LCAMP

Learner Centric Advanced Manufacturing Platform

FANUC INDUSTRIAL ROBOTS PROGRAMMING

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

ROBOTIC PROGRAMME

2.1.3 Program

A program contains motion instructions, input/output instructions, register instructions, and branch instructions. (For the program structure, see Chapter 4.) Each instruction is assigned a statement number. The target work is accomplished by sequentially executing the instructions.
The teach pendant is used to create or correct a program. (For creation of a program, see Chapter 5.) The program contains the following instructions. Fig.2.1.3 shows a basic program for manipulating workpieces.

- **Motion instruction**: Moves the tool to the target position within the operating range.
- **Additional motion instruction**: Performs an additional (special) operation during a motion.
- **Register instruction**: Places (loads) numerical data into a register.
- **Position register instruction**: Places (loads) position data into a register.
- **Input/output instruction**: Sends or receives a signal to or from a peripheral unit.
- **Branch instruction**: Changes the flow of a program.
- **Wait instruction**: Holds execution of the program until the specified conditions are satisfied.
- **Routine call instruction**: Calls and executes a subprogram.
- **Macro instruction**: Calls a specified program and executes it.
- **Palletizing instruction**: Palletizes workpieces.
- **Program end instruction**: Terminates execution of a program.
- **Comment instruction**: Adds a comment to a program.
- **Other instructions**

FANUC INDUSTRIAL ROBOTS PROGRAMMING

LCAMP
Learner Centric Advanced Manufacturing Platform

LEARNING/PROGRAMMING A ROBOT

- robot program
- selecting/opening a program
- program parameters
- testing (T1/T2) of the program
- program start (AUTO)
- robot program
- creation of a robot program
- editing the robot program
- archiving and restoration of robot programs

FANUC INDUSTRIAL ROBOTS PROGRAMMING

Learner Centric Advanced Manufacturing Platform

PROGRAMMING PHASES

We approach programming similarly to the development of other products.

- product development

- learning / programming an industrial robot

PROGRAMMING PHASES

Project task

- is a textual description, with graphic attachments (sketches, photographs, etc.), of an automated process, so that the programmer understands the automated process in detail
- presumably written by the client

Planning

- is a step-by-step description of a procedure (textual and/or graphical) that solves a given problem step by step
- text description
- flowchart chart)

Coding and co/documentation

- write the flowchart in a sequence of commands
- comments and descriptions for later understanding of the program and instructions for end users

PROGRAMMING PHASES

Testing and debugging

- Testing is checking the operation of a program before automatically executing it.
- is very important and must be carried out to ensure the safety of people and prevent damage to equipment

Product

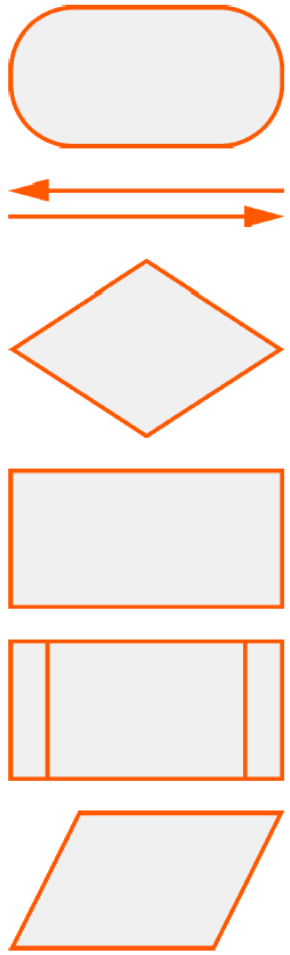
- a working (commented) program for an automated process including documentation

FANUC INDUSTRIAL ROBOTS PROGRAMMING

Learner Centric Advanced Manufacturing Platform

PROGRAMMING PLANNING

Flowchart



Beginn und Ende eines Prozesses oder Programms

Verknüpfung von Anweisungen und Operationen

Verzweigung

Allgemeine Anweisungen im Programmcode

Unterprogrammaufruf

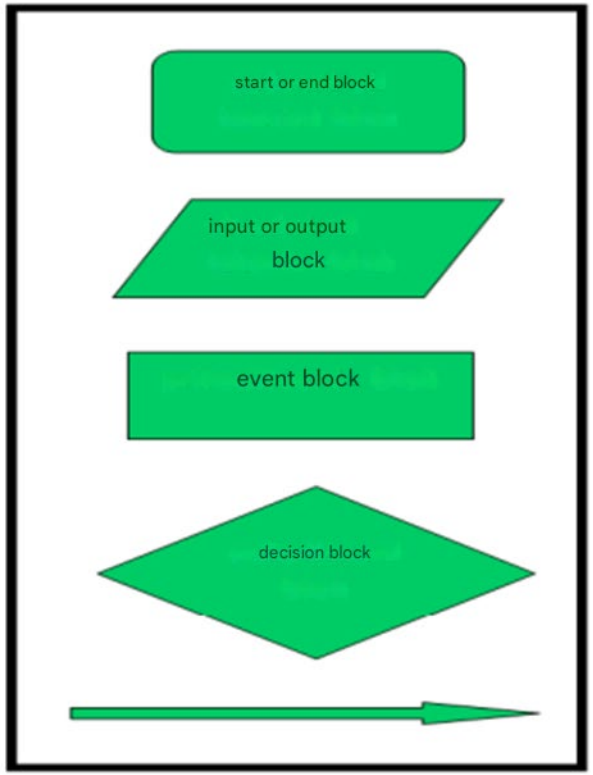
Ein-/Ausgabe Anweisung

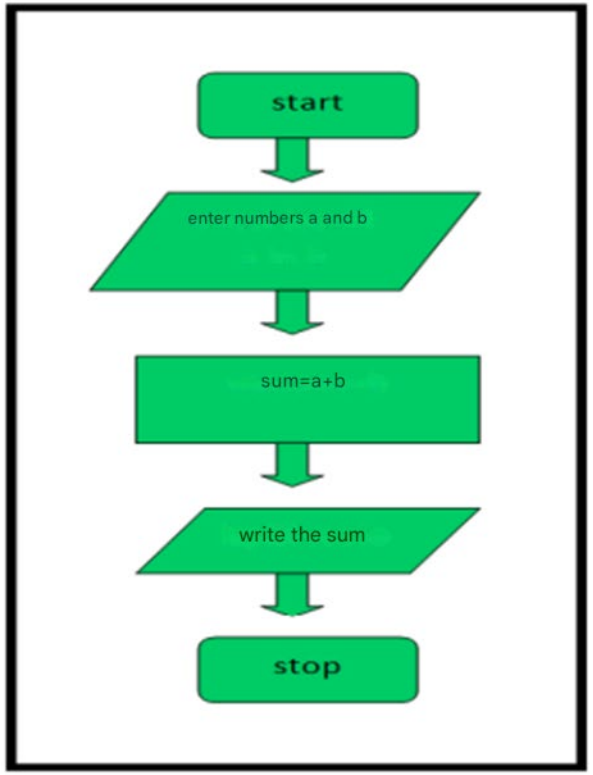
FANUC INDUSTRIAL ROBOTS PROGRAMMING

Learner Centric Advanced Manufacturing Platform

PROGRAMMING PLANNING

- Flowchart (Flowchart) Chart)
- It is a graphically displayed algorithm, i.e. a sequence of events described in a human-understandable way.
- It is used in planning programs and also in describing other activities (e.g., Family Doctor in book form, Municipal Instructions for Obtaining Permits , etc.).
- It is wise to approach a programming problem in a flowchart fashion to clarify the step-by-step implementation and anticipate what the computer will need to do.





FANUC INDUSTRIAL ROBOTS PROGRAMMING

Learner Centric Advanced Manufacturing Platform

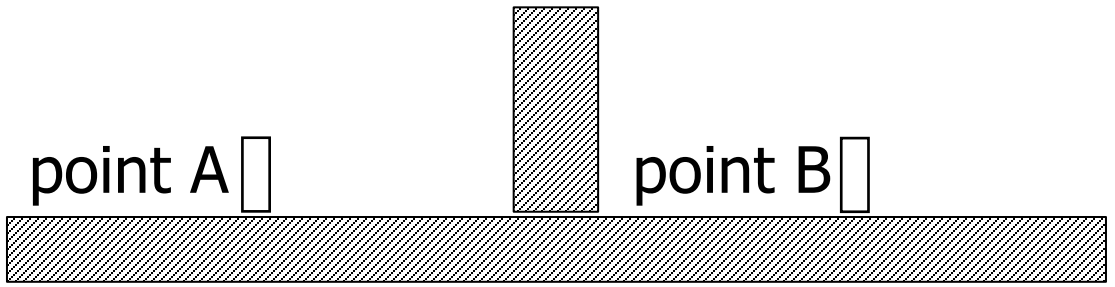
PROGRAMMING PLANNING

- A programmer must understand in detail the problem for which he is writing the program. His work can be broken down into the following tasks:
- Problem analysis is the development of a step-by-step procedure to solve a given problem. This procedure is called an algorithm.
- Program organization. You organize the algorithm into a sequence of operations using flowcharts. You can do this work on a personal computer if you have a flowchart drawing tool.
- Coding. You write the flowchart into a sequence of commands – a program using a text editor for your personal computer. Save it in a so-called source file.
- Linking and compilation. You link the program with other parts of the program that you have written in the past, or with programs that you get in a library (written by the manufacturer of the compiler and linker). The linking is done by a linker . You translate the entire program with a compiler into a format that uP understands . The linker and compiler are software for a personal computer.
- Inserting a program into a microcomputer is done using a programmer that is connected to a personal computer with the associated software.
- Testing. Test the program by running it on a microprocessor to determine whether it produces the appropriate results. To do this, create a test card. Correct any errors you discover by going back to the coding and repeating the process.
- Documentation. For later refinement or modification of the program, it is necessary to document the program. Edit the documentation with the necessary comments at all task points (technical description, report, study).

EXAMPLE OF PROGRAMMING PHASES

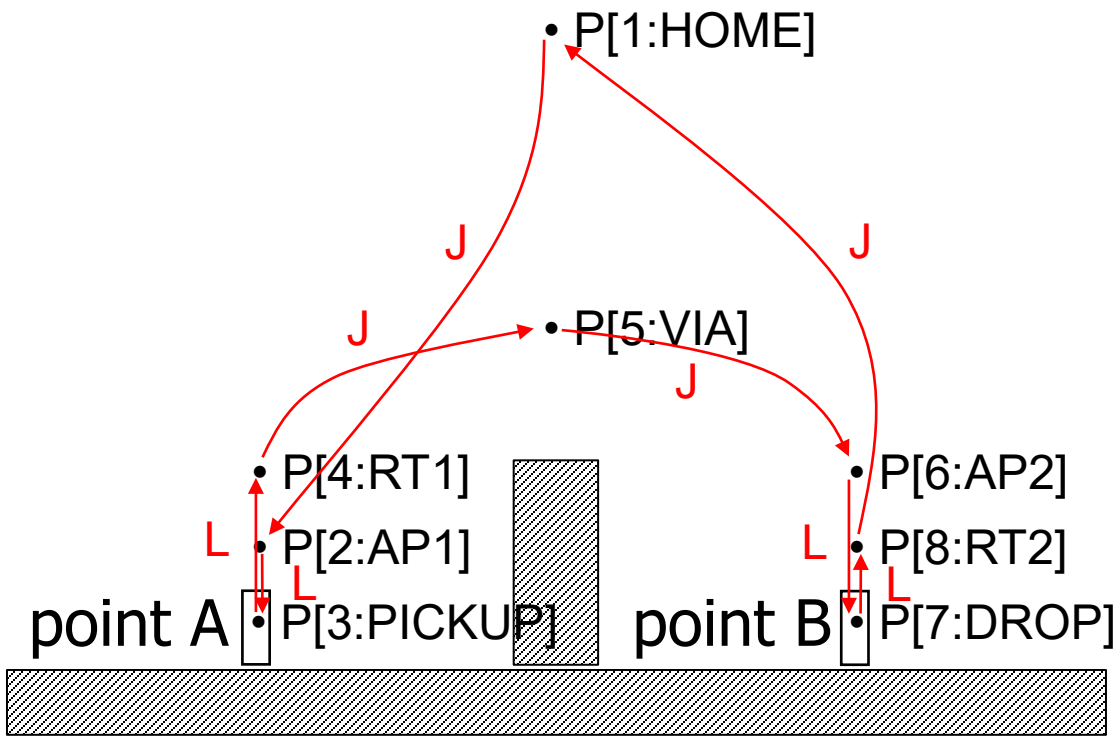
Learner Centric Advanced Manufacturing Platform

1. Project task
   - Create a robotic application that will pick up a product and drop it on the other side of an obstacle.



EXAMPLE OF PROGRAMMING PHASES

2. Planning (textual/graphic description and/or flowchart)
   - graphic description



EXAMPLE OF PROGRAMMING PHASES

Learner Centric Advanced Manufacturing Platform

2. Planning (textual/graphic description and/or flowchart)
   - text description
     - use a gripper for the tool,
     - use basic workspace
     - use general half speed
     - Quickly positions the TCP precisely at the starting point
     - TCP travels towards/past the approach point(s) with rapid movement
     - with linear movement and appropriate speed, it precisely places the TCP at the pick-up point
     - close the gripper
     - TCP travels towards/past the return points with linear motion
     - Move the product over the obstacle to the other side with a quick movement
     - …

EXAMPLE OF PROGRAMMING PHASES

Learner Centric Advanced Manufacturing Platform

2. Planning (textual/graphic description and/or flowchart)
   - flowchart

ROBOTIC PROGRAM

- **learning/programming** a robot **means** ,
  - to **manually guide the robot to the target point and save** the position or TCP coordinates ; at this point **we also set what the robot should do** ...
  - the robot **to the next point** ...

