAT2
```



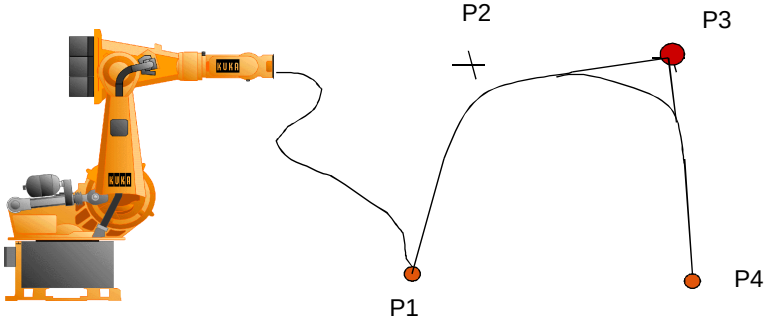
Sätta en utgång med aktiv advance (med CONT)

```
PTP P1    VEL=100% PDAT1
LIN P2 CONT VEL=1.0m/s CPDAT1
LIN P3 CONT VEL=1.0m/s CPDAT1
OUT 1 STATE=TRUE CONT
LIN P4  VEL=1.0m/s CPDAT1
```



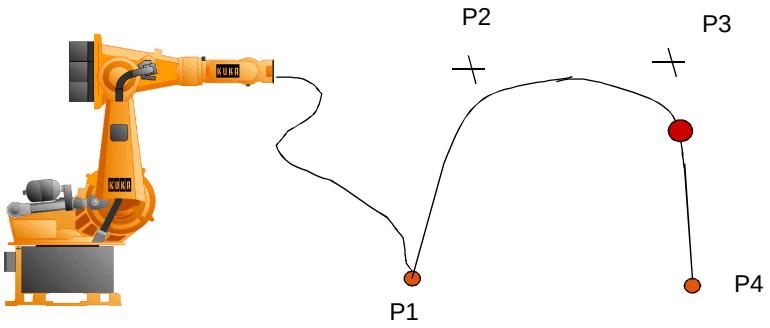
Sätta en utgång med stoppad advance (utan CONT)

