Rapid and robot programming

Lab preparation and exercises

Mikael Norrlöf

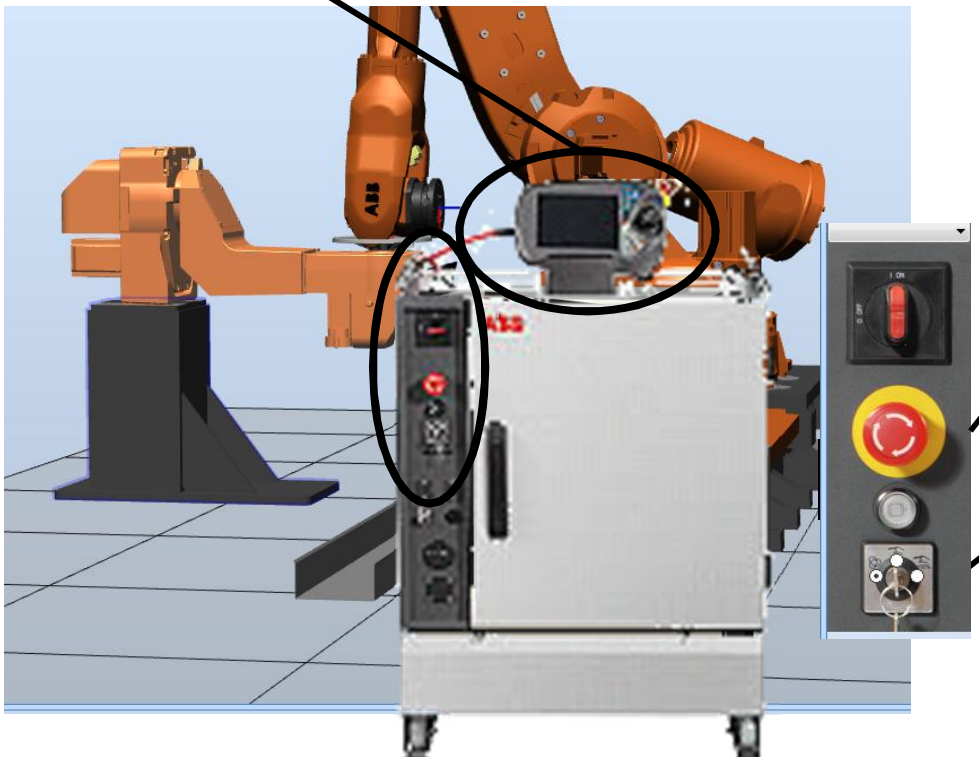
Robot safety

- The robot control system can be in three distinct modes
  - Manual, the speed is reduced to a maximum 250 mm/s
  - Manual 100%, full performance is reached but the user has to keep pushed an extra button on the flex pendent
  - Auto, the robot moves at full speed completely “autonomous”

- Under no circumstances it is allowed to be inside the robot cell when the robot runs in “Manual 100%” or “Auto”!
- The robot is potentially dangerous also at low speeds. Do not stay close when running a program (also in “Manual”)

Robot hardware

Teach pendent
(FlexPendant)