ROBOTIC PROGRAM

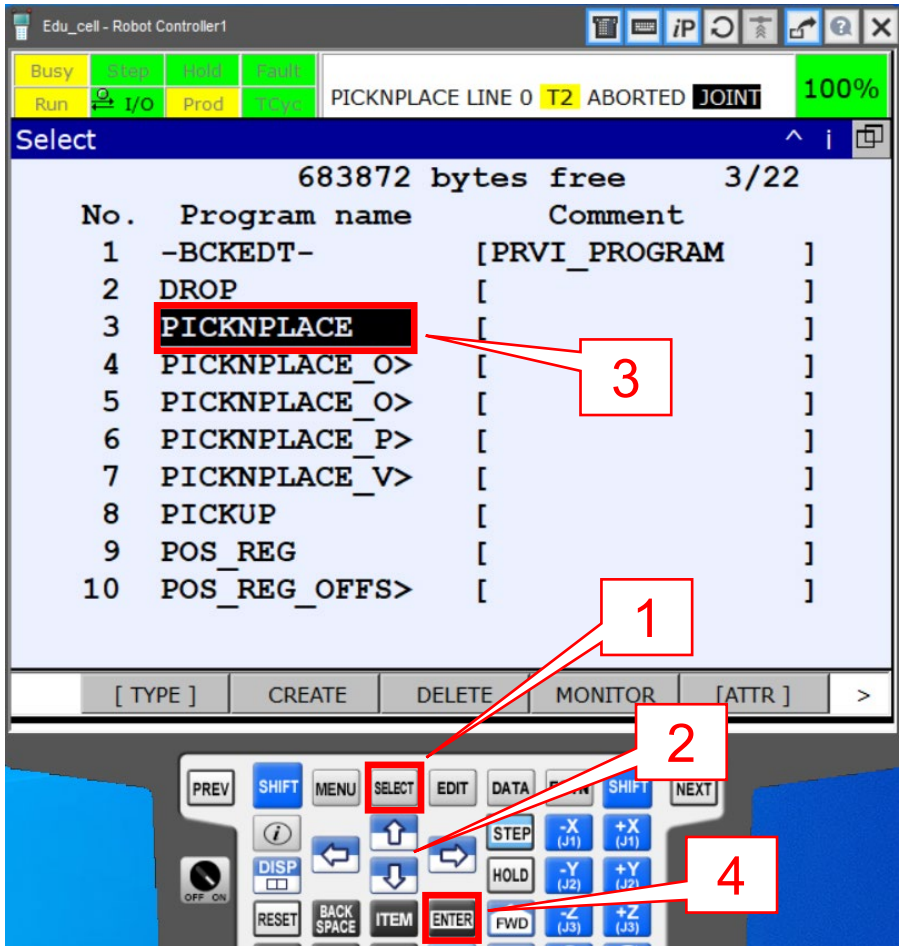
- **setting initial values**
  - **TF** – the tool we are using (determines the TCP location and tool orientation),
  - **UF** – the “space” in which we program
  - **payload** – dimensions, mass/inertia and center of gravity of the tool with/without products
  - **registers** or variables …
- **sequence of points – positions and orientations of TCP (tool or TF) in UF and**
- using **other commands**
  - LBL/JMP LBL ,
  - IF / SELECT,
  - FOR /ENDFOR,
  - WAIT, CALL …

EXAMPLE OF A ROBOT PROGRAM

```
  1: UTOOL_NUM=1
  2: UFRAME_NUM=1
  3:
  4: LBL[1:LOOP]
  5:
  6: JP[1] 50% CNT100
  7: LP[2] 100mm/s FINE
  8: JP[3] 50% FINE
  9:
 10: IF DI[101]=ON, JMP LBL[1]
 11:
 12: JP[4] 50% CNT100
[END]
```

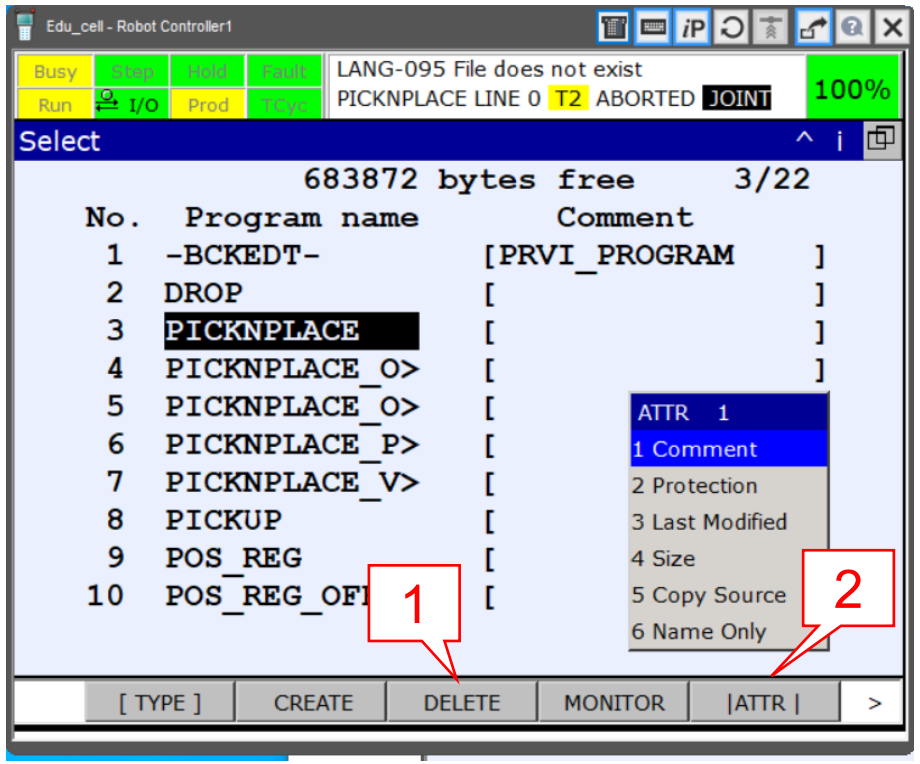
SELECTING OR OPENING A PROGRAM

1. on the UE click SELECT,
2. in the list with arrows ↑ ↓
3. find the desired program and
4. confirm selection with ENTER



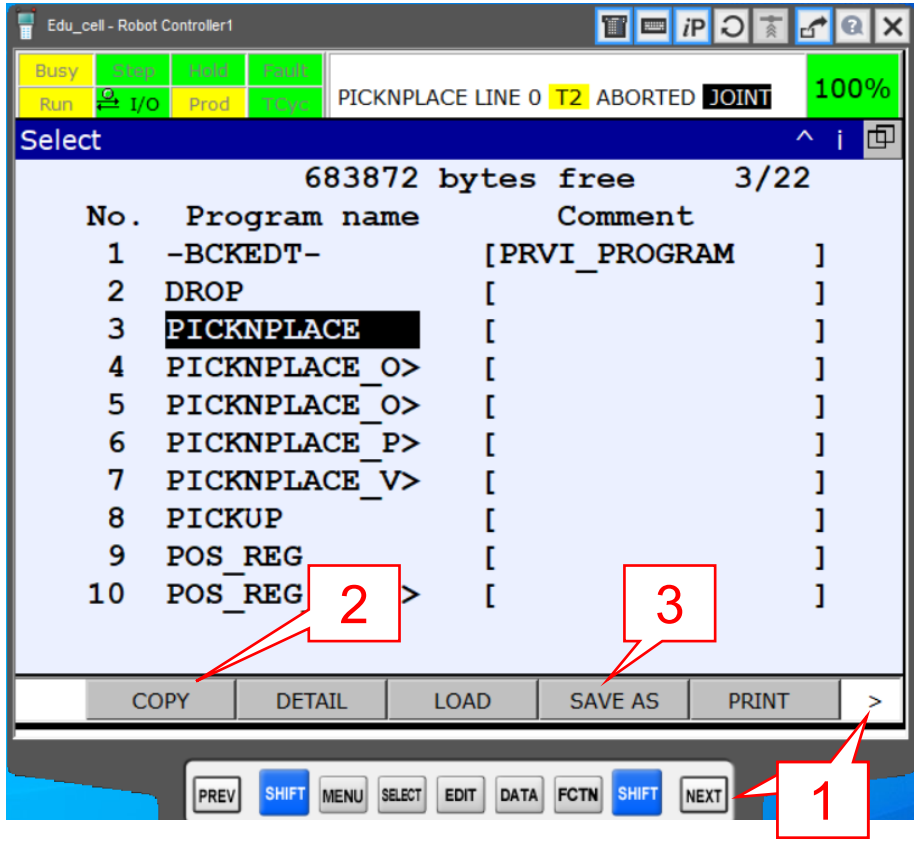
PROGRAM PARAMETERS

- in the Select window , the program can
  1. by clicking F3 DELETE you delete
  2. by clicking on F5 [ATTR] you can view its parameters
     1. or add a comment (Comment)
     2. protection
     3. last saved (Last Modified)
     4. size
     5. source (Copy Source)
     6. display only the name (Name Only)



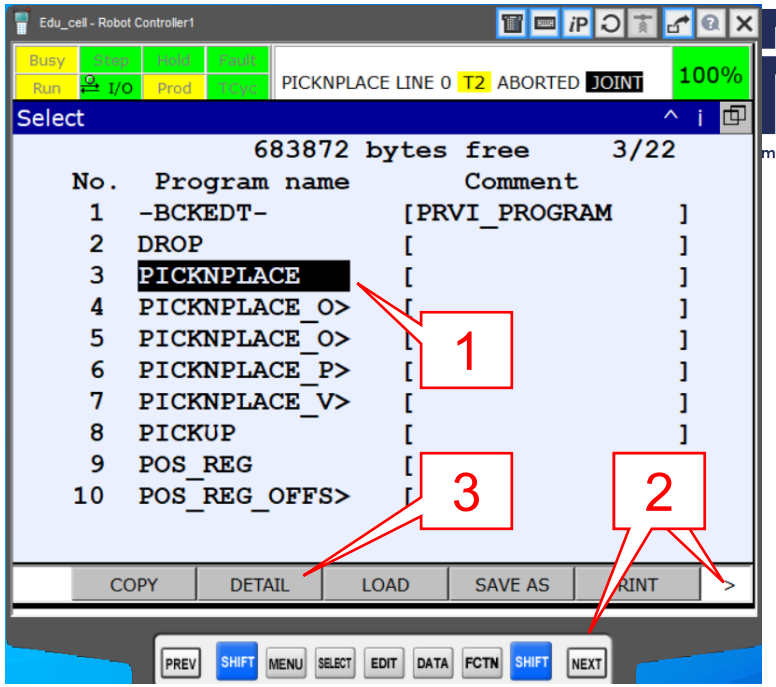
PROGRAM PARAMETERS

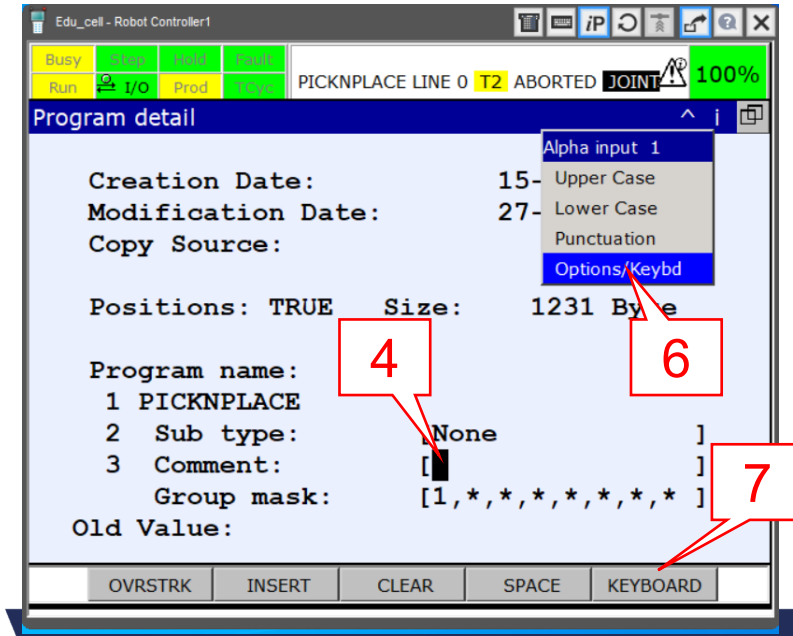
- By clicking on NEXT or the > symbol, the program can:
  1. by clicking F1 COPY you copy or
  2. Save under a different name by clicking F4 SAVE AS

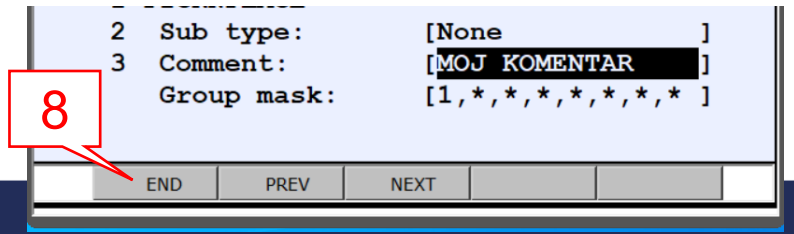


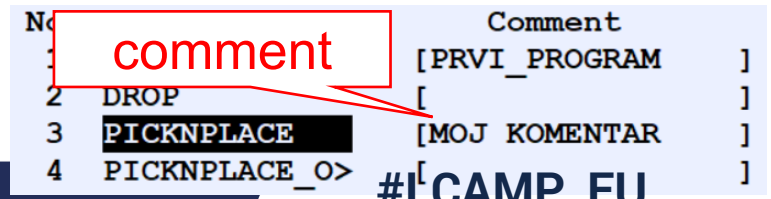
PROGRAM PARAMETERS

- program commentary
  1. Select the desired program from the list.
  2. by clicking on NEXT or the > symbol, switch to
  3. F2 DETAIL and click on it
  4. scroll to Comment in the list
  5. click ENTER
  6. select Option/Keyboard
  7. press F5 KEYBOARD and enter a comment
  8. end with ENTER and F1 END









#LCAMP_EU

TESTING (T1/T2) OF THE ROBOT PROGRAM

- Testing is **checking the operation of a program** before automatically executing it.
  - is very important and **must** be carried out to ensure the safety of people and prevent possible damage to equipment
  - **step mode** (STEP)

    - execution from the current program line, line by line
  - **continuous mode** (CONT)

    - program execution from the current program line to the end of the program, the [END] mark, or until an error occurs/testing is aborted

TESTING AND START-UP

testing (and debugging) on the robot :

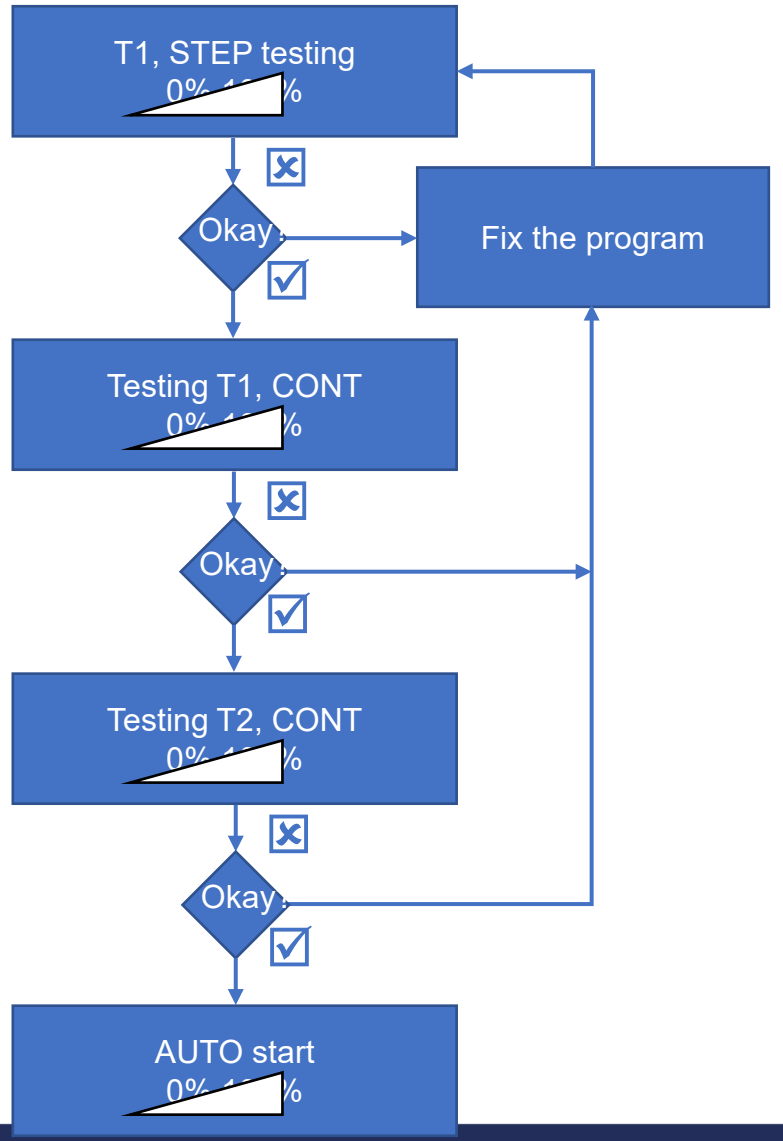
1. T1, step mode, low speed
2. T1, continuous mode, low/medium/high speed
3. T2, continuous mode, low/medium/high speed

testing (and debugging) in RoboGuide :