```
PTP P1    VEL=100% PDAT1
LIN P2 CONT VEL=1.0m/s CPDAT1
LIN P3 CONT VEL=1.0m/s CPDAT1
OUT 1 STATE=TRUE
LIN P4  VEL=1.0m/s CPDAT1
```



Sätta en utgång i en approximerad pos. (SYNOUT)

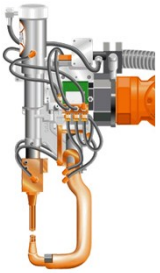
```
PTP P1    VEL=100% PDAT1
LIN P2 CONT VEL=1.0m/s CPDAT1
LIN P3 CONT VEL=1.0m/s CPDAT1
SYNOUT 1 STATE=TRUE at START DELAY= 0ms
LIN P4  VEL=1.0m/s CPDAT1
```



The polling of the inputs and setting of the outputs are used for communication between the robot controller and the periphery (e.g. tools, sensors, etc.)

Outputs:
$OUT[1] ... $OUT[4096]

Inputs:
$IN[1] ... $IN[4096]
$IN[1025]=TRUE
$IN[1026]=FALSE

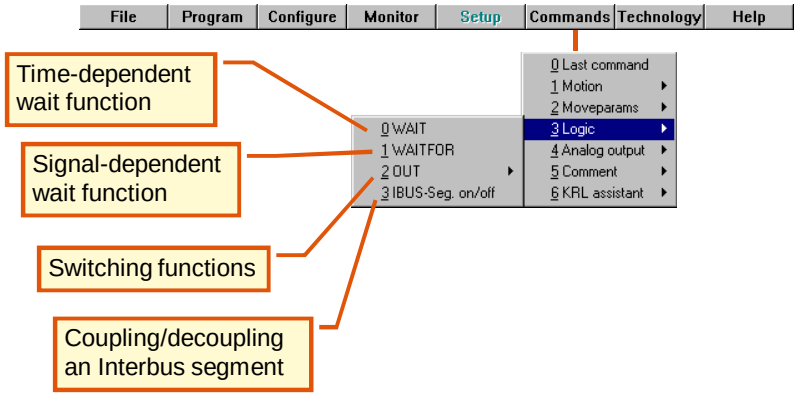


Robot controller

Periphery

Logic commands available

The following logic commands can be selected:



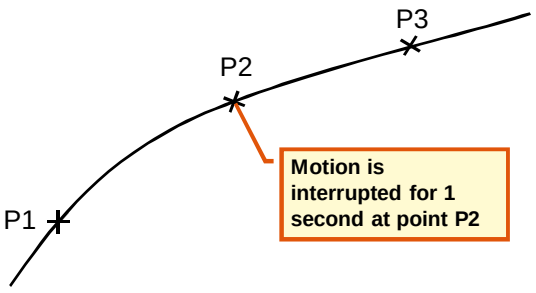
Time- and signal-dependent wait functions, and simple switching and pulse functions can trigger an advance run stop.

Time-dependent wait function (WAIT)

If “WAIT” has been selected, the wait time can be specified. This command always triggers an advance run stop, even with a wait time of 0 seconds.

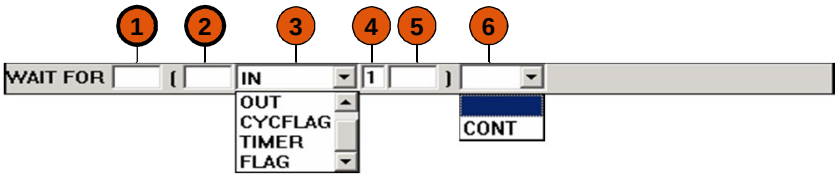
Example:

PTP P1 VEL=100% PDAT1
PTP P2 VEL=100% PDAT2
WAIT Time=1 sec
PTP P3 VEL=100% PDAT3



Signal-dependent wait function (WAIT FOR)

If “WAIT FOR” has been selected, the following parameters can be specified:



Box	Values	Remarks
1	“ ” **“NOT”**	• Insertion of an external logic operation (e.g. WAIT FOR **(IN1) AND** (IN2)) • Negation of the Boolean expression
2	“ ” **“NOT”**	• Insertion of an internal logic operation (e.g. WAIT FOR **(IN1) AND** (IN2)) • Negation of the Boolean expression

Signal-dependent wait function (WAIT FOR)

The wait condition can be programmed, for example, in the following general form:

WAIT FOR (IN1 **OR** IN2 **OR** IN3) **AND** (NOT OUT1 OR OUT2) **OR NOT** (IN4)

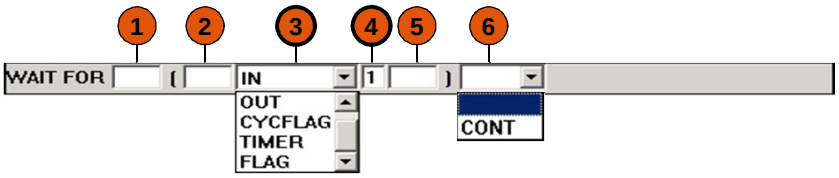
Internal logic operation: the operator is situated inside a bracketed expression.

External logic operation: the operator is situated between the bracketed expressions.

Mixed forms are possible: a maximum of 12 operands can be linked in a form.

Signal-dependent wait function (WAIT FOR)

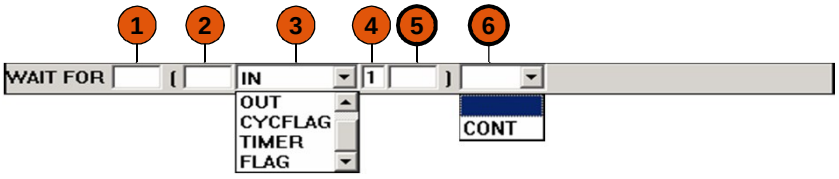
If “WAIT FOR” has been selected, the following parameters can be specified:



Box	Values	Remarks
3	**IN, OUT, TIMER, FLAG, CYCFLAG, user variable**	Inputs/outputs, various flags, timers or user-defined names are possible
4	**1- 4096**	Value of the input/output, flag or timer