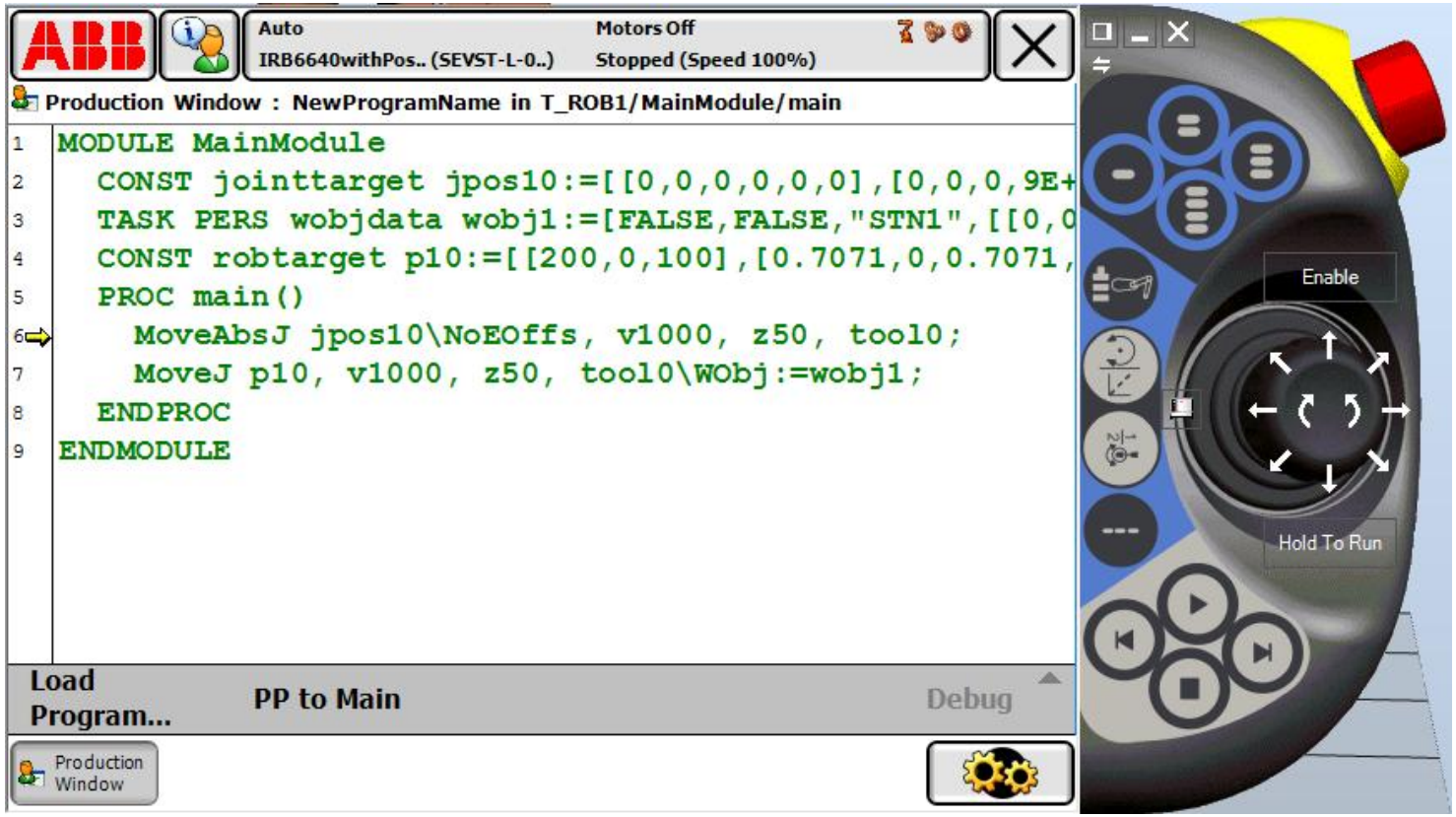
Motors on / acknowledge restart of motors after system stop