1. T2, step mode, low speed
2. T2, continuous mode, low/medium/high speed

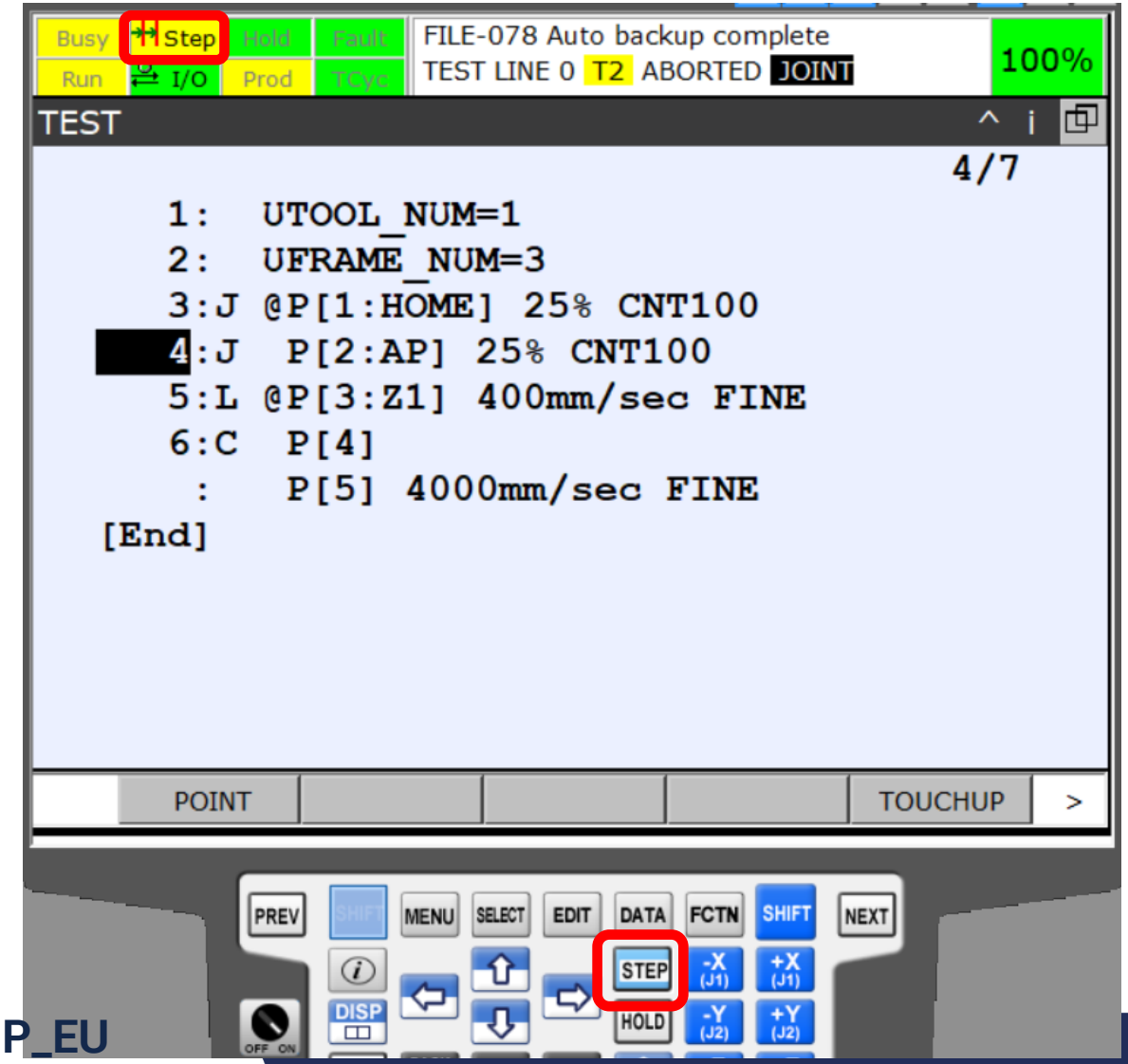
startup:

1. AUTO mode, programmed speed



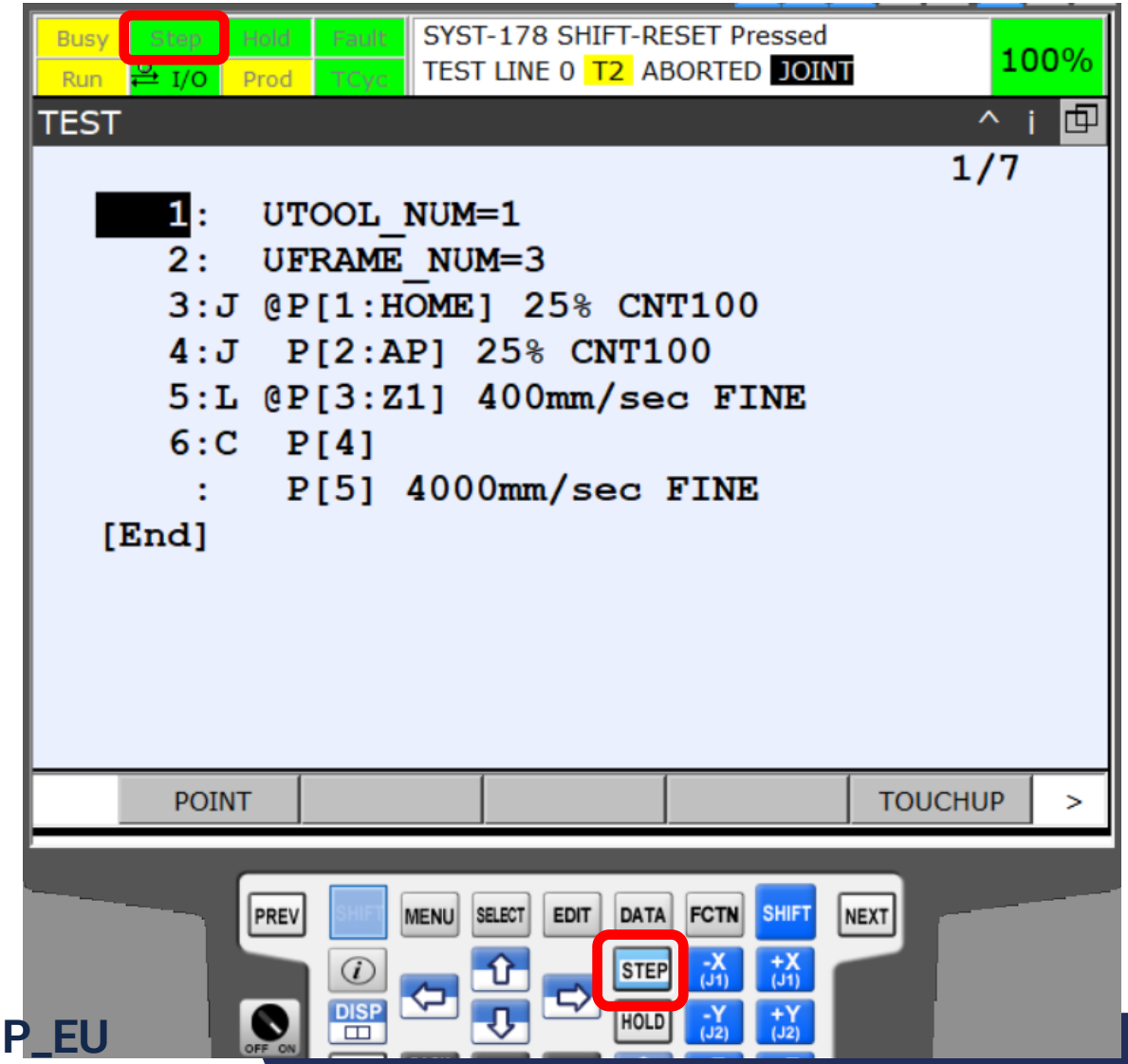
TESTING (T1)

- **step mode** (STEP)
  - set T1 on the controller
  - enable the learning unit (ON)
  - select a program (SELECT …)
  - position yourself in the desired program line
  - turn on STEP (icon on STEP light)
  - set low speed
  - activate the permission button and clear the errors (RESET)
  - Test the program line by line forward/backward by pressing the SHIFT + FWD/BWD combination
  - The @ sign indicates the current program line or at which point in the program the TCP is located.



TESTING (T1, T2)

- **continuous mode**
  - set T1 or T2 on the controller
  - position yourself in the desired program line
  - turn off STEP (no icon on STEP light)
  - set low/medium/high speed
  - activate the permission button and clear the errors (RESET)
  - by pressing the combination SHIFT + FWD test the program from the selected program line to the end [End] or until a possible interruption/error
  - The @ sign indicates the current program line or at which point in the program the TCP is located.



PROGRAM START (AUTO)

- if necessary, select a program (SELECT …)
- Place yourself in the desired program line (presumably the first one)
- set the operating mode to AUTO on the controller
- disable the learning unit (OFF)
- clear errors (RESET)
- Start the program by pressing the CYCLE START button on the controller.
- The @ sign indicates the current program line or at which point in the program the TCP is located.

REPEAT

Learner Centric Advanced Manufacturing Platform

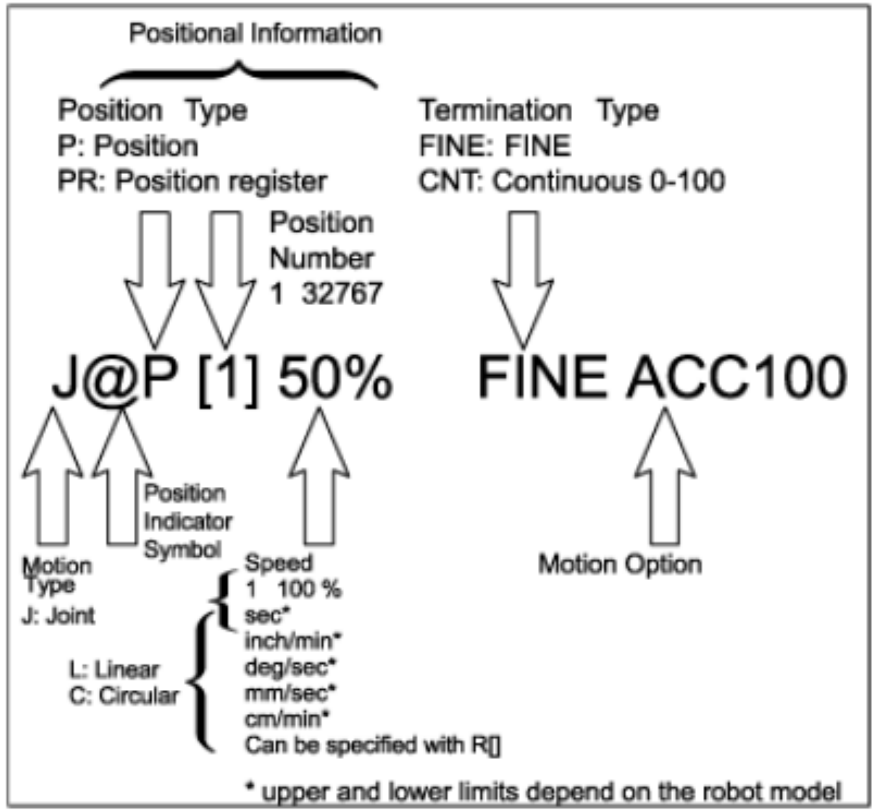
1. What does it mean to teach or program a robot? What is a robot program?
2. Explain what you specify in the robot program with the commands UTOOL_NUM and UFRAME_NUM.
3. What does it mean to select a robot program? What can we do with the program in the Select window ?
4. Explain the difference between testing and running a program. Why is program testing very important?
5. What testing methods do we know and explain the correct procedure for testing a program.
6. Describe the step-by-step method of testing a program.
7. Describe the continuous method of program testing.
8. Describe the program startup.

MOVE COMMAND

- example of a movement command:

J @ P[1] 100% FINE ACC100
a) b) c) d) e) f)

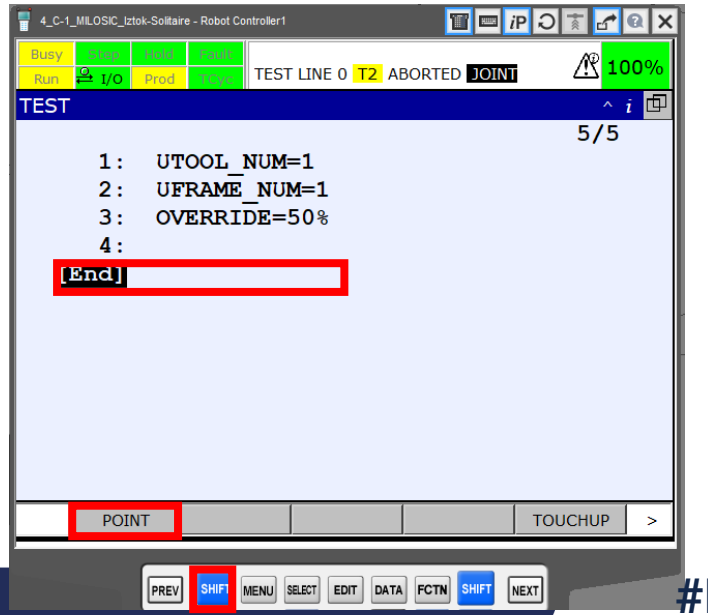
a) movement mode (J, L, C)
b) tool location tag (TCP)
c) TCP point, position and orientation (P[1], PR[2,1])
d) speed (mm/sec, %, deg /sec, sec …)
e) precision, termination (FINE, CNT)
f) additional motion parameter(s)

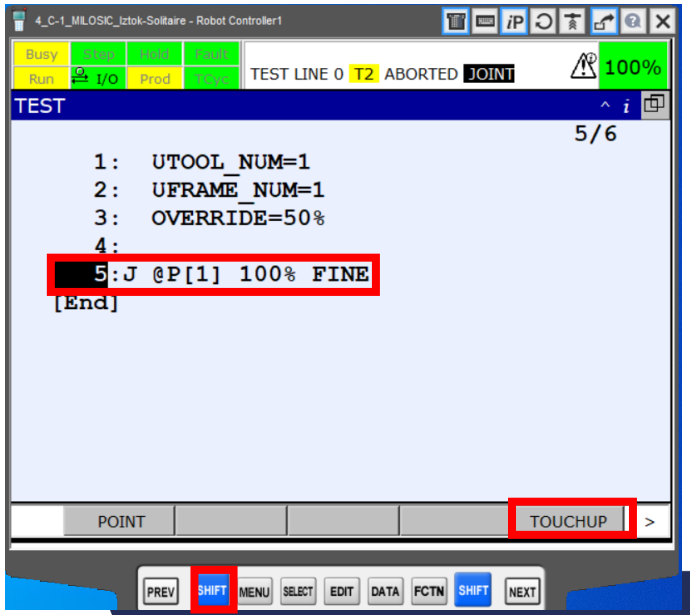


MOVE COMMAND

- SHIFT + F1 POINT – create a movement command in the desired line
- SHIFT + F5 TOUCHUP – correct TCP and UF coordinates and TF in the movement command line

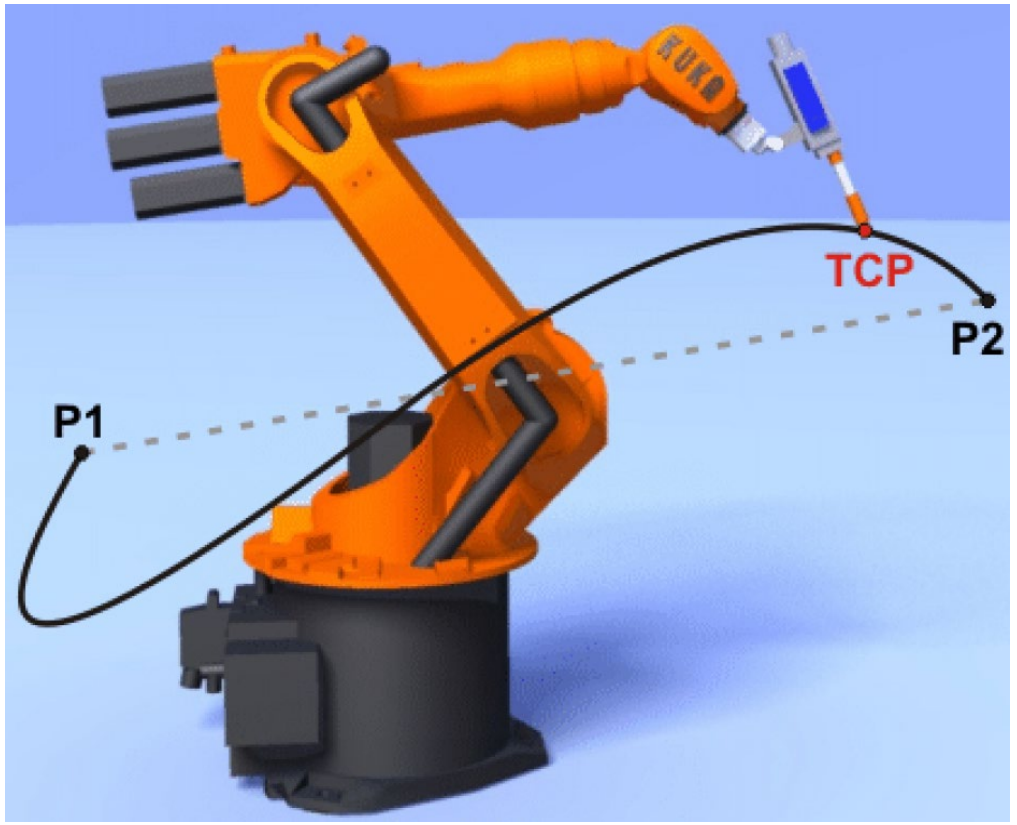
ALWAYS FIRST PUT YOURSELF IN THE DESIRED LOCATION WITH TCP AND ONLY THEN EXECUTE ONE OF THE ABOVE COMMANDS!!!