Signal-dependent wait function (WAIT FOR)

If “WAIT FOR” has been selected, the following parameters can be specified:



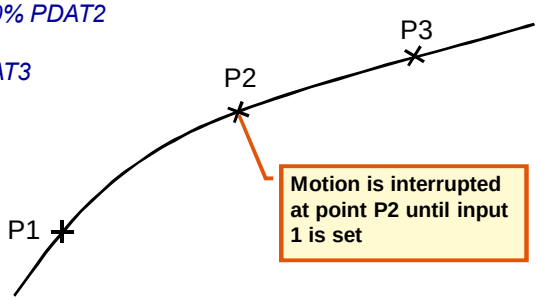
Box	Values	Remarks
5	“ ” **Existing long text name**	The long text name can be programmed in Expert mode with the system list activated
6	“ ” **“CONT”**	•Execution **with** advance run stop •Execution **in** the advance run

Signal-dependent wait function (WAIT FOR)

If “WAIT FOR” is selected with advance run stop activated, exact positioning is always carried out, even if the event condition is met.

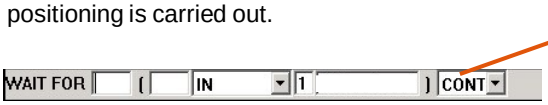
Example:

PTP P1 VEL=100% PDAT1
PTP P2 CONT VEL=100% PDAT2
WAIT FOR IN 1
PTP P3 VEL=100% PDAT3



Signal-dependent wait function (WAIT FOR)

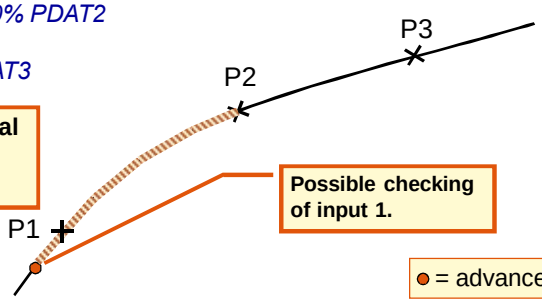
If “WAIT FOR” is selected with “CONT”, the event is checked in the advance run. If the event condition is met, then approximate positioning is carried out.



Example:

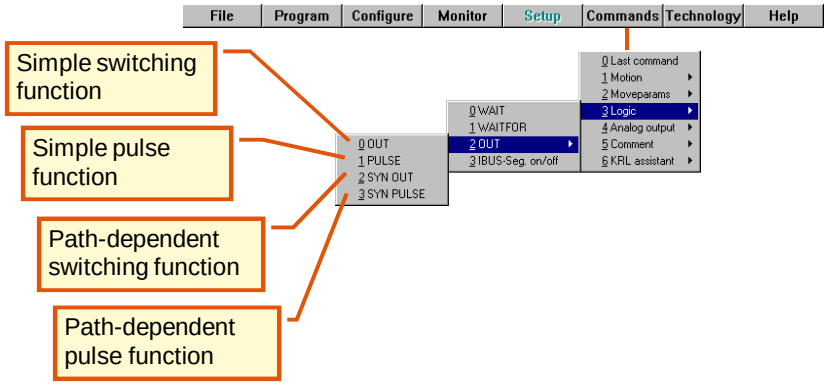
PTP P1 VEL=100% PDAT1
PTP P2 CONT VEL=100% PDAT2
WAIT FOR IN 1 CONT
PTP P3 VEL=100% PDAT3

Subsequent signal changes are <u>not</u> detected.



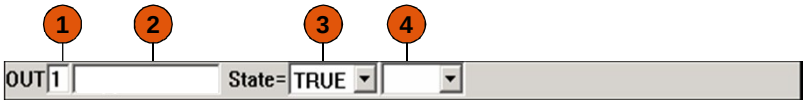
Switching functions

The following switching functions can be selected:



Simple switching function (OUT)

If “OUT” has been selected, the following parameters can be specified:



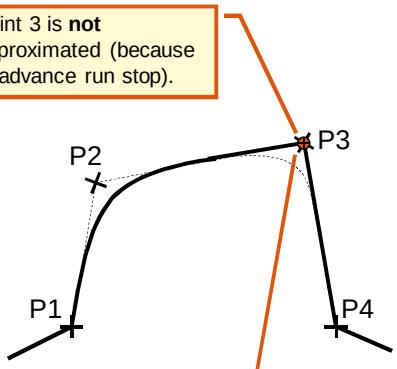
Box	Values	Remarks
1	**1- 4096**	Output number
2	“ ” **Existing long text name**	The long text name can be programmed in Expert mode with the system list activated
3	**TRUE FALSE**	State to which the output is switched
4	“ ” **“CONT”**	•Execution **with** advance run stop •Execution **in** the advance run

Simple switching function (OUT)

OUT 1 State= TRUE

Example:

LIN P1 VEL=0.2 m/s PDAT1
LIN P2 CONT VEL=0.2 m/s PDAT2
LIN P3 CONT VEL=0.2 m/s PDAT3
OUT 1 ' ' State= TRUE
LIN P4 VEL=0.2 m/s PDAT4

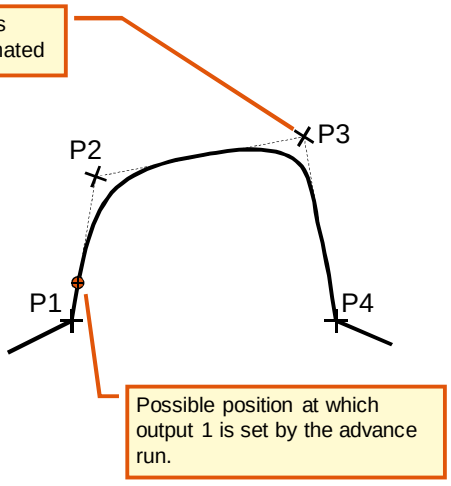


Simple switching function (OUT)

OUT 1 State= TRUE CONT

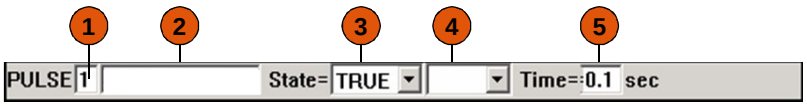
Example:

LIN P1 VEL=0.2 m/s PDAT1
LIN P2 CONT VEL=0.2 m/s PDAT2
LIN P3 CONT VEL=0.2 m/s PDAT3
OUT 1 ' ' State= TRUE CONT
LIN P4 VEL=0.2 m/s PDAT4



Simple pulse function (PULSE)

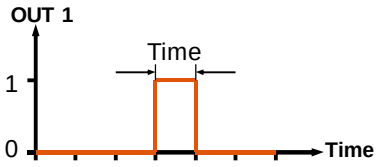
If “PULSE” has been selected, the following parameters can be specified:



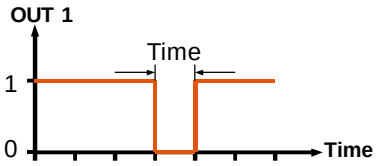
Box	Values	Remarks
1	**1- 4096**	Output number
2	**“ ” Existing long text name**	The long text name can be programmed in Expert mode with the system list activated