Mode selector key

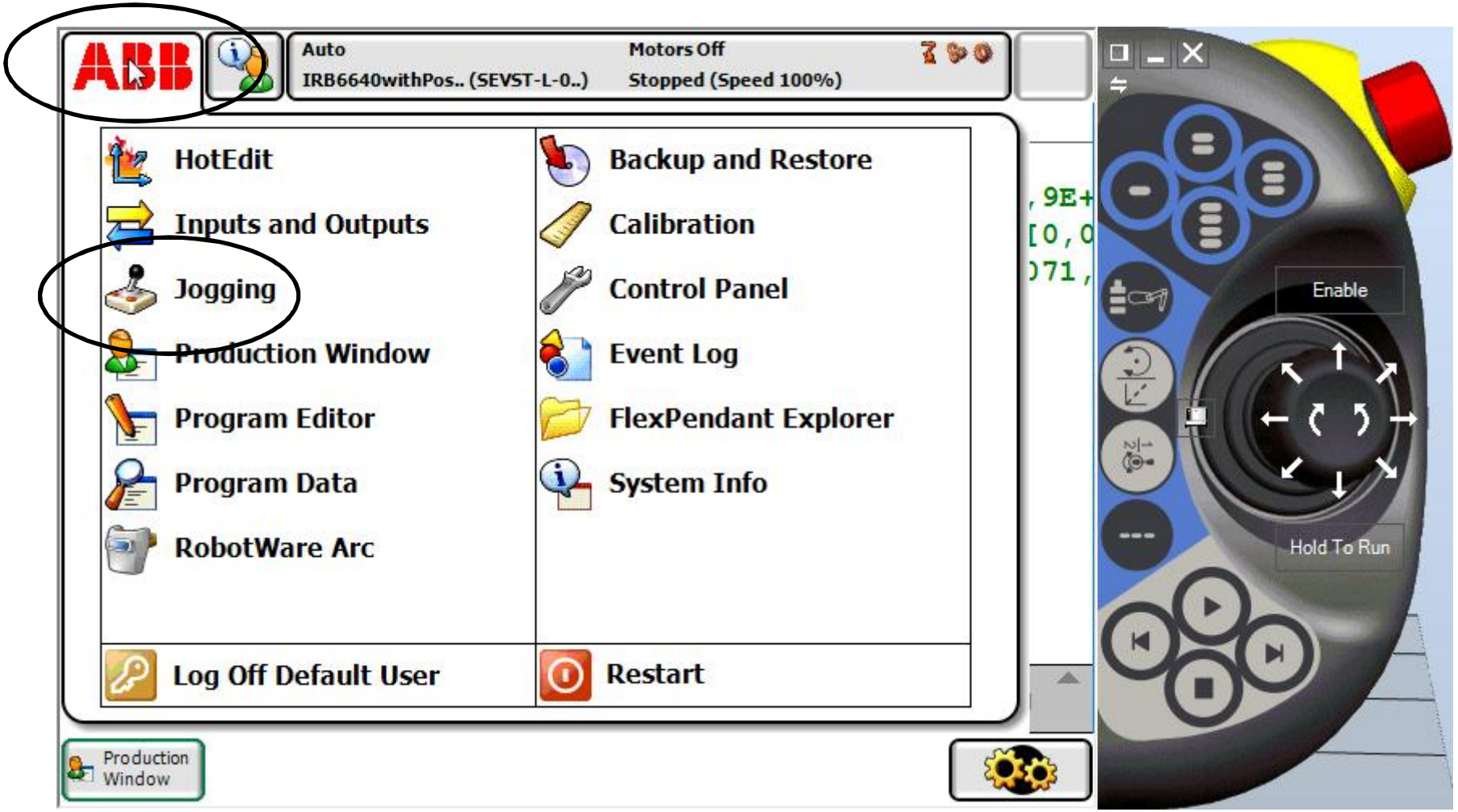
Emergency stop button (also available on the TeachPendant)