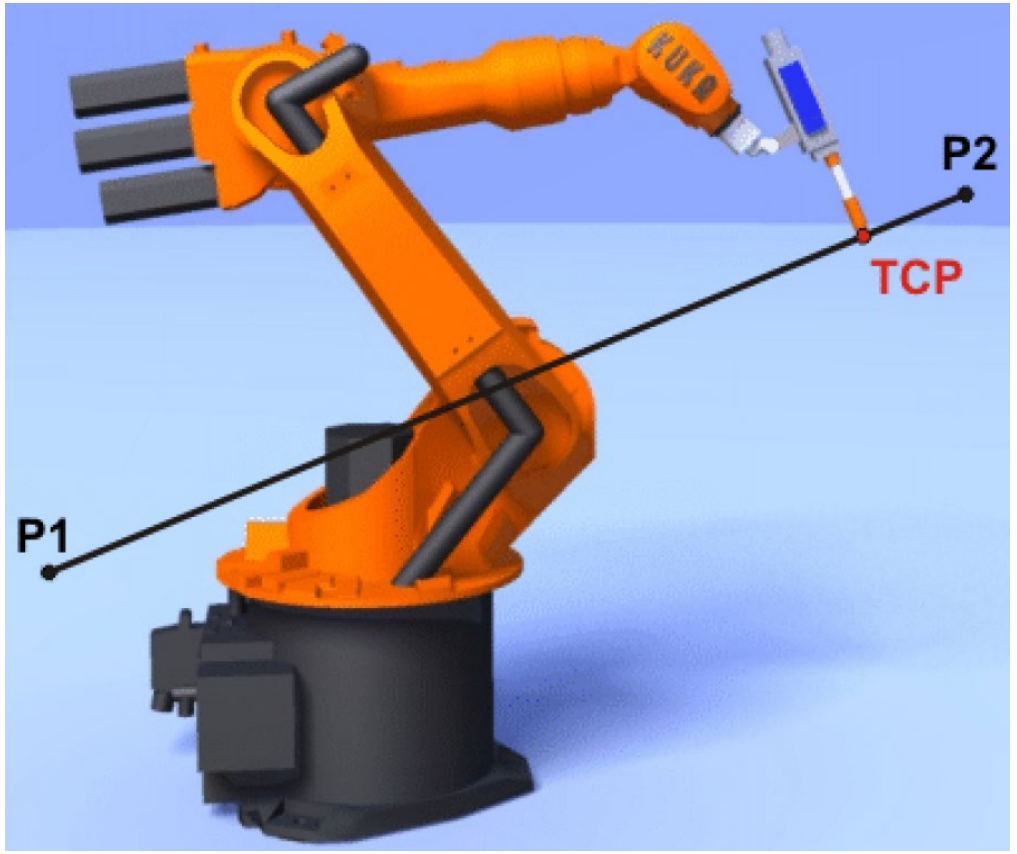
a) MOVEMENT METHOD

- **free movement J (joint)**
- the robot itself calculates the fastest path or trajectory
- is the fastest and simplest movement for a robot
- **The path, speed and orientation of the tool (TCP) are not known in advance** , so it must be used with caution.
- **axial characteristic movement**



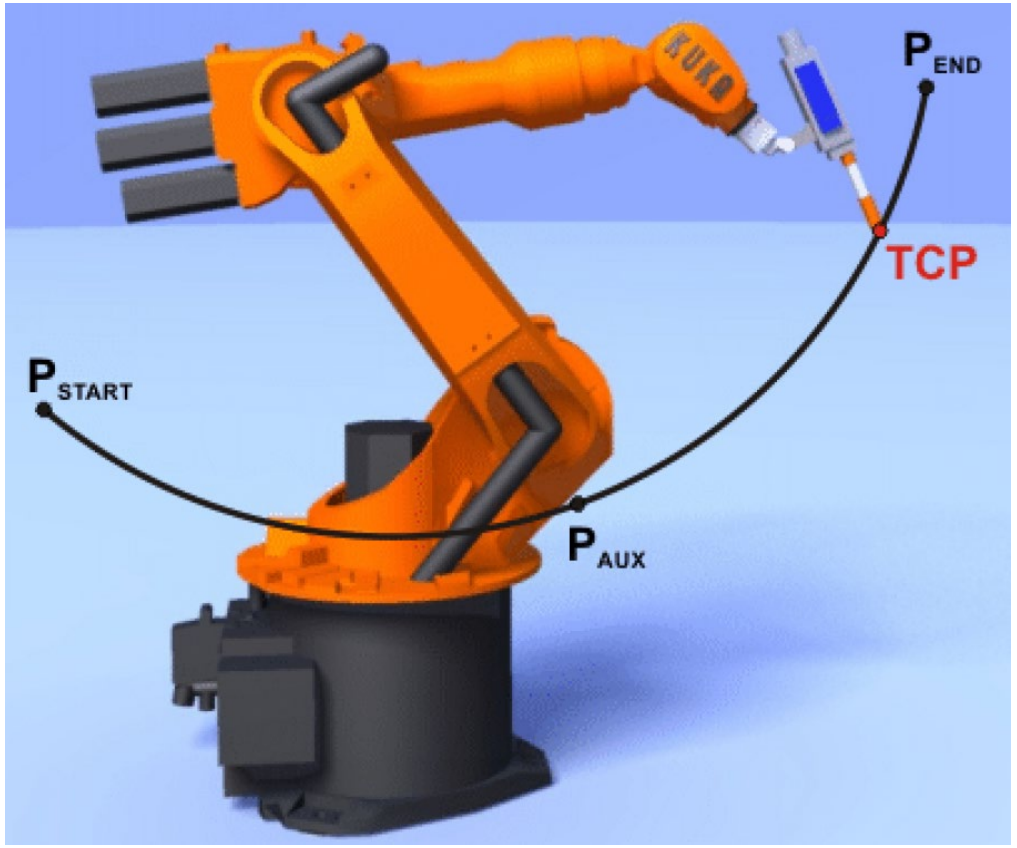
a) MOVEMENT METHOD

- **linear motion L (linear)**
- the tool (TCP) moves linearly from the starting point to the end point
- **the path of movement, speed and orientation of the tool (TCP) are known in advance**
- **TCP movement along the path**



a) MOVEMENT METHOD

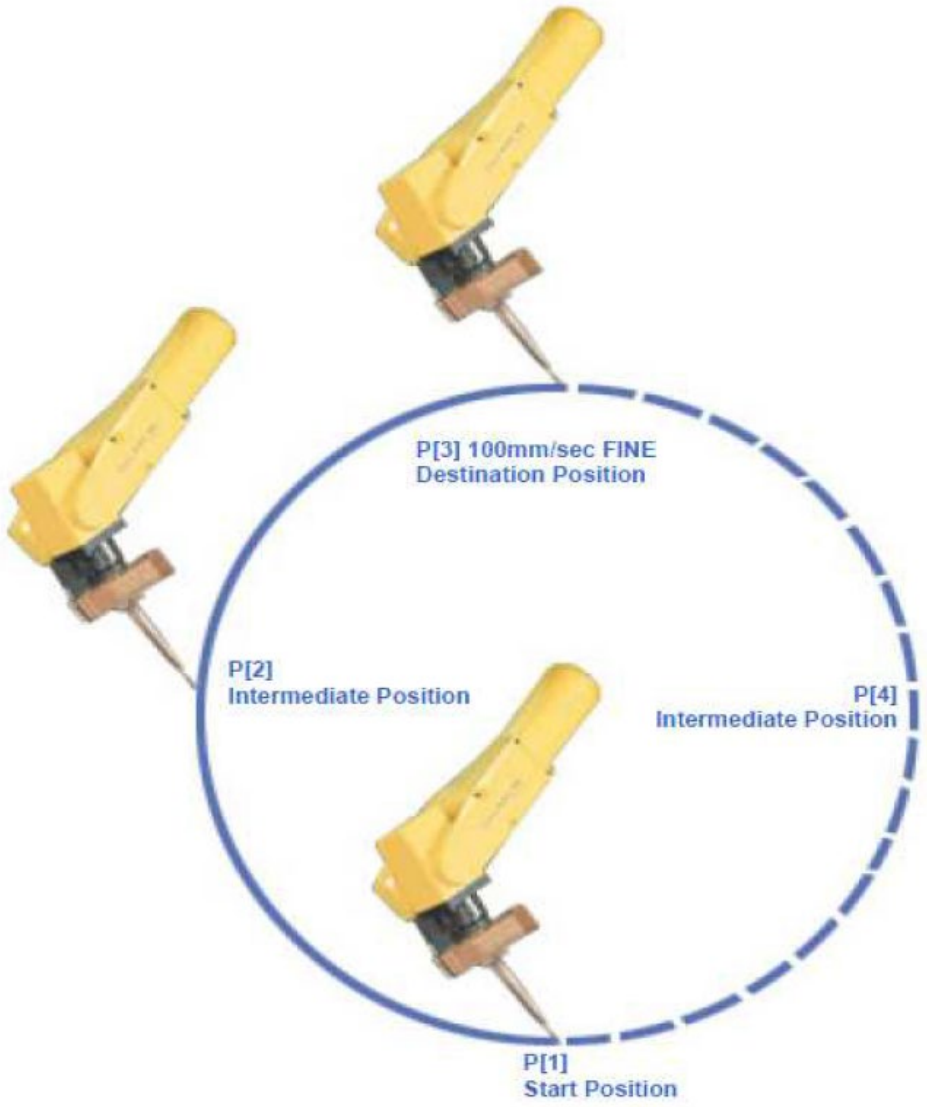
- **circular motion C (circular)**
- the tool (TCP) moves in an arc from the starting point P[1] through the intermediate point P[2] to the end point P[3]
- **the path of movement, speed and orientation of the tool (TCP) are known in advance**
- **TCP movement along the path**



a) MOVEMENT METHOD

- **circular motion C (circular)**
- we can only perform half a circle with one command – for a full circle we need two semicircular movements
- The circular motion command contains the intermediate point P[2] and the end point P[3], the starting point P[1] is contained in the previous command.

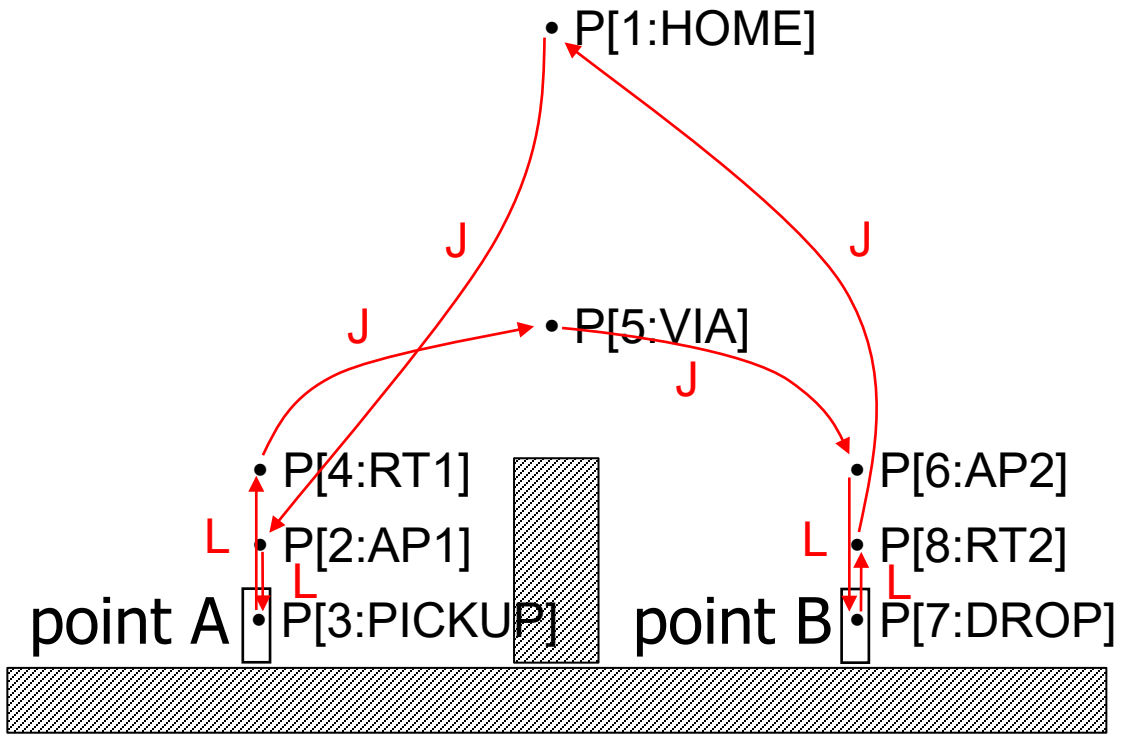
1: JP[1] 100% FINE
2: CP[2]
: P[3] 100mm/sec FINE
3: CP[4]
: P[1] 100mm/sec FINE



b) OBSTACLES

Example:

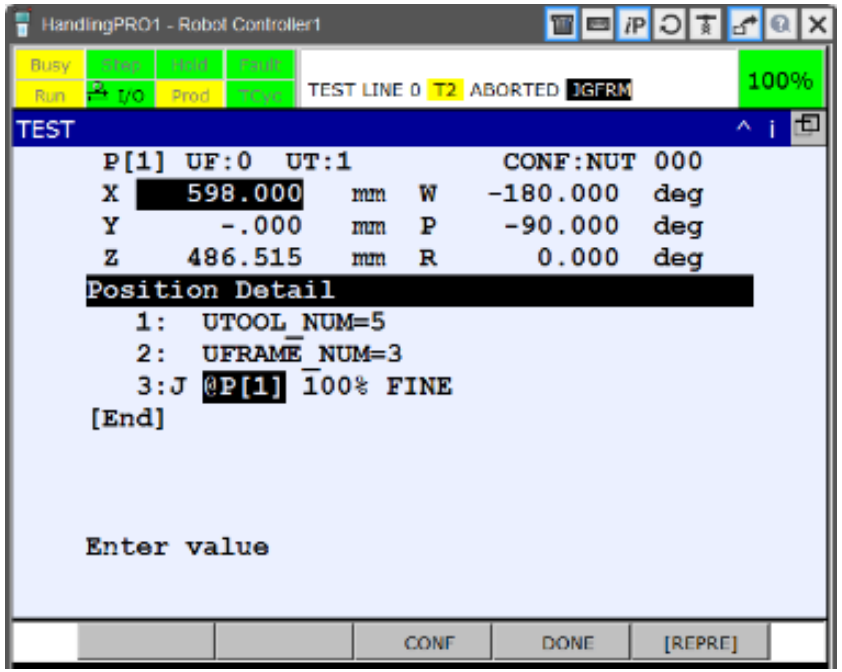
- You want to use a robot to transport an object from point A over an obstacle to point B.
- the robot starts at the starting point and returns there
- plot all TCP points
- explain/write the movements you would use



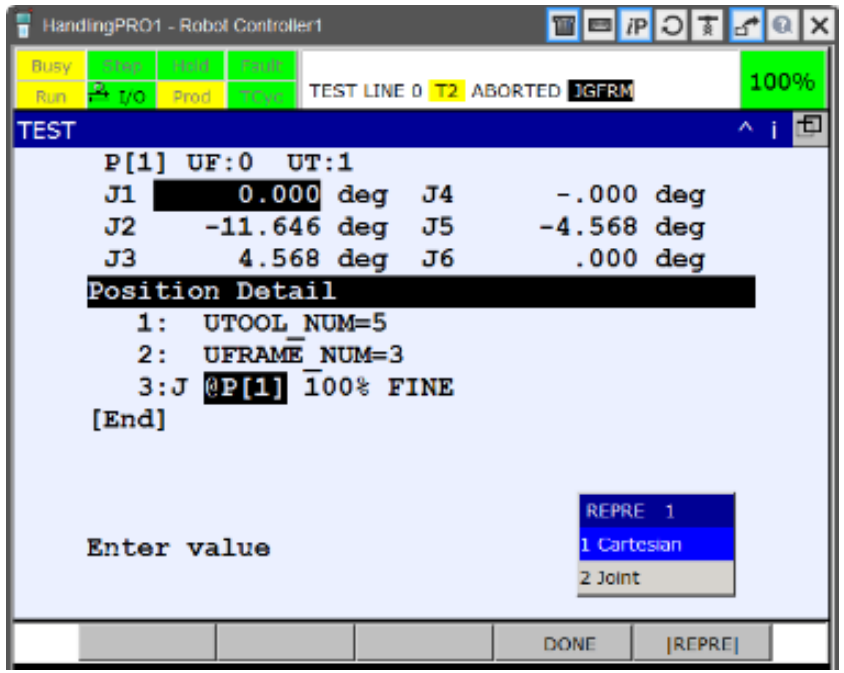
c) TCP POSITION AND TOOL ORIENTATION

- at point P[ ] the position and orientation of TCP is stored in:
  1. **Cartesian format** : X, Y, Z, W, P, R, UTOOL and UFRAME and CONFIG **or**
  2. **axially characteristic shapes** : J1, J2, J3, J4, J5, J6, UTOOL and UFRAME

Cartesian coordinate form

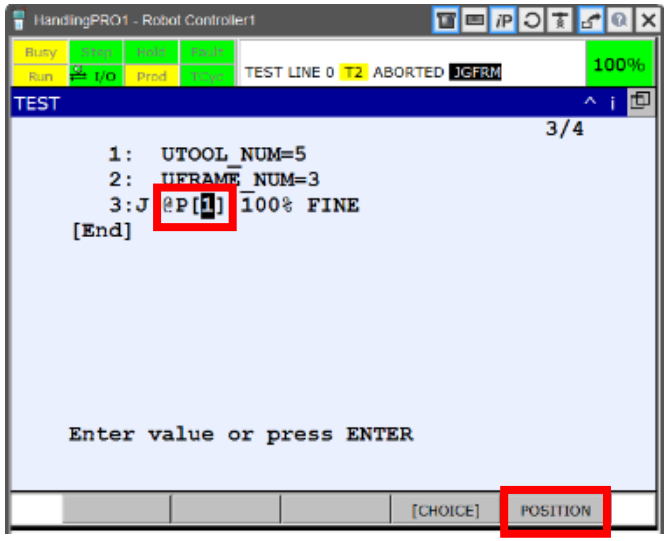


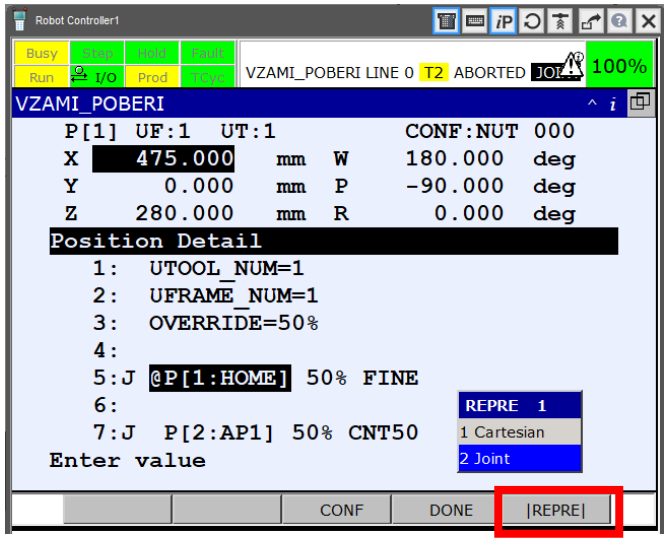
Axis-specific coordinate form

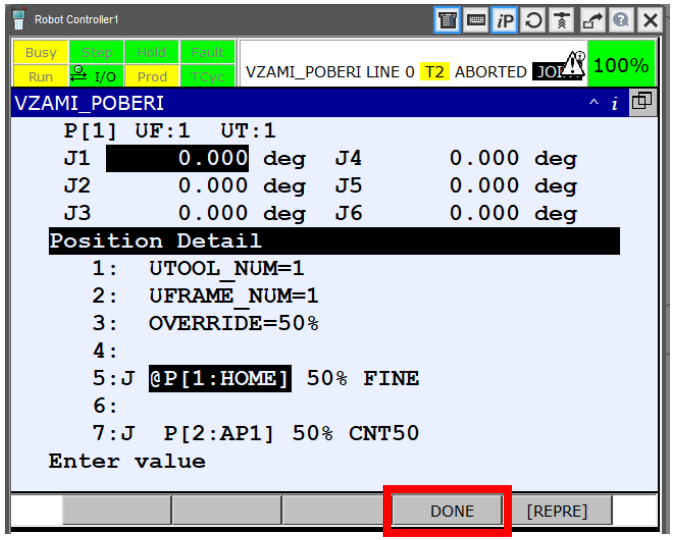


c) TCP POSITION AND TOOL ORIENTATION

- **You can view/change the coordinates** of point P[ ] :
  - **use the keyboard to position yourself at the point and click F5 [POSITION]**
  - **F5 [REPRE]** , switch between
    - **axial coordinates** , **Joint** (J1, J2, J3, J4, J5, J6) or
    - **Cartesian coordinates** , **Cartesian** (X, Y, Z, W, P, R)
  - enter/change the desired coordinates and confirm with F4 DONE







c) TCP POSITION AND TOOL ORIENTATION

Learner Centric Advanced Manufacturing Platform

Configuration

A configuration represents the attitude of the robot. Several configurations are available which meet the condition of Cartesian coordinates (x, y, z, w, p, r). The Turn Number and Joint Placement of each axis must be specified.

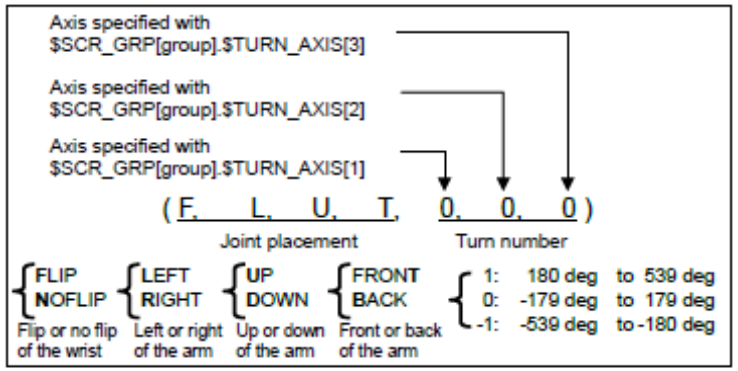


Fig. 4.3.2 (c) Configuration

- **Joint placement**

Joint placement specifies the placement of the wrist and arm. This specifies which side the control point of the wrist and arm is placed on against the control plane. When a control point is placed on the control plane, the robot is said to be placed at a singular point, or to be taking a peculiar attitude. At the singular point, since the configuration can not be decided to one by the specified Cartesian coordinate values, the robot can not move.

- An operation that ends at a singular point cannot be programmed. (In some cases, the most feasible configuration can be selected.) To specify such an operation, define the axial coordinate values.
- During linear or circular motion or circle arc motion, the tool cannot pass through a singular point (the joint placement cannot be changed). In this case, execute a joint motion. To pass through a singular point on the wrist axis, a wrist joint motion (Wjnt) can also be executed.

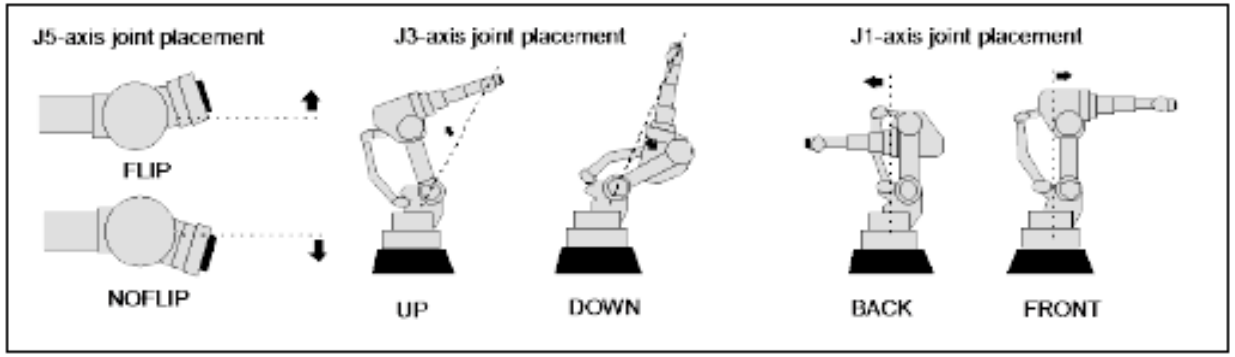


Fig. 4.3.2 (d) Joint placement

Turn number

Turn number represents the number of revolutions of the wrist axis (J4, J5, J6). Each axis returns to the original position after one revolution. So, specify how many turns have been made. Turn number is 0 when each axis is at an attitude of 0.

The turn numbers for up to three axes can be displayed. The axis number to correspond to each field is specified with system variable $SCR_GRP[i].$TURN_AXIS[j] (where i is a group number), as follows:

Left field : Axis number specified with $SCR_GRP[i].$TURN_AXIS[1]
Middle field : Axis number specified with $SCR_GRP[i].$TURN_AXIS[2]
Right field : Axis number specified with $SCR_GRP[i].$TURN_AXIS[3]

When programmed linear motion or circular motion or circle arc motion is executed, the robot tool moves toward the target point while adopting an attitude very similar to that at the start point. The number of revolutions performed at the target point is selected automatically. The actual number of revolutions performed at the target point may differ from the number specified in the position data.

c) TCP POSITION AND TOOL ORIENTATION

LCAMP
Learner Centric Advanced Manufacturing Platform

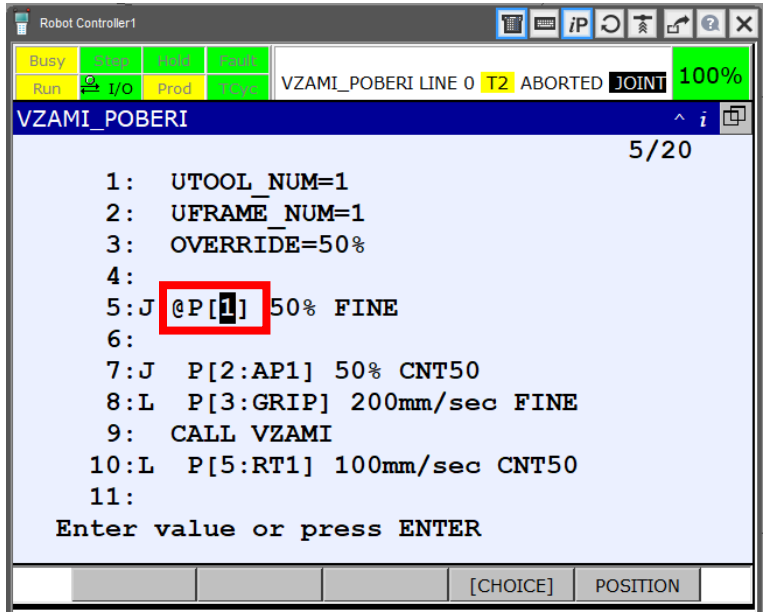
Cartesian coordinate system reference

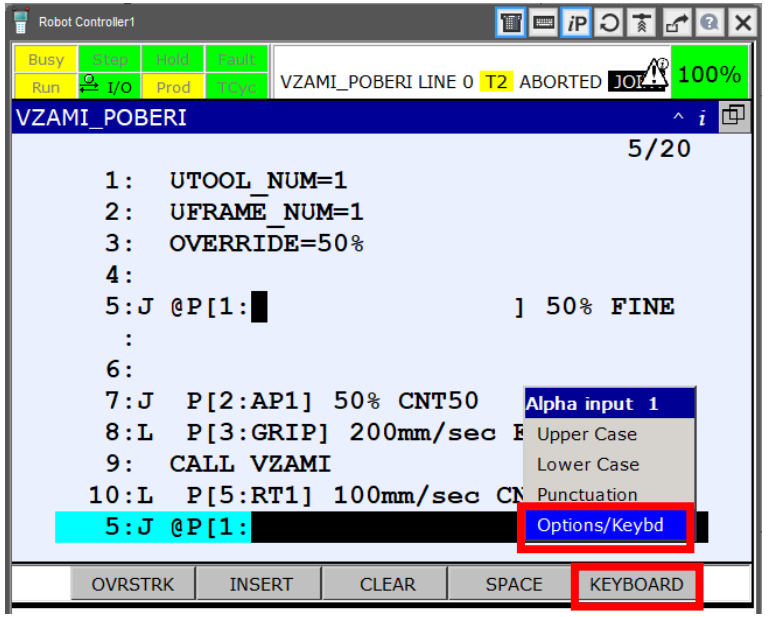
In playback of position data consisting of Cartesian coordinates, a Cartesian coordinate system reference checks the coordinate system number of a Cartesian coordinate system to be used.
If the coordinate system number (a number from 0 to 10 for the tool coordinate system, and a number from 1 to 9 for the user coordinate system) specified in the position data does not match the coordinate system number currently selected, the program is not executed for safety, and an alarm is issued.
A coordinate system number is written into position data in position teaching.
To change a coordinate system number after it has been written, use the tool replacement/coordinate replacement shift function.

