# Contents

## 0.1 Coordinate systems

### 0.1.1 Axis specific

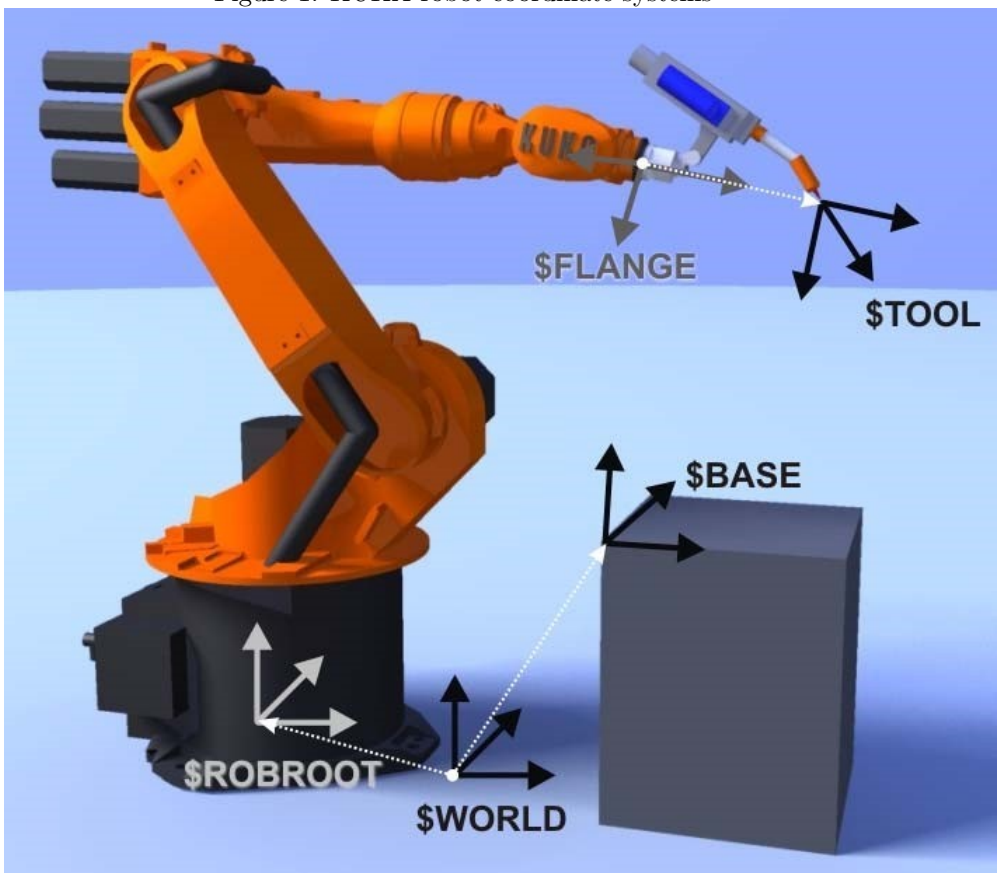
### 0.1.2 Cartesian

#### 0.1.2.1 Coordinate systems in conjunction with robots

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

- WORLD Coordinate System
  Fixed, rectangular coordinate system whose origin is located at the base of the robot. It is the root coordinate system for the ROBROOT and BASE coordinate systems. By default, the WORLD coordinate system is located at the robot base.
- ROBROOT Coordinate System
  Fixed, rectangular coordinate system whose origin is located at the base of the robot. It is the root coordinate system for the ROBROOT and BASE coordinate systems. By default, the WORLD coordinate system is located at the robot base.
- BASE Coordinate System Fixed, rectangular coordinate system whose origin is located at the base of the robot. It is the root coordinate system for the ROBROOT and BASE coordinate systems. By default, the WORLD coordinate system is located at the robot base.
- TOOL Coordinate System
  a Cartesian coordinate system which is located at the tool center by default, the origin of the TOOL coordinate system is located at the flange center point. The TOOL coordinate system is offset to the tool center point by the user

Figure 1: KUKA robot coordinate systems




## 0.2 KUKA Robot Language (KRL) Quick Guide

KRC 4 controller uses KRL KUKA programming language.

### 0.2.1 Variables and Declarations

All system variables start with $sign, mind not starting any "user-defined" name with this sign to avoid syntax errors.

#### 0.2.1.1 Names in KRL

- Can have a maximum length of 24 characters
- Can consist of letters(A - Z), numbers(0 - 9) and the special characters '$'.
- Must not begin with a number.
- Must not be a keyword.

#### 0.2.1.2 Declaration and initialization of variables

- Variables (simple and complex) must be declared in the SRC file before the INI line and initialized after the INI line
- Variables can optionally also be declared and initialized in a local or global data list.
- Every variable is linked to specific data type.
- The data type must be declared before use.
- The keyword for the declaration is DECL. It can be omitted in case of the four simple data type
- In order to place syntax before the INI line, the DEF line must be activated:

**Open file >Edit >View >DEF line**

```
DEF program name ( )
DECL data type user defined variable
; declaration section of variables
INI
; Initialization section of user defined variables.

; Instruction Section
...
END
```

#### 0.2.1.3 Simple Data types

| Data Type | Integer | Real | Boolean | Character |
|---|---|---|---|---|
| Keyword | INT | REAL | BOOL | CHAR |
| Meaning | Floating point number | Logic state | Boolean | Character |
| Range | $-2^3 12^3 1 - 1$ | $1.1E-383.4E+38$ | TRUE, FALSE | ASCII character |
| Example | 2 | 4.23 | TRUE | C |

Table 1: is an example of KRL Data Types

**Structure Types**

- AXIS: A1 to A6 are angle values (rotational axes) or translation values (translational axes)

| Axis: A1 .., A2 .., A3 .., A4, A5 .., A6 .. |
|---|

- FRAME: X, Y, and Z are space coordinates, while A, B, and C are the orientation of the coordinate system.

| FRAME: X .., Y .., Z .., A .., B .., C .. |
|---|

- POS and E6POS: S (Status) and T (Turn) define axis positions unambiguously

| POS: X .., Y .., Z .., A .., B .., C .., S ..., T |
|---|

## 0.3 Motion Programming

### 0.3.1 Motion Types

The robot can move in various motion types. Paths are created according to the operation of each axis. Thus, the robot can be controlled to create either linear or circular path.

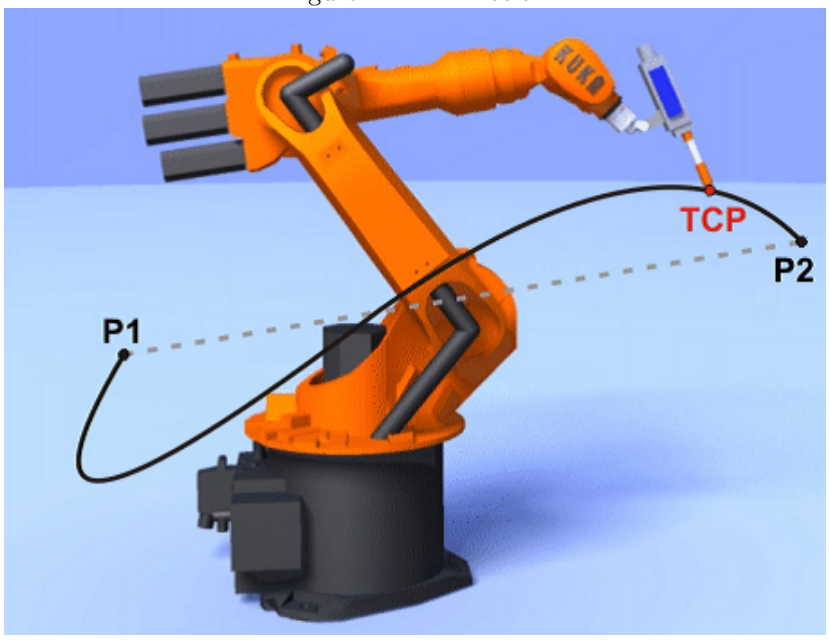
#### 0.3.1.1 Axis-specific motion

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. The first motion in the program must be PTP as status and turns are only evaluated here. The coordinates of the end point are absolute.

**characteristics**

- smooth motion
- Robot can move from start to end singularity free. As long as both the starting and ending points are in the working envelope, the robot will get to the end point without collision or sudden movement.
- Control is much simpler than continuous path control.

Figure 2: PTP Motion




#### 0.3.1.2 CP motion

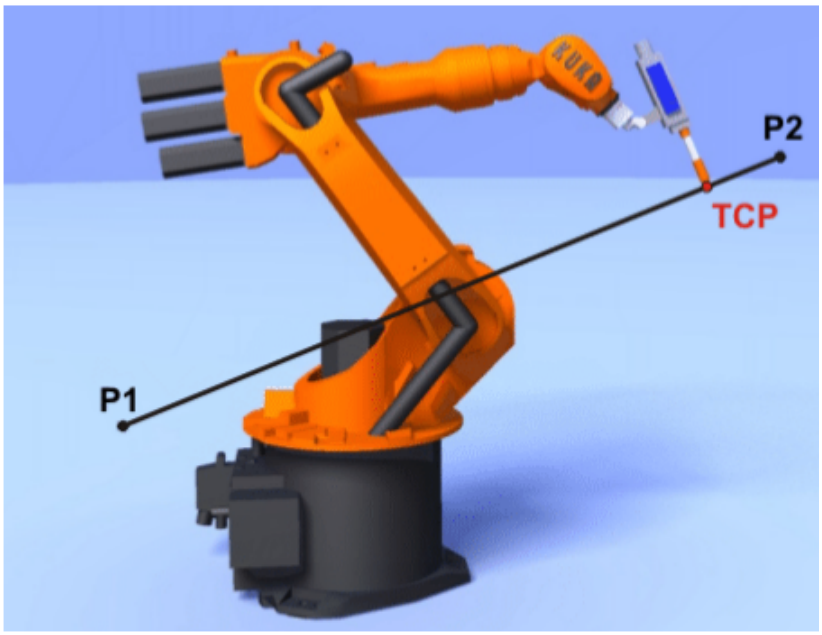
**LIN Motion** Motion at a defined velocity and acceleration along a straight line. This motion requires the programmer to teach one point. The robot uses the point defined in the previous move as the start point and the point defined in the current command as the end point and interpolates a straight line in between the two points.

**CIRC Motion** Motion at a defined velocity and acceleration along a circular path or a portion of a circular path. This motion requires the programmer to teach two points, the mid-point and the end point. Using the start point of the robot (defined as the end point in the previous motion command) the robot interpolates a circular path through the mid-point and to the end point.

### 0.3.2 Approximate Positioning

Approximate positioning of motion means that the next programmed point will not be exactly reached. This can help to shorten cycle times

Figure 3: LIN Motion




#### 0.3.2.1 PTP-PTP approximate positioning

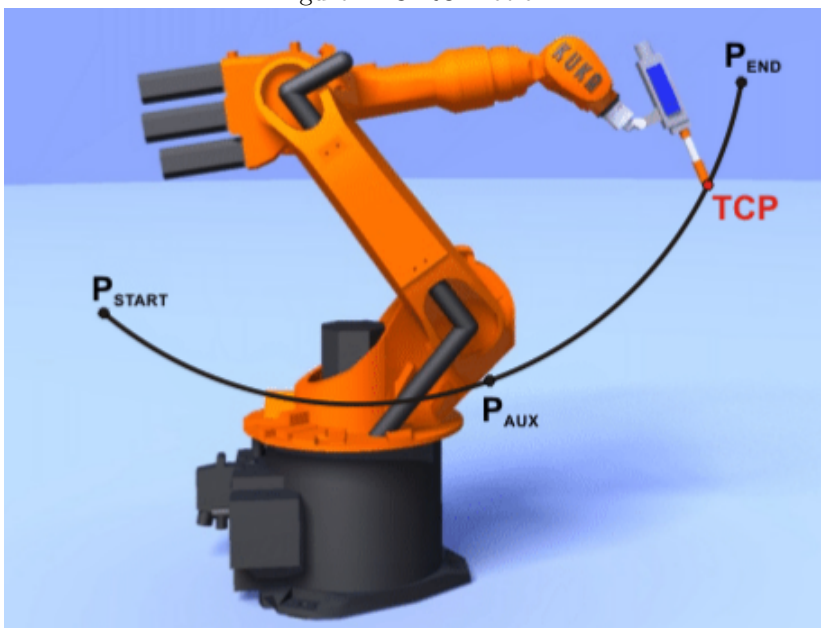
For the purposes of PTP approximate positioning,the controller calculates the distances the axes are to move in the approximate positioning range, and plans velocity profiles for each axis which ensure tangential transition from the individual instructions to the approximate positioning contour.

System Variable, $APO.CPTP enables the start of approximate positioning to be specified as a percentage of these maximum values. The approximate positioning of a point is displayed in the PTP command by adding the key word $C_PTP$:

| |
|---|
| $APO.CPTP = 80 |
| PTP HOME $C_PTP$ |

The greater this value the, the more path is rounded.

Figure 4: CIRC Motion




**Status and Turns** The position of x,y,z and orientation A,B,C values of TCP are not sufficient to define the robot position ,as different axis positioning are possible for the same TCP . Status and turns serve to define the position that can be achieved with different axis positions.

### 0.3.3 User Programming

Inline forms are available in the KSS for frequently used instruction. They simplify programming and facilitates user interface with controller without the need of knowing detail information about KUKA programming Language

### 0.3.4 Expert Programming

In the Expert interface, can achieve advanced programming using the KRL programming language and perform complex application programs including subprograms, interrupt programming, loops, and program branches.

Figure 5: Speed Profile:
a) If all points approached exactly
b)In case of approximate positioning of the points

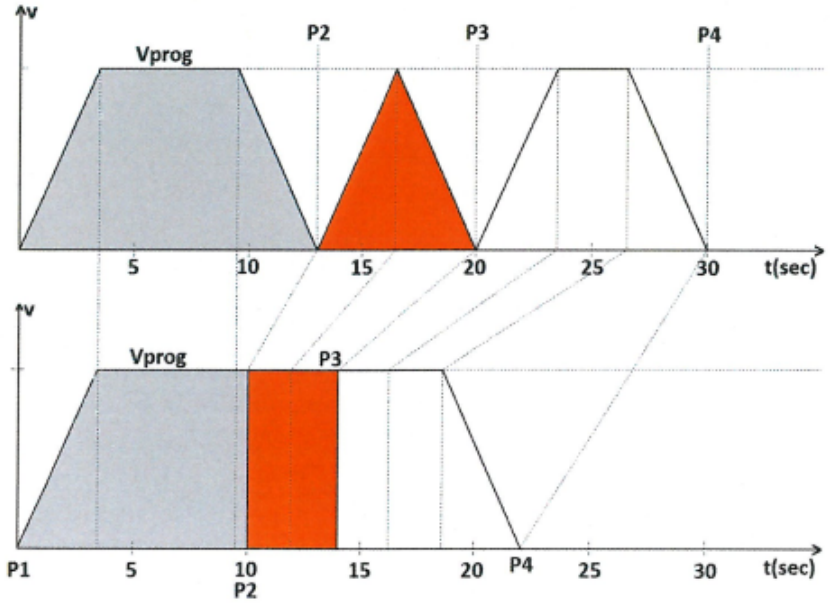



Figure 6: Approximate positioning of an auxiliary points

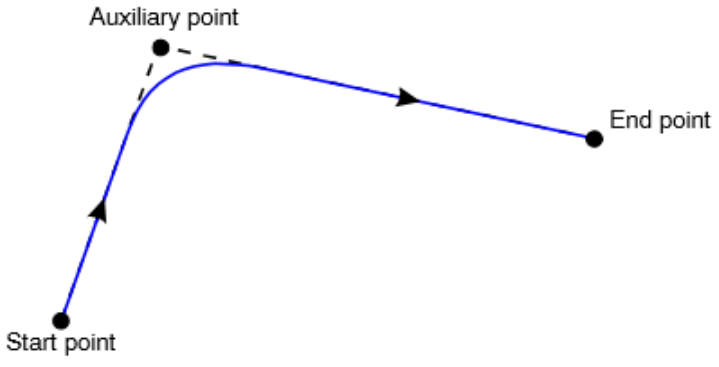


Figure 7: Same TCP, different axis position