Robot hardware

Teach pendant



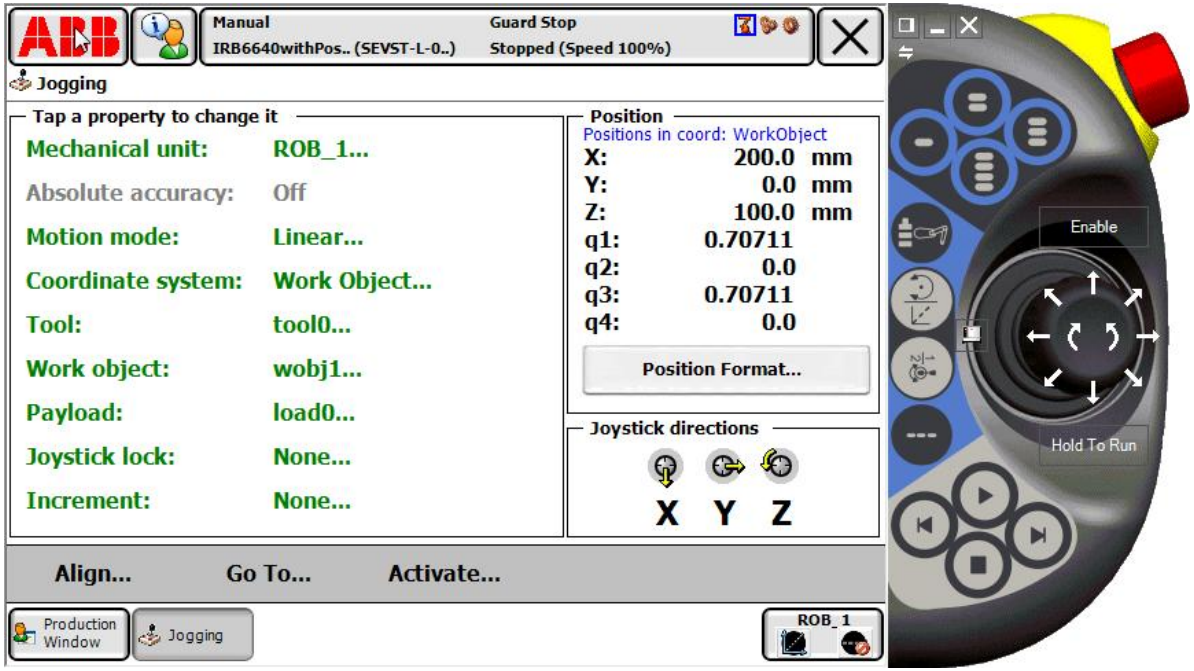
Robot hardware



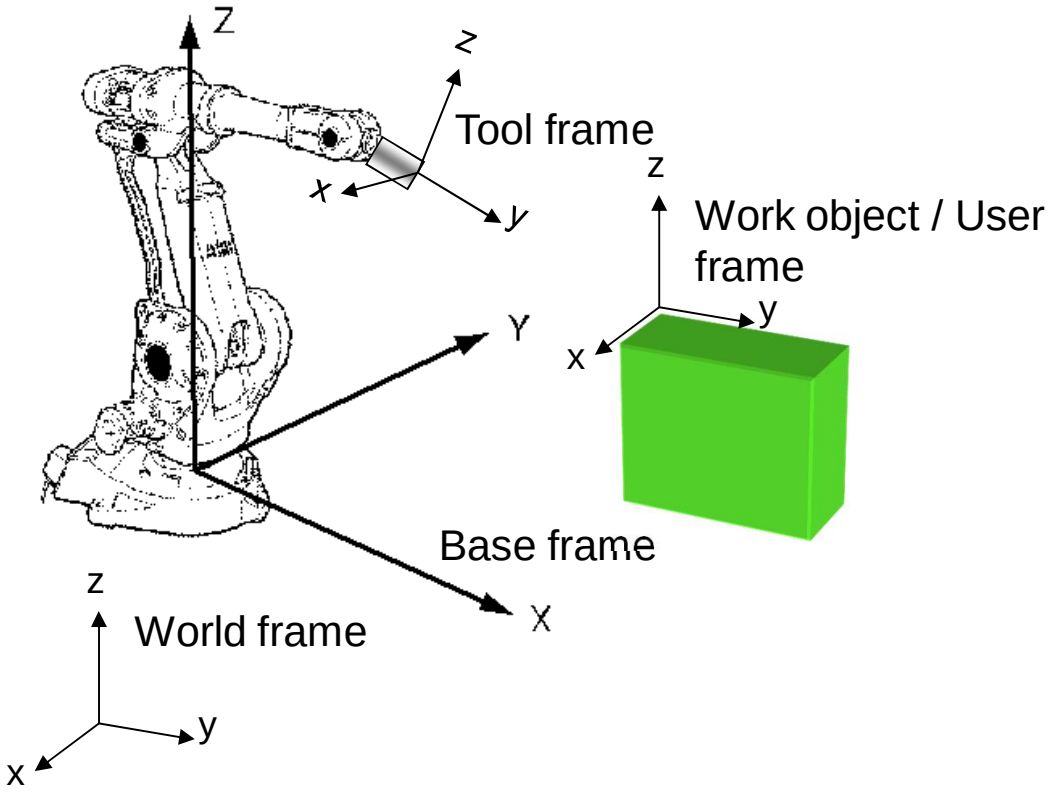
Moving the robot with the joystick (jogging)

Choose jog-mode

- *Motion mode* decides how to use the joystick
- *Coordinate system* affects the jogging directions
- *Tool* affects the position and also the jogging directions if Coordinate system is tool
- *Work object* affects the position and also the jogging directions if Coord is wobj