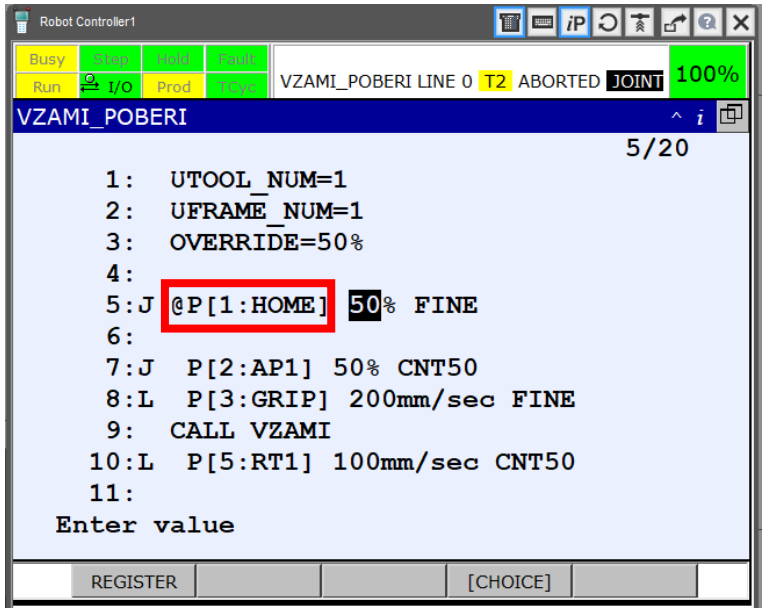
c) TCP POSITION AND TOOL ORIENTATION

Learner Centric Advanced Manufacturing Platform

- **enter/change the name** of the point P[ ] :
  - Use the arrow keys to position yourself at point P[ ] and click ENTER
  - from the menu select Options / Keyboard and press F5 KEYBOARD
  - use the keyboard to enter/change the name of the point and confirm with ENTER

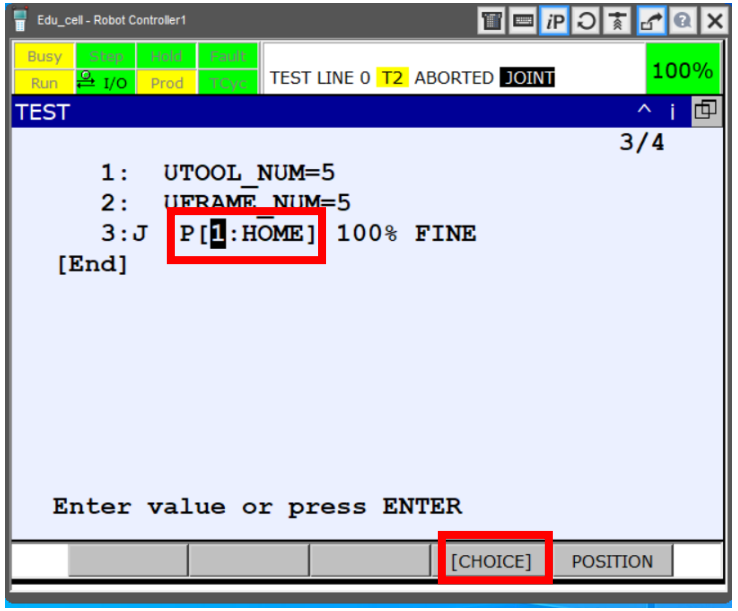


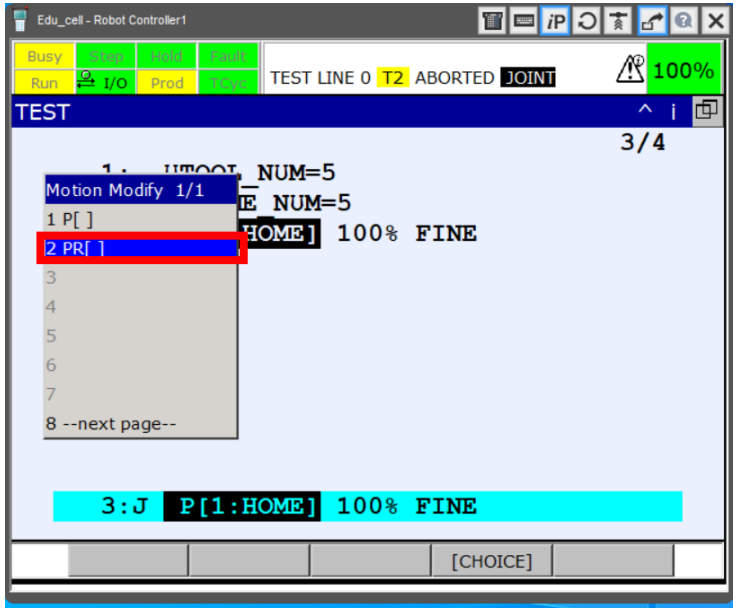


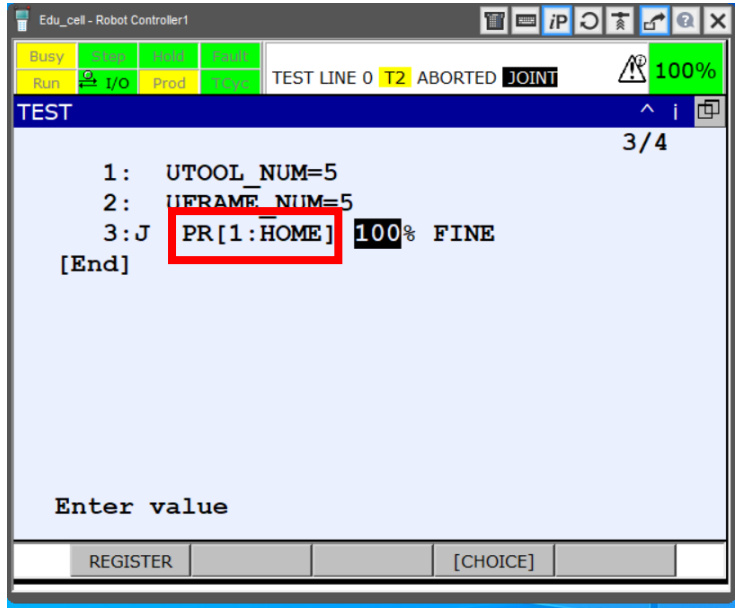


c) TCP POSITION AND TOOL ORIENTATION

- The TCP point can also be stored in the position register PR[ ]
  - **operate with the coordinates** in PR[ ] (+, -, *, / …) or **change them programmatically**
    - at point P[ ] you click F4 CHOICE
    - select PR[ ] and confirm with ENTER

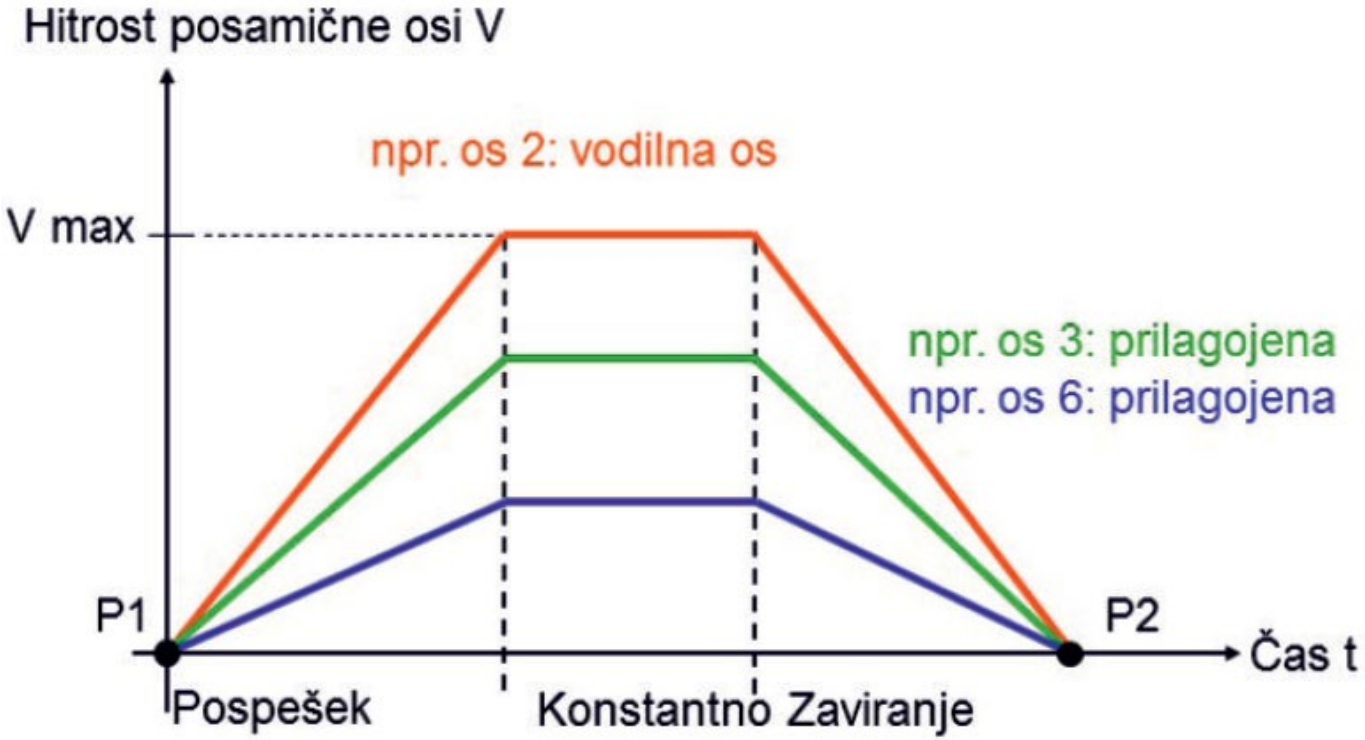






d) SPEED

- JOINT (% , sec)
  - it is not possible to enter a speed higher than the maximum speed of the robot
  - leading axis speed – the axis that takes the longest time to reach the target point

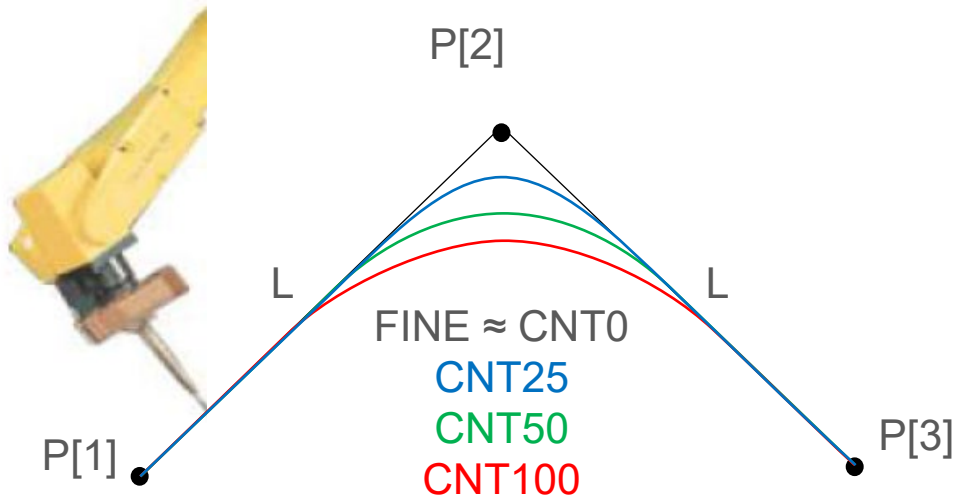


d) SPEED

- LINEAR, CIRCULAR (**mm/sec** , cm/min, inch/min, deg /sec, sec)
  - it is not possible to enter a speed higher than the maximum speed of the robot
  - TCP speed
  - in certain cases TCP does not reach the required speed, e.g. welding applications when the robot arm makes a large movement for a small tool rotation

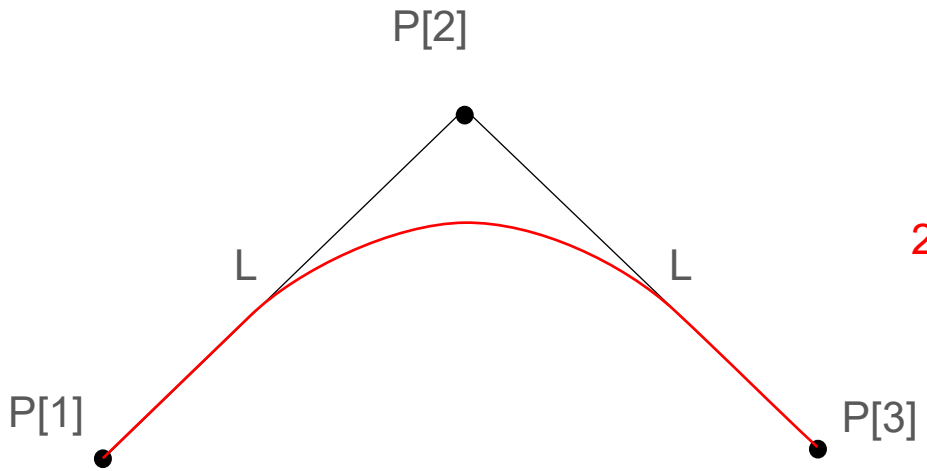
e) ACCURACY, TERMINATION

- determines how accurately TCP reaches a point
  - **FINE**
    - **the exact point and TCP stops**
  - **CNT** – CONTINUOUS
    - we determine how much **TCP approaches the point, TCP does not stop**
    - CNT set a value between 0 and 100
      - CNT0 ≈ FINE, TCP is closest to the point (**maximum delay**)
      - CNT100, TCP approaches the point far enough that the movement can be performed at full speed or without braking

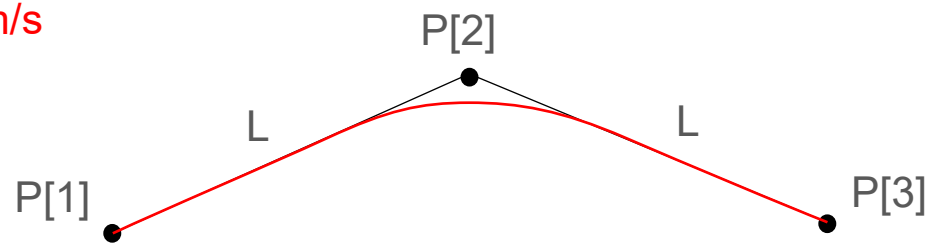


e) ACCURACY, TERMINATION

- Example 1

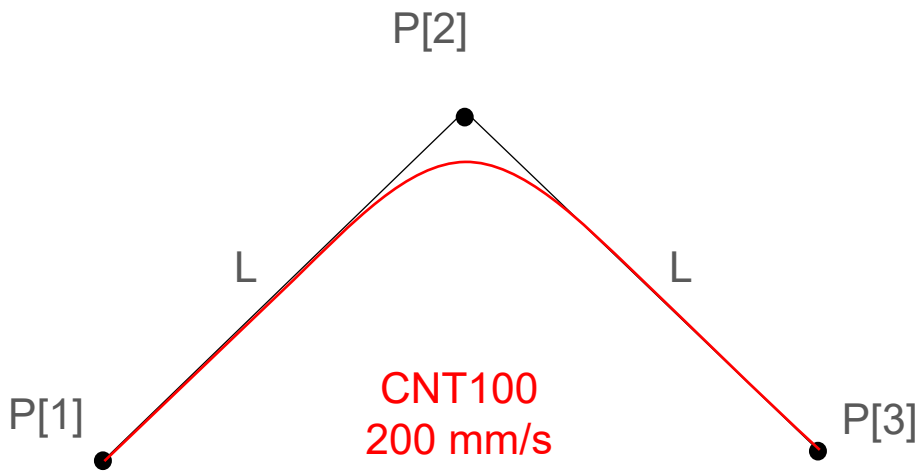


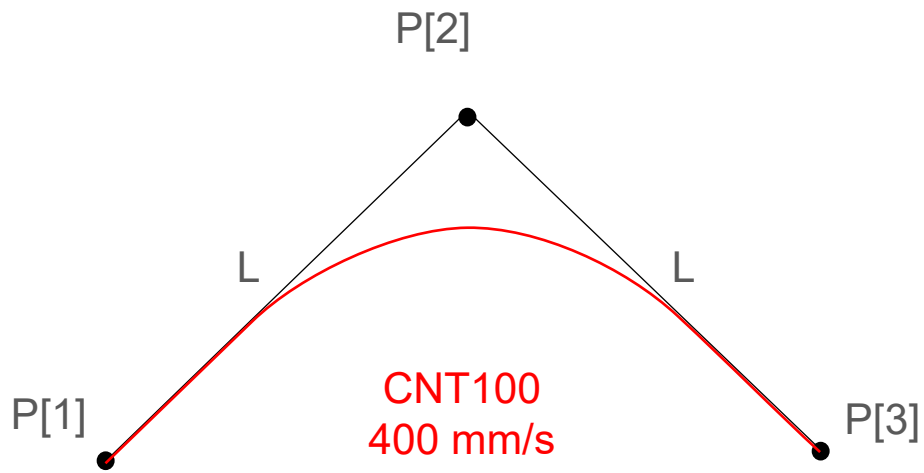
CNT100
200 mm/s



e) ACCURACY, TERMINATION

- Example 2



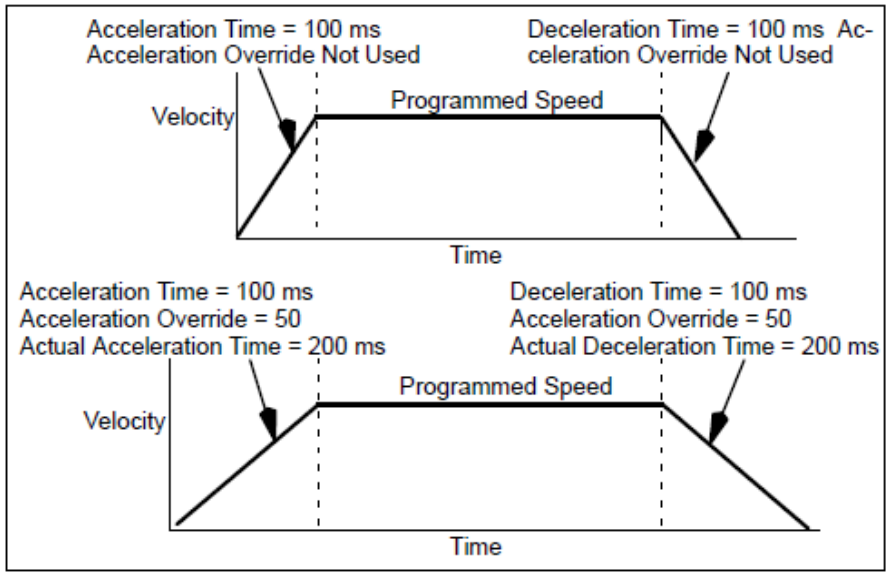


e) ACCURACY, TERMINATION

- advantages:
  - we achieve higher robot movement speeds – we shorten the work cycle
  - reduce or eliminate robot vibrations
  - longer robot lifespan
  - the robot must “dance like a ballerina”…

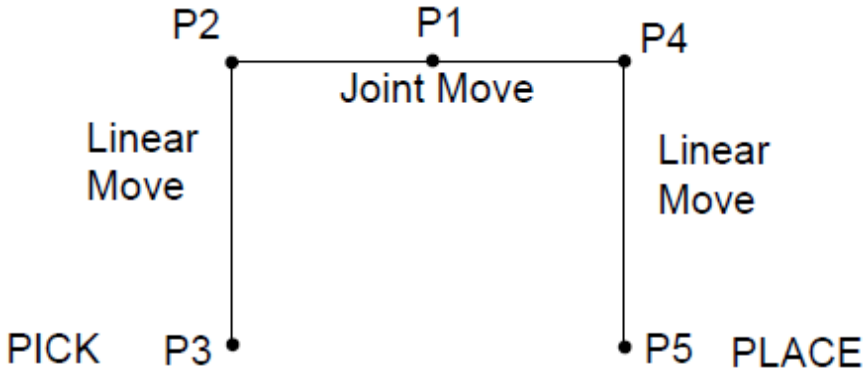
e) ADDITIONAL MOVEMENT PARAMETER: ACC

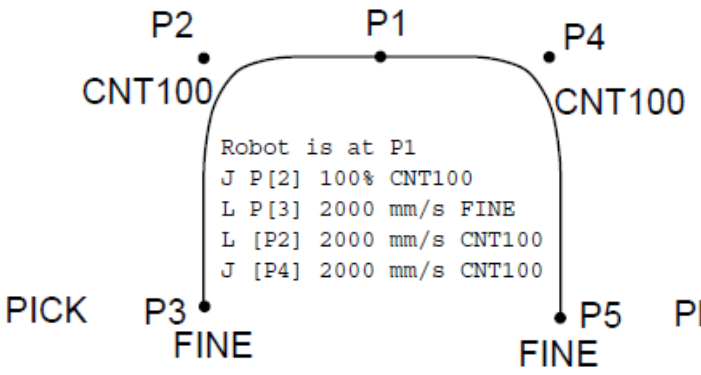
- ACC50
- is the value in % between 20% and 150%
- a value of 50% means that the robot will need 2× more time to accelerate and decelerate
- a value above 100% should be used with caution, as it places greater strain on the robot

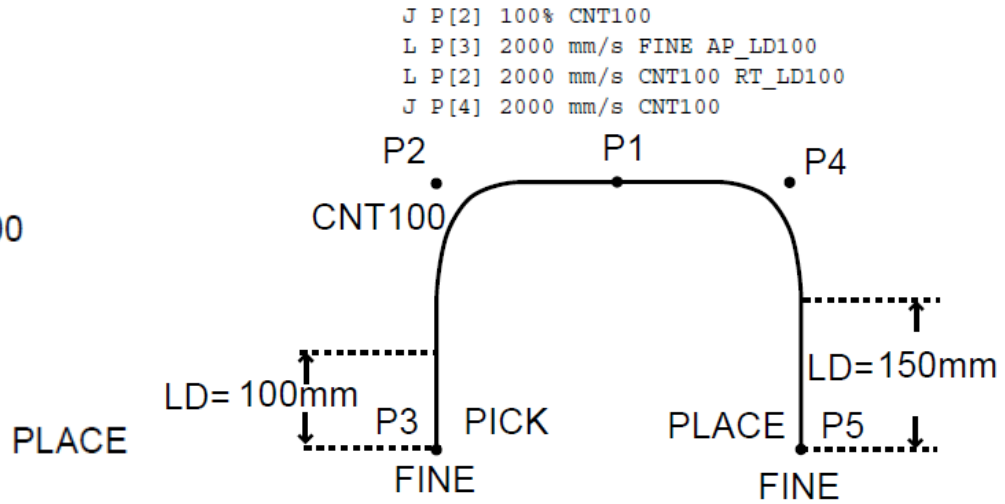


e) ADDITIONAL MOVEMENT PARAMETER: AP_LD, RT_LD

- **AP_LDXX** (AProach_LinearDistance), **RT_LDXX** (ReTrieve_LinearDistance), XX in mm, R[]
- suitable for pick & place applications
- TCP travels from P1 through P2 to P3 (pick up) and back through P2 and P4 to P5 (post), image left
- to achieve a shorter cycle time, use the CNT parameter, middle image
  - problem: we don't know how much linearity we have above P3 or P5
  - solution: we can experiment with different CNTs or raising/lowering P2 and P4
- better: use the LinearDistance parameter , image on the right

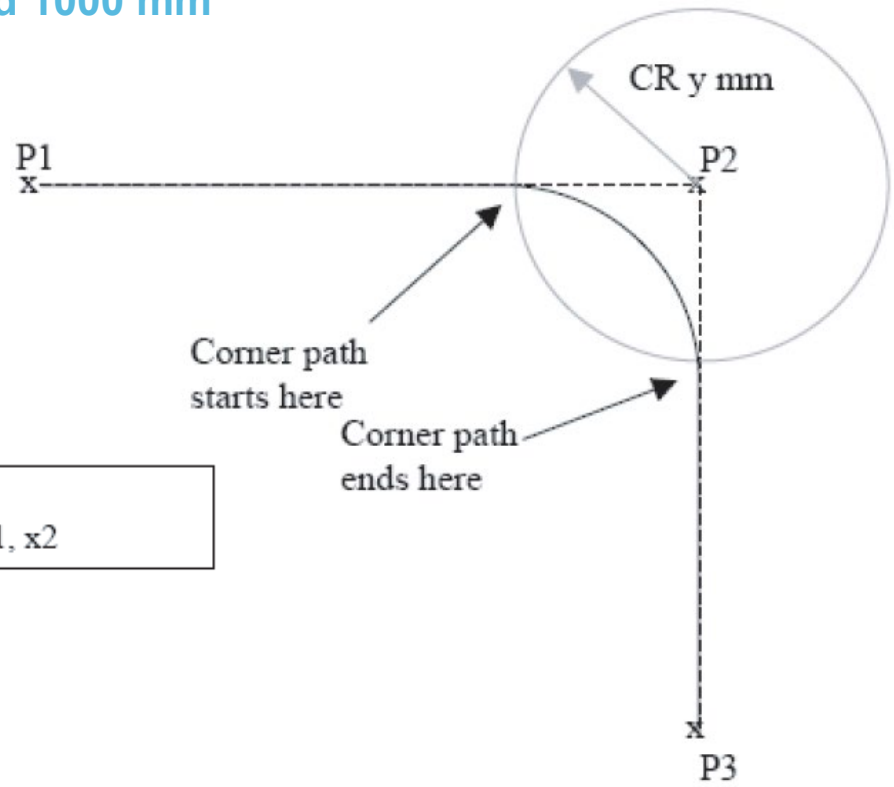




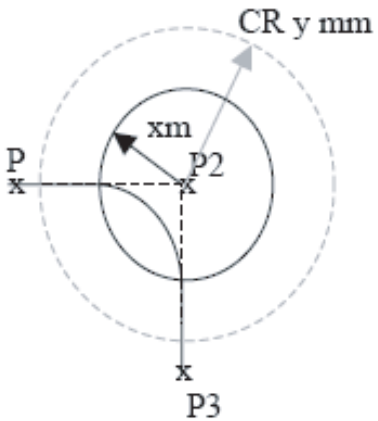


e) ADDITIONAL MOVEMENT PARAMETER: CRy

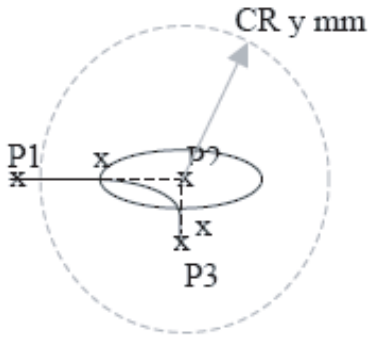
- **CRy** (Corner Region Termination Type), **y given in mm between 0 and 1000 mm**
- is only supported on L and C movements (LP[2] 100 mm/sec CRy)
- is the value how many mm before point P[2] the TCP leaves the path, makes a circular arc and returns to the programmed path again
- y can be at most half of the segment between P [1] and P[2] or between P [2] and P[3]



CR ymm > half segment distance
Actual CR x = half segment distance

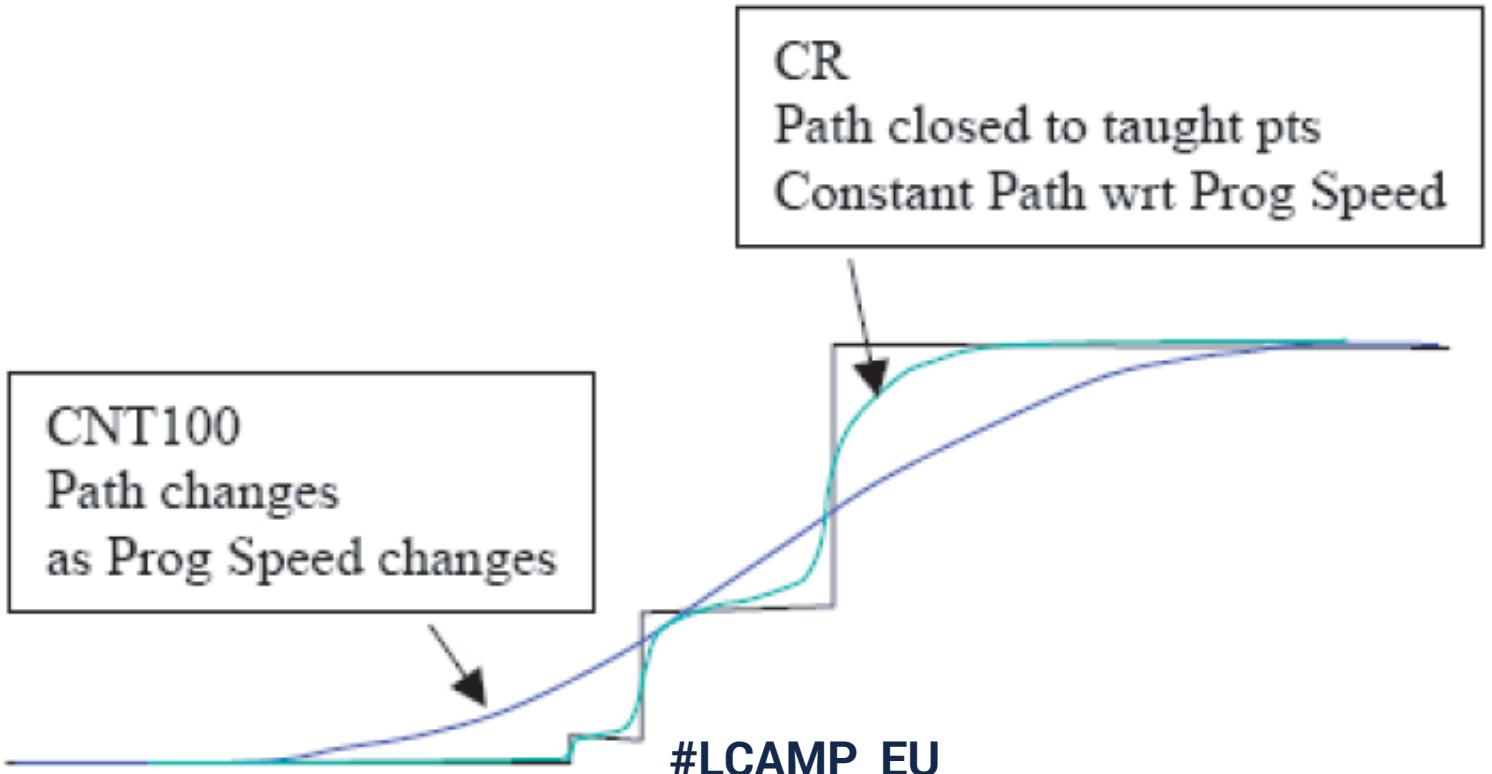


Unequal segment lengths case
Actual CR = half segment distances x1, x2



e) COMPARISON OF MOVEMENT PARAMETERS : CNT and CR

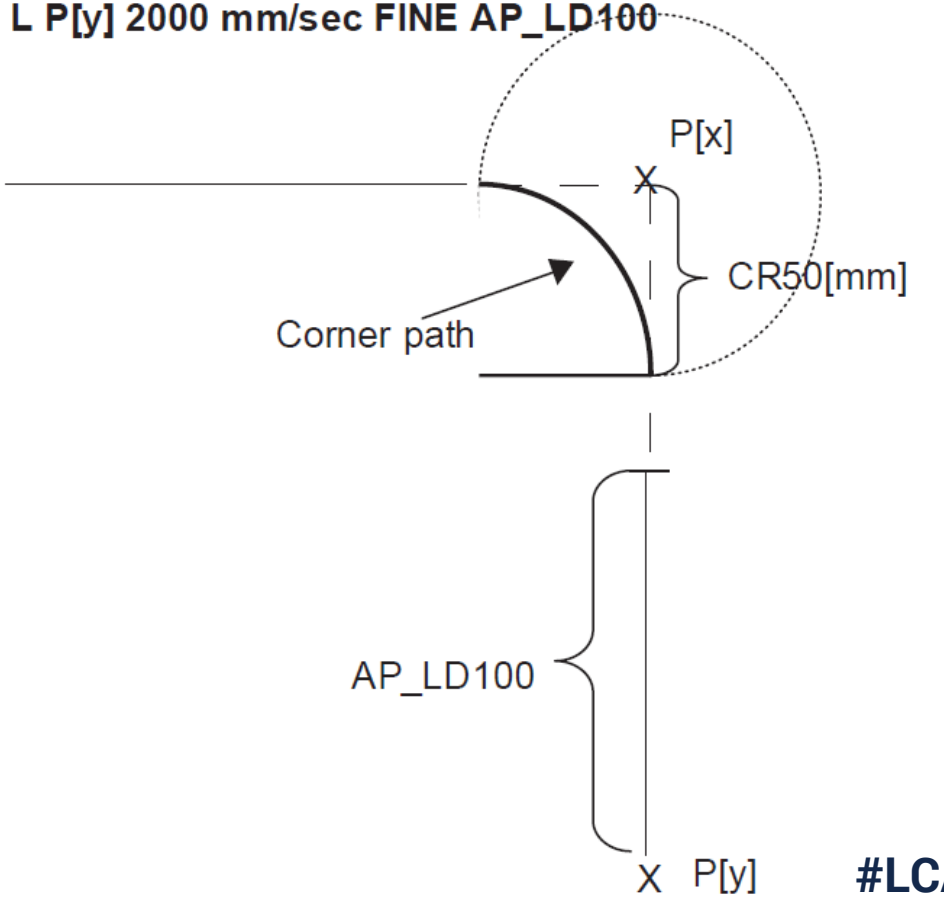
- The distance from the points **changes** at a programmed speed with CNT
- The distance from the points **does not change** with the programmed speed in CR.



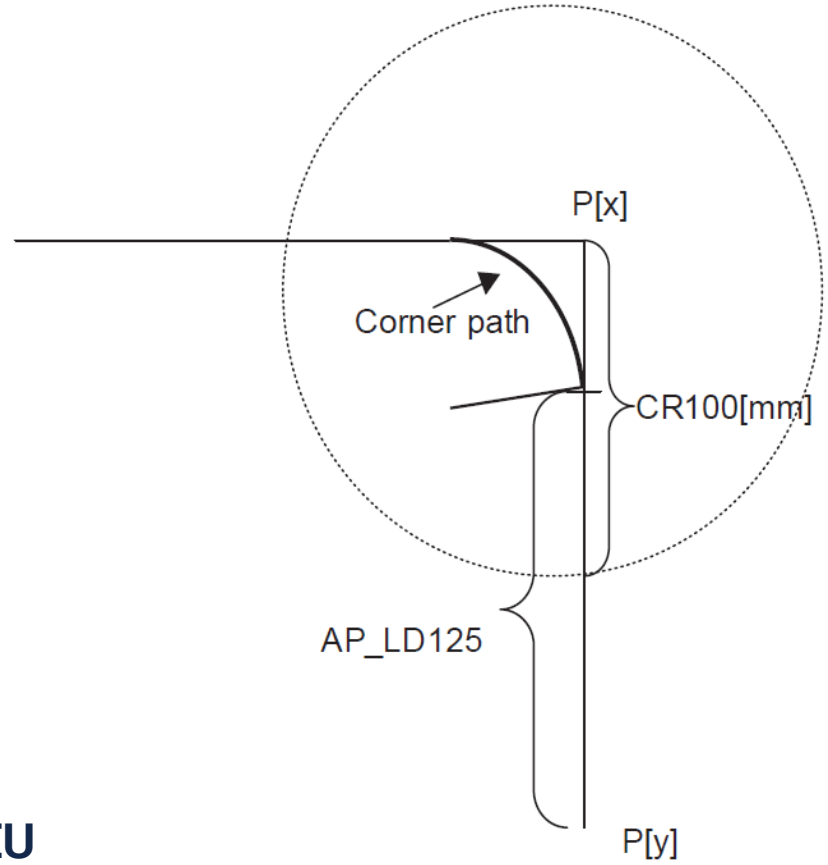
e) USE OF MOVEMENT PARAMETERS: AP_LD and CR

Learner Centric Advanced Manufacturing Platform

L P[x] 2000 mm/sec CR50
L P[y] 2000 mm/sec FINE AP_LD100



L P[x] 2000 mm/sec CR100
L P[y] 2000 mm/sec FINE AP_LD125



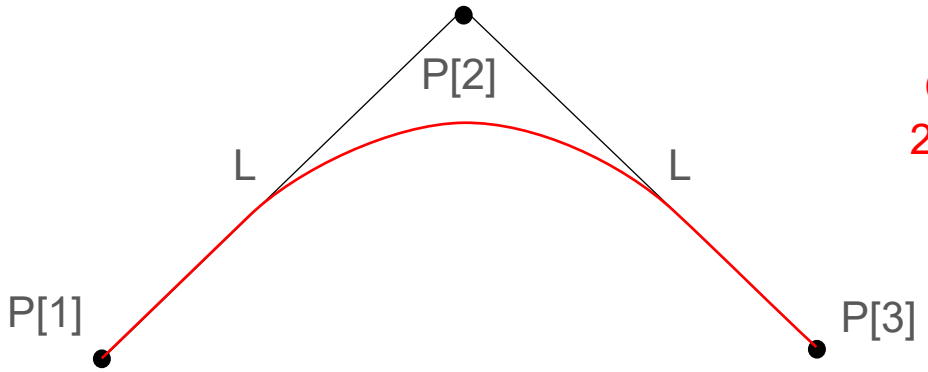
REPEAT

Learner Centric Advanced Manufacturing Platform

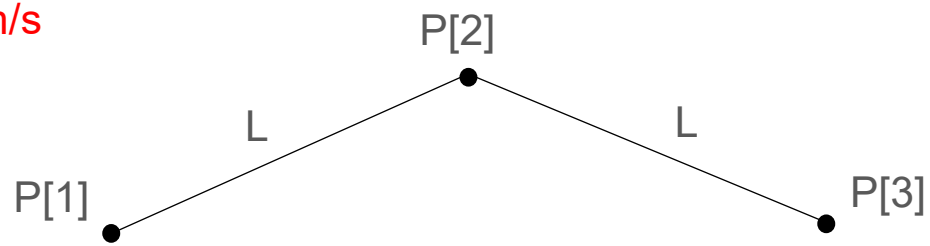
1. What is a move command and how do we create it? How do we change the coordinates in a move command to the current position of the TCP?
2. Describe free movement.
3. Describe linear motion.
4. Describe circular motion.
5. How can a program point be saved? How do we view the coordinates of a point?
6. How can coordinates be stored in point P[ ]? How do we switch between them?

REPEAT

7. How do we name a point? Why do we name points?
8. What is the advantage of storing the point in PR[ ]?
9. Explain how the speed of movement is given and what this speed refers to (what is traveling at this speed)?
10. Describe the precision or termination of a point. What are the advantages of using the CNT parameter?
11. Draw the path of TCP's movement.

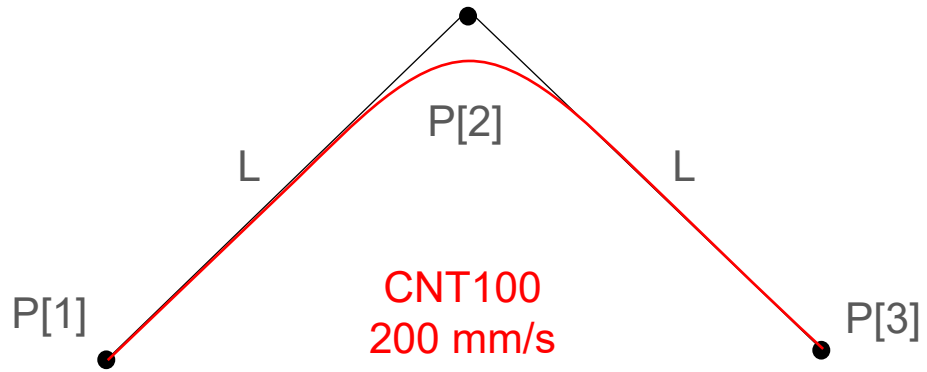


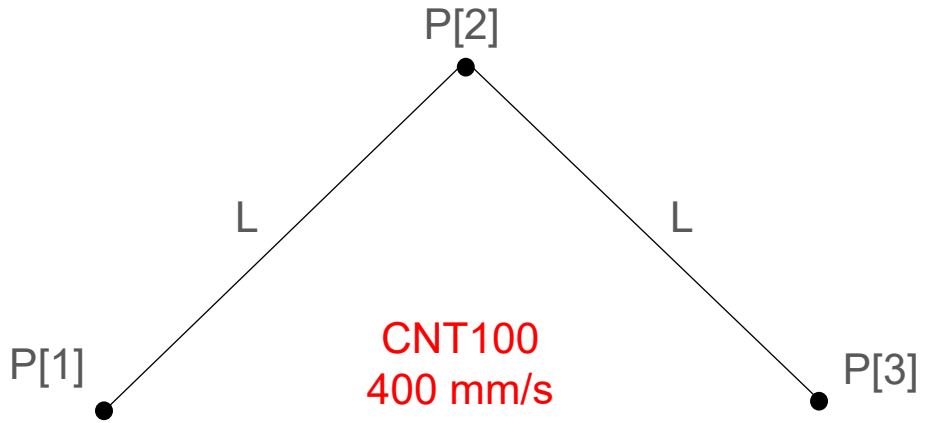
CNT100
200 mm/s



REPEAT

12. Draw the path of TCP's movement.





13. Describe the additional ACC parameter.

DESCRIPTION OF LEARNING / PROGRAMMING AT UE (TEACH PENDANT)

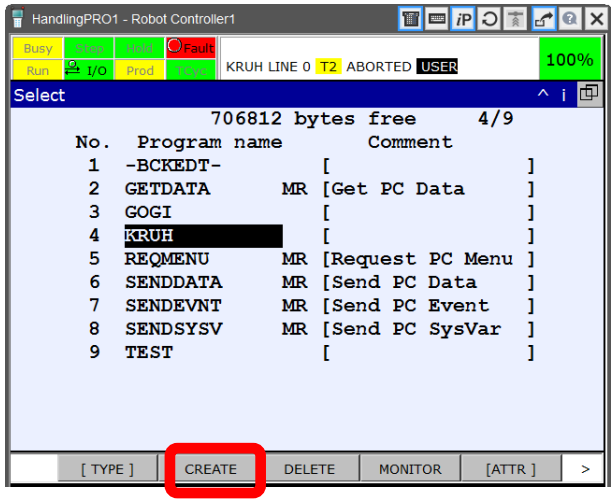
Learner Centric Advanced Manufacturing Platform

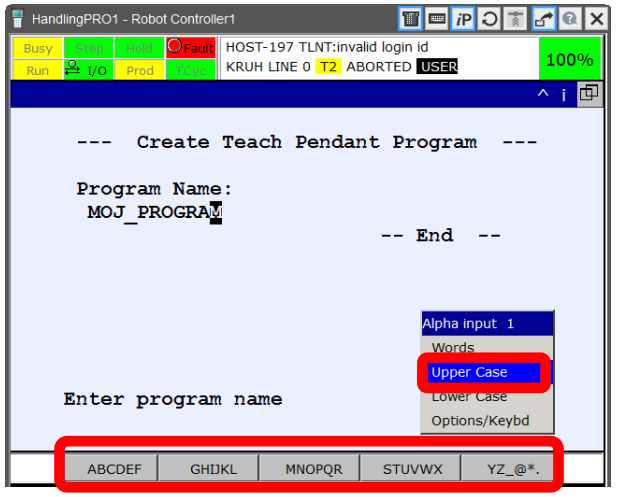
1. Create a new empty robot program.
2. Insert the tool (TF or TCP) with which to program the robot and the space (UF).
3. Set initial values (variables – registers, payload , override …)
4. Move the TCP to the destination point.
5. Insert a movement command.
6. Set the motion parameters (motion mode, point, speed, termination , etc.).
7. At this point, set the necessary activities (close/open the gripper, check the status of the inputs and take appropriate action, activate the outputs, etc.)
8. Repeat steps 4 to 7, and if necessary also 2 (in case of changing tools and/or space).

LEARNING / PROGRAMMING PROCESS ON UE (TEACH PENDANT)

Learner Centric Advanced Manufacturing Platform

1. Set the tool KS (3 or 6 point method) – TOOL FRAME (TF), (if necessary).
2. Set the user frame (3-point method) – USER FRAME (UF), (if necessary).
3. Create a new, empty program with a meaningful name.
   - SELECT / F2 CREATE
   - Select UPPER CASE and use the F1 to F5 keys to write the program name, ENTER
   - Press F3 EDIT to open the program editor.

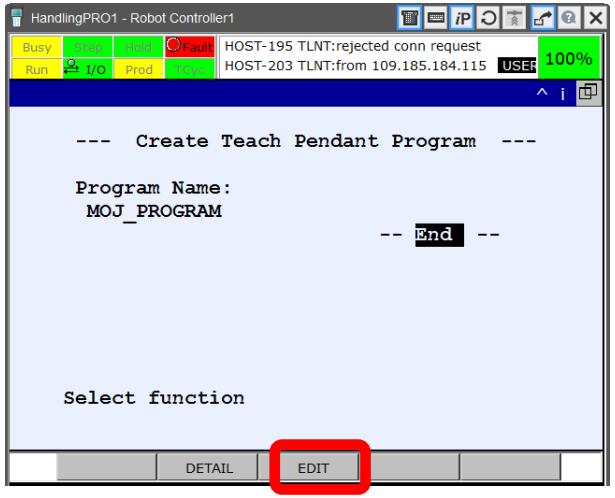


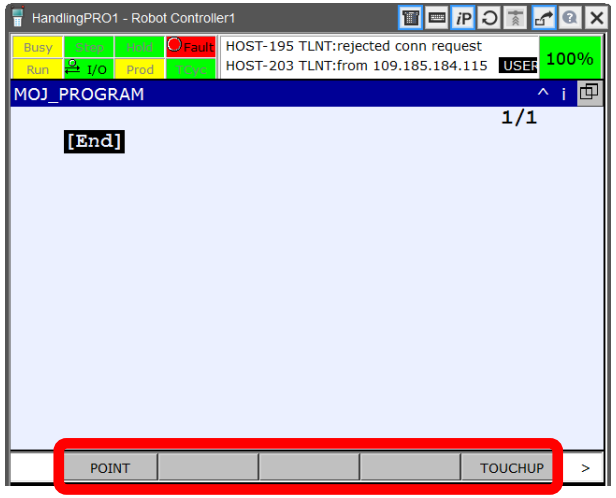