3	**TRUE FALSE**	State to which the output is switched
4	**“ ” “CONT”**	•Execution **with** advance run stop •Execution **in** the advance run
5	**0.1 … 3**	Length of the pulse in seconds

Simple pulse function (PULSE)

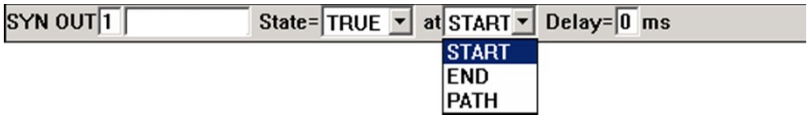
“HIGH” level switching: **STATE=TRUE**



“LOW” level switching: **STATE=FALSE**



Time-distance function "SYN OUT"



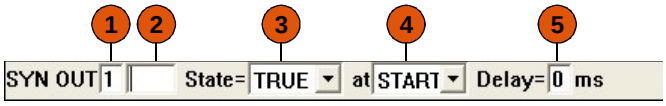
Path-dependent switching (with PTP, LIN, CIRC) can be triggered relative to a start or end point.

Such applications include, e.g.:

- Closing or opening the weld gun during spot welding
- Switching the welding current on/off during arc welding
- Starting or stopping the flow of adhesive in bonding or sealing applications.

Switching action at the start or end point of the path

If "SYN OUT" has been selected, the following parameters can be specified: here end point with **START** or **END**

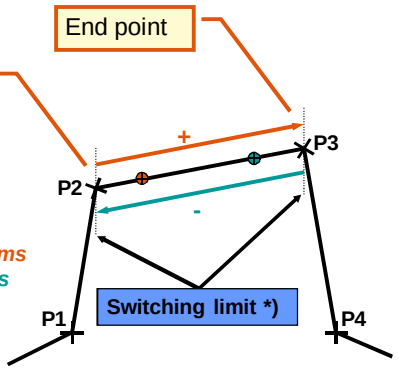


Box	Values	Remarks
1	**1- 4096**	Output number
2	**" "** **Existing long text name**	The long text name can be programmed in Expert mode with the system list activated
3	**TRUE** **FALSE**	State to which the output is switched
4	**"START"** **"END"**	End point at which the switching function is executed
5	**-1000 … +1000**	Delay of the switching action (in ms)

Switching action at the start or end point of the path

Start point and end point are exact positioning points:

SYN OUT 1 ' ' State= TRUE at START Delay= 0 ms



Example:

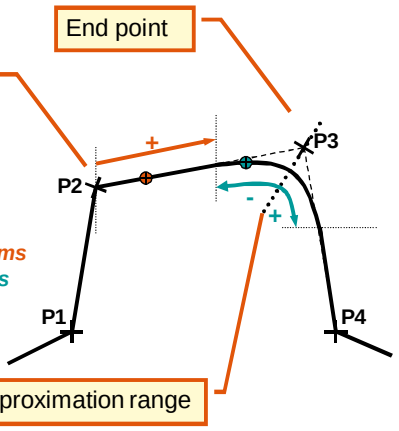
LIN P1 VEL=0.3 m/s CPDAT1
LIN P2 VEL=0.3 m/s CPDAT2
SYN OUT 1 ' ' State= TRUE at START Delay=20 ms
SYN OUT 2 ' ' State= TRUE at END Delay=-20 ms
LIN P3 VEL=0.3 m/s CPDAT3
LIN P4 VEL=0.3 m/s CPDAT4

*) Switching limit: If the specified values are outside the switching limits, the controller automatically switches at the switching limit.

Switching action at the start or end point of the path

Start point is exact positioning point, end point is approximated:

SYN OUT 1 ' ' State= TRUE at START Delay= 0 ms



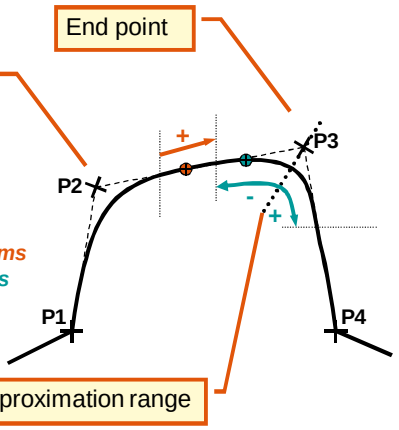
Example:

LIN P1 VEL=0.3 m/s CPDAT1
LIN P2 VEL=0.3 m/s CPDAT2
SYN OUT 1 ' ' State= TRUE at START Delay=20 ms
SYN OUT 2 ' ' State= TRUE at END Delay=-20 ms
LIN P3 CONT VEL=0.3 m/s CPDAT3
LIN P4 VEL=0.3 m/s CPDAT4

Switching action at the start or end point of the path

Start point and end point are approximated:

SYN OUT 1 ' ' State= TRUE at START Delay= 0 ms



Example:

```
LIN P1 VEL=0.3 m/s CPDAT1
LIN P2 CONT VEL=0.3 m/s CPDAT2
SYN OUT 1 ' ' State= TRUE at START Delay=20 ms
SYN OUT 2 ' ' State= TRUE at END Delay=-20 ms
LIN P3 CONT VEL=0.3 m/s CPDAT3
LIN P4 VEL=0.3 m/s CPDAT4
```

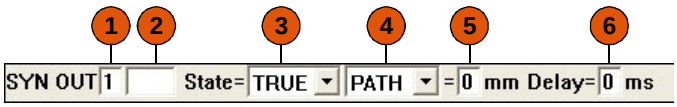

Switching action at any point on the path

- If you are using the path-related SYNOUT-PATH statement, you can trigger the switching action at any position along the path by specifying a distance.
- As with switching actions at the start or end points, this again can additionally be delayed or brought forward.
- The path-related switching action is only allowed with continuous-path motions (LIN or CIRC).
- The SYNOUT-PATH statement refers here to the next programmed motion block.

If a SYNOUT-PATH statement with path specification is programmed for a PTP motion, this will be refused by the interpreter when the motion is executed.

Switching action at the start or end point of the path

If “SYN OUT” has been selected, the following parameters can be specified: here end point with **PATH**



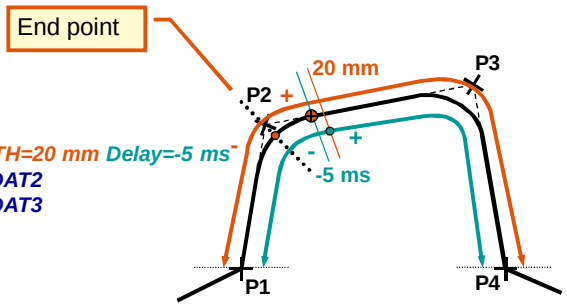
Box	Values	Remarks
1	**1- 4096**	Output number
2	**“ ” Existing long text name**	The long text name can be programmed in Expert mode with the system list activated
3	**TRUE FALSE**	State to which the output is switched
4	**“PATH”**	End point at which the switching function is executed
5	**-2000 … +2000**	Distance of the switching action from the end point (in mm)
6	**-1000 … +1000**	Delay of the switching action (in ms)

Switching action at any point on the path

Start point is exact positioning point, end point is approximated:

Example:

LIN P1 VEL=0.3 m/s CPDAT1
SYN OUT 1 ‘ ‘ State= TRUE PATH=20 mm Delay=-5 ms
LIN P2 CONT VEL=0.3 m/s CPDAT2
LIN P3 CONT VEL=0.3 m/s CPDAT3
LIN P4 VEL=0.3 m/s CPDAT4



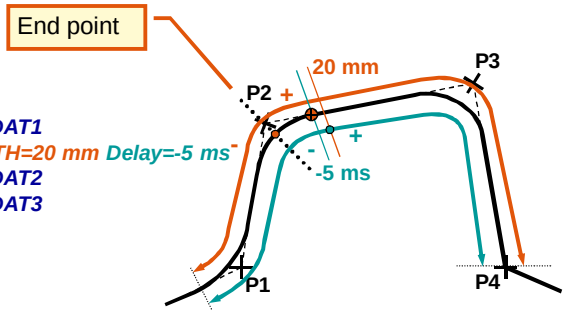
Switching action at any point on the path

Start point and end point are approximated:

SYN OUT 1 ' ' State= TRUE PATH = 20 mm Delay= -5 ms

Example:

```
LIN P1 CONT VEL=0.3 m/s CPDAT1
SYN OUT 1 ' ' State= TRUE PATH=20 mm Delay=-5 ms
LIN P2 CONT VEL=0.3 m/s CPDAT2
LIN P3 CONT VEL=0.3 m/s CPDAT3
LIN P4 VEL=0.3 m/s CPDAT4
```

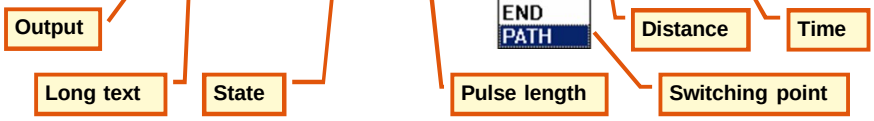


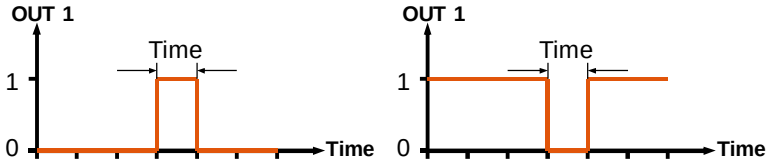
Switching action at any point on the path

If "SYN PULSE" has been selected, the following parameters can be specified:

SYN PULSE 1 End State= TRUE Time= 0.1 sec PATH = 0 mm Delay= 0 ms

START
END
PATH



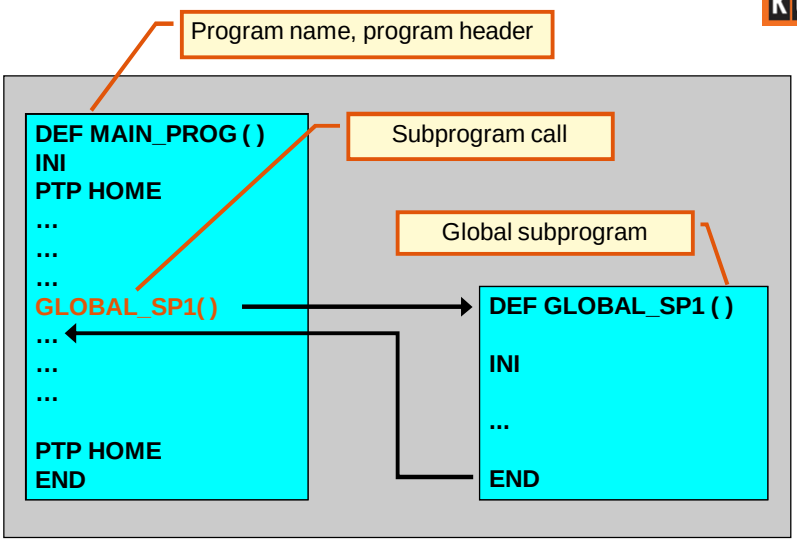


Subprograms

Subprograms are used for identical program sections that are repeated frequently.

- Subprograms **reduce the amount of typing** during programming.
- Subprograms **reduce the program length** thus making the program more transparent.
- Subprograms **can be reused** in other programs.
- Subprograms can be used for **structuring a program**.

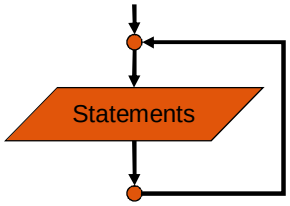
Global subprograms



Endless loop (LOOP)

Description

Cyclic executions can be programmed using LOOP. The statement block in the LOOP is continually repeated. If you want to end the repeated execution of the statement block, you must call the EXIT statement.

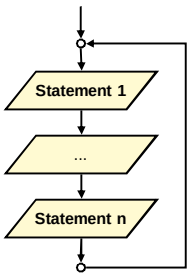


Endless loop

Syntax:

```
LOOP
        Statement 1
        ...
        Statement n
ENDLOOP
```

```
...
PTP HOME

LOOP
    LIN P1
    LIN P2
    LIN P4
ENDLOOP

PTP HOME
...
```



Conditional branch (IF..THEN..ELSE)

Description

Depending on a condition, either the first statement block (THEN block) or the second statement block (ELSE block) is executed.

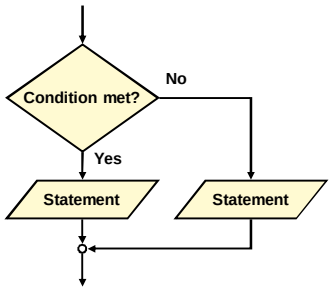
- There is no limit on the number of statements contained in the statement blocks.
- Several IF statements can be nested in each other.
- The keyword ELSE and the second statement block may be omitted.
- There must be an ENDIF for each IF.

Conditional branch

Syntax:

```
IF  Execution_Condition     THEN
        Statement
ELSE
        Statement
ENDIF
```

```
…
IF  $IN[22]==TRUE  THEN
      PTP HOME
ELSE
    $OUT[17]=TRUE
    $OUT[18]=FALSE
    PTP HOME
ENDIF
…
```



Unconditional exit from loops (EXIT)

Description

The EXIT statement appears in the statement block of a loop. It may be used in any loop.

The EXIT statement can be used to exit the current loop. The program is then continued after the ENDLOOP statement.

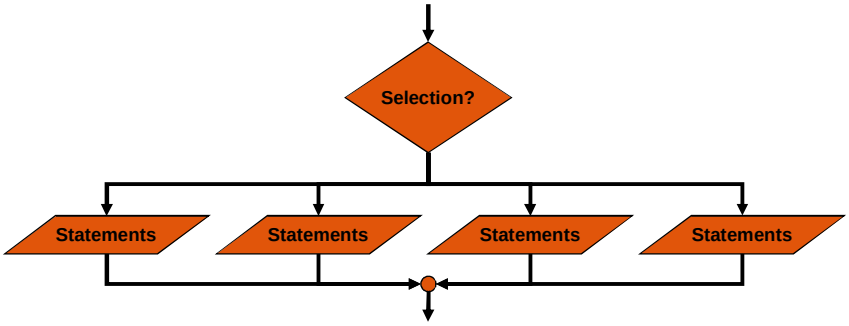
Unconditional exit from loops

Syntax: **EXIT**

```
DEF EXIT_PRO ( )
   PTP HOME
    LOOP              ;Start of endless loop
     LIN P1
       IF $IN[1] == TRUE THEN
       EXIT           ;Terminate when input 1 set
       ENDIF
     LIN P2
    ENDLOOP           ;End of endless loop
   PTP HOME
END
```

Description

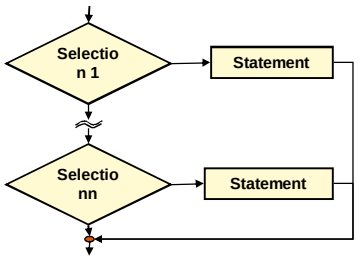
The SWITCH statement is a selection instruction for various program branches. Only one program branch is executed and the program then jumps immediately to the ENDSWITCH statement.



Syntax:

```
SWITCH Variable
    CASE 1
        Statement
    CASE 2
        Statement
    DEFAULT
ENDSWITCH
```

```
SWITCH PROG_NR
   CASE 1
      Part1 ( )          ; if Prog_No = 1
   CASE 2
      Part2 ( )          ; if Prog_No = 2
   DEFAULT
      ERROR_SP( ) ; all other values
ENDSWITCH
```



Data manipulation

Arithmetic operators

Basic arithmetic operations for the data types INTEGER and REAL:

Operator	Description
+	Addition or positive sign
-	Subtraction or negative sign
*	Multiplication
/	Division

Result of an arithmetic operation:

Operands	INT	REAL
INT	INT	REAL
REAL	REAL	REAL

Program example