Coordinate systems

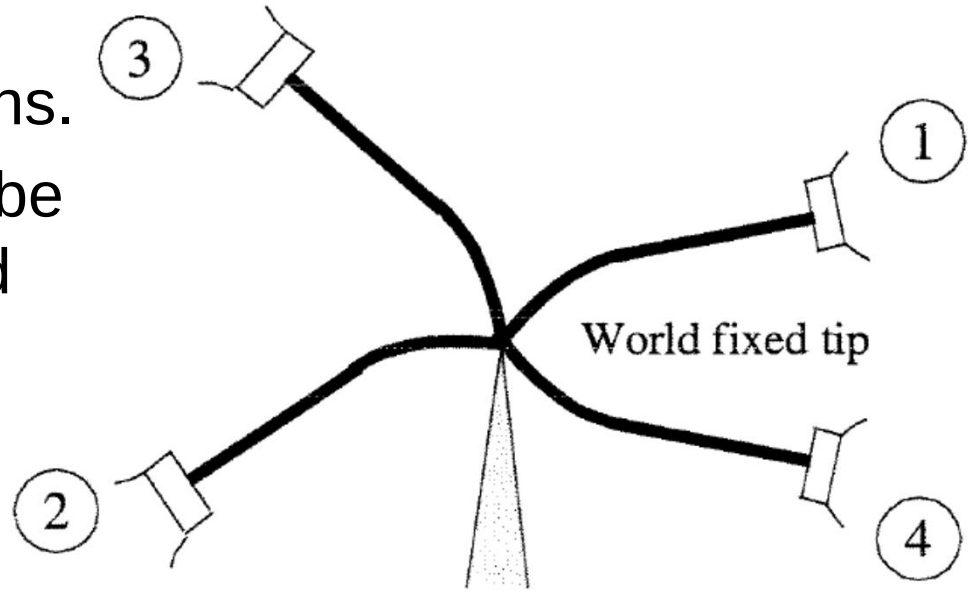


To help in the programming it is possible to define a number of coordinate systems. Some can be defined/calibrated using the robot.

Define a tool coordinate system

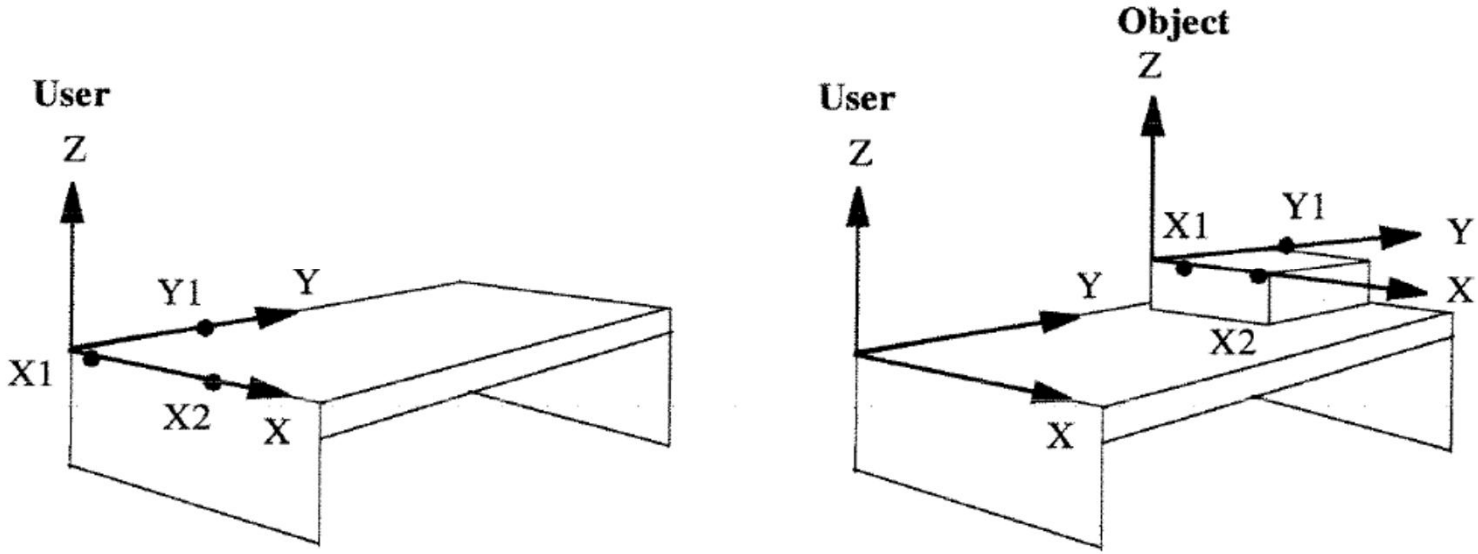
Approach a world fixed tip from (at least) four directions.

Mass and inertia data can be identified using pre-defined movements of the robot.



Define a work object

- The work object coordinate frame can be defined using a three point method.



A Rapid module (program)

Language originally developed in collaboration with SoftLab.

```
MODULE MainModule
   VAR num length;
   VAR num width;
   VAR num area;

   PROC main()
        length := 10;
        width := 5;
        area := length * width;
        TPWrite "The area of the rectangle is “ \Num:=area;
   END
PROCENDMODULE
```


Controlling the program flow

- IF THEN ELSE (ELSEIF)

```
VAR num time := 38.7;
IF time < 40 THEN
  TPWrite "Part produced at fast rate";
ELSEIF time < 60 THEN
  TPWrite "Part produced at average rate";
ELSE
  TPWrite "Part produced at slow rate";
ENDIF
```

- FOR

```
FOR i FROM 1 TO 5 DO
  TPWrite "Hello";
ENDFOR
```

- WHILE

```
VAR num sum := 0;
VAR num i := 0;
WHILE sum <= 100 DO
  i := i + 1;
  sum := sum + i;
ENDWHILE
```


Other useful commands

- Control external devices (digital output)

  To close the gripper,
  SetDO do1, 1;

  To open the gripper,
  SetDO do1, 0;

Syntax comments

- A statement in Rapid ends with a semicolon, exceptions are IF ENDIF, FOR ENDFOR, …
- A comment in Rapid starts with a !

```
! Calculate the area of the rectangle
area := length * width;
```

- The Rapid interpreter is not case sensitive but it is recommended that all reserved words (e.g. VAR, PROC) are written in capital letters.

Move instructions

```
MoveL p10, v1000, fine, tool0;
```

- p10 specifies the position that the robot shall move to.
- v1000 specifies that the speed of the robot shall be 1000 mm/s.
- fine specifies that the robot shall go exactly to the specified position and not cut any corners on its way to the next position.
- tool0 specifies that it is the mounting flange at the tip of the robot that should move to the specified position.

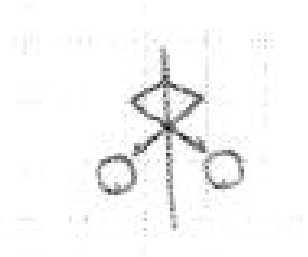
```
MoveC p10, p20, v1000, fine, tool0;
MoveJ p10, v1000, fine, tool0;
MoveAbsJ j10, v50, z50, tool0;
```


Offset function

- With Offs() it is possible to add an offset to a point in the x-, y-, and z-direction

```
MoveL Offs(p10,0,0,20), v1000, fine, tool0;
```

The robot will move to a point 20mm i z-direction relative to p10. Offs() is a very efficient tool to offline produce a desired motion given a reference-point.