```
DEF ARITH()
;------- Declaration section --------
INT A,B,C
REAL K,L,M
;-------- Initialization section -------  ;All variables are invalid prior to initialization!
A = 2                                      ;A=2
B = 9.8                                    ;B=10
C = 7/4                                    ;C=1
K = 3.5                                    ;K=3.5
L = 0.1 E01                                ;L=1.0
M = 3                                      ;M=3.0
;-------------Main section------------
A = A * C                                  ;A=2
B = B - 'HB'                               ;B=-1
C = C + K                                  ;C=5
K = K * 10                                 ;K=35.0
L = 10 / 4                                 ;L=2.0
L = 10 / 4.0                               ;L=2.5
L = 10 / 4.                                ;L=2.5
L = 10./ 4                                 ;L=2.5
C = 10./ 4.                                ;C=3
M = (10/3) * M                             ;M=9.0
END
```


Relational operators

Using relational operators, it is possible to form logic expressions. The result of a comparison is always of data type BOOL.

Operator	Description	Permissible data types
==	equal to	INT, REAL, CHAR, ENUM, BOOL
<>	not equal to	INT, REAL, CHAR, ENUM, BOOL
>	greater than	INT, REAL, CHAR, ENUM
<	less than	INT, REAL, CHAR, ENUM
>=	greater than/equal to	INT, REAL, CHAR, ENUM
<=	less than/equal to	INT, REAL, CHAR, ENUM

Example:
```
BOOL A,B
...
B = 10 < 3                                  ;B=FALSE
A = 10/3 == 3                               ;A=TRUE
B = ((B == A) <> (10.00001 >= 10)) == TRUE  ;B=TRUE
```

Logic operators

Logic operators are used for logic operations with Boolean variables, constants and simple logic expressions.

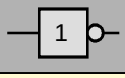
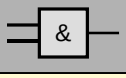
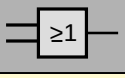
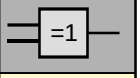
Operator	Operand number	Description
NOT	1	Inversion
AND	2	Logic AND
OR	2	Logic OR
EXOR	2	Exclusive OR

Example:

```
...
BOOL A,B,C
...
A = TRUE                                    ;A=TRUE
B = NOT A                                   ;B=FALSE
C = (A AND B) OR NOT (B EXOR NOT A);C=TRUE
A = NOT NOT C                               ;A=TRUE
...
```


Truth table

Truth table for logic operations:

Operation		NOT A 	A AND B 	A OR B 	A EXOR B 
A=TRUE	**B=TRUE**	FALSE	TRUE	TRUE	FALSE
A=TRUE	**B=FALSE**	FALSE	FALSE	TRUE	TRUE
A=FALSE	**B=TRUE**	TRUE	FALSE	TRUE	TRUE
A=FALSE	**B=FALSE**	TRUE	FALSE	FALSE	FALSE

Bit operators

Bit operators are used to perform logic operations on the individual bits of whole numbers.

Operator	Operand number	Description
B_NOT	1	Bit-by-bit inversion
B_AND	2	Bit-by-bit AND operation
B_OR	2	Bit-by-bit OR operation
B_EXOR	2	Bit-by-bit exclusive OR operation

As ASCII characters can also be addressed via the integer ASCII code, the data type of the operands may also be CHAR besides INT. The result is always of type INT.

Bit operators

Values	2^3	2^2	2^1	2^0	
	8	4	2	1	
Example: 1st number	0	1	0	1	=5
2nd number	1	1	0	0	=12
B_AND	0	1	0	0	=4
B_OR	1	1	0	1	=13
B_EXOR	1	0	0	1	=9

Bit-by-bit inversion does not simply involve all the bits being inverted. Instead, 1 is added to the operand and the sign is changed, e.g.:
B_NOT 10 = -11 or **B_NOT -10 = 9**

Priority of operators

Operators will be executed in order of priority.

Priority	Operator
1	NOT B_NOT
2	* /
3	+ -
4	AND B_AND
5	EXOR B_EXOR
6	OR B_OR
7	== <> < > >= <=

Example:

```
INT A,B
BOOL E,F
A = 4
B = 7
E = TRUE
F = FALSE
E = NOT E OR F AND NOT (-3 + A * 2 > B)      ;E=FALSE
```


Standard functions

Standard functions for calculating mathematical problems:

Description	Function
Absolute value	ABS (X)
Square root	SQRT (X)
Sine	SIN (X)
Cosine	COS (X)
Tangent	TAN (X)
Arc cosine	ACOS (X)
Arc tangent	ATAN2 (Y, X)