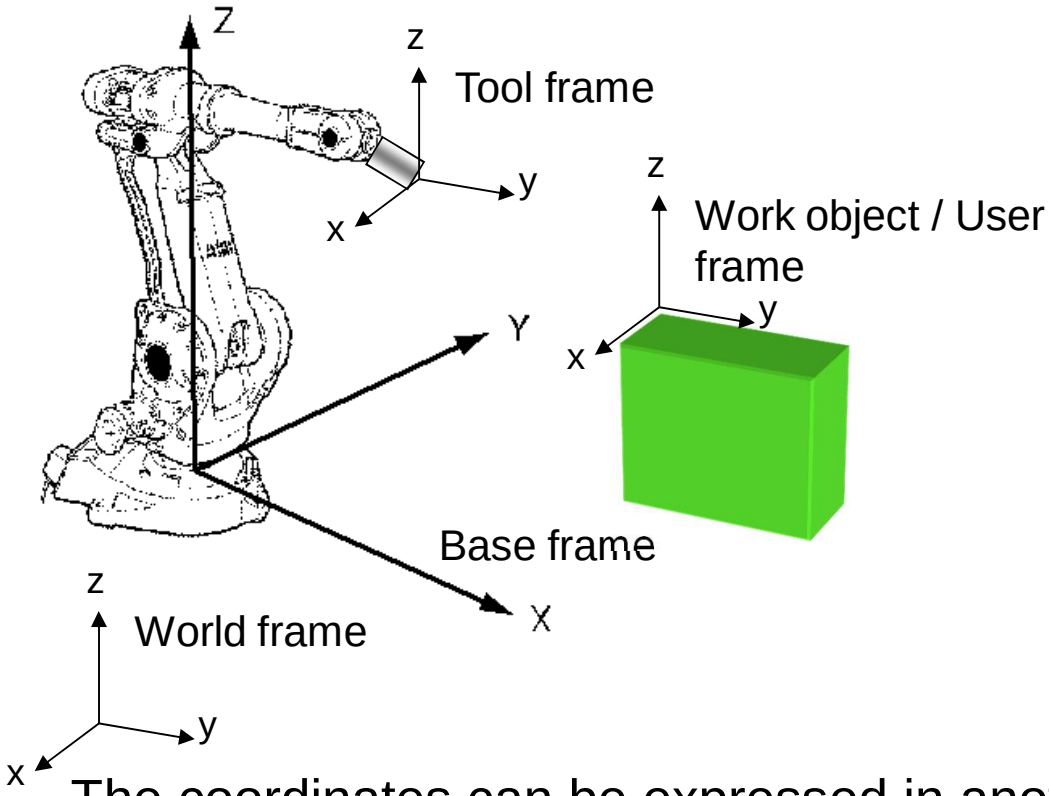


Configuration supervision

- To avoid problems with configuration warning/error it is possible to use

  ConfL \Off; ! For MoveL
  ConfJ \Off; ! For MoveJ

Coordinate systems



The coordinates can be expressed in another coordinate system (work object) by attaching an additional argument to the move instruction:

```
MoveL p10, v1000, fine, tool0 \WObj:=wobj1 ;
```

RobotStudio – some tricks and tips

- Zoom in and out,
  ctrl + right mouse button + move mouse left/right
- Pan
  ctrl + left mouse button + move mouse up/down/left/right
- Rotate
  shift + ctrl + left mouse button + move move mouse up/down/left/right

Lab

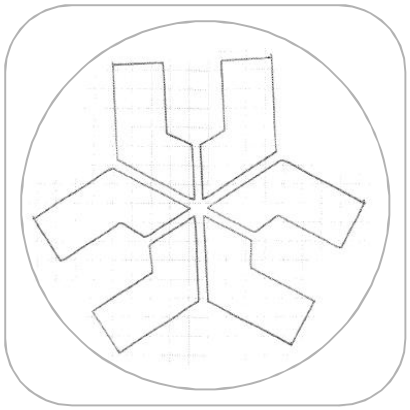
Exercise 1.

a) Jog the robot using the teach pendant while testing the different jog-modes.
b) Try jogging close to a singularity
c) Jog the other mechanical units while using different coordinate systems.

Lab

Exercise 2.

Program the robot to draw the figure below. The inner geometry is optional. Simulate that the robot uses a pen by moving to a specified fixed position before and after the figure is drawn.



It is important that the program uses the right tool and work object. Try to move the work object (using the positioner) to see how this affects the program. You can also move the robot using the track while drawing.

Program template

```
MODULE MainModule
CONST jointtarget jpos10:=[[0,0,0,0,0,0],[0,0,0,9E+09,9E+09,9E+09]];
TASK PERS wobjdata wobj1:=[FALSE,FALSE,"STN1",[[0,0,0],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
CONST robtarget p10:=[[200,0,100],[0.7071,0,0.7071,0],[0,0,0,0],[0,0,0,9E+09,9E+09,9E+09]];
CONST robtarget p20:=[[1662.50,0.00,2055.00],[0.707107,0,0.707107,0],[0,0,0,0],[0,0,0,9E+09,9E+09,9E+09]];
PROC main()
    MoveAbsJ jpos10\NoEOffs, v1000, z50, tool0;
   !MoveJ p10, v1000, z50, tool0\WObj:=wobj1;
ENDPROC

PROC DrawLiuLogo(
     PERS tooldata LocalTool,
     PERS wobjdata LocalWorkObj)

 ENDPROC

 PROC TestInterpSingPos()
    SingArea\Wrist;
    MoveL Offs(p20,0,-400, 0), v100, fine, tool0;
    MoveL Offs(p20,0, 400, 0), v100, fine, tool0;
    SingArea\Off;
    ConfL\Off;
    MoveL Offs(p20,0,200, 20), v100, fine, tool0;
    MoveL Offs(p20,0, -200, 20), v100, fine, tool0;
    ConfL\On;
 ENDPROC

ENDMODULE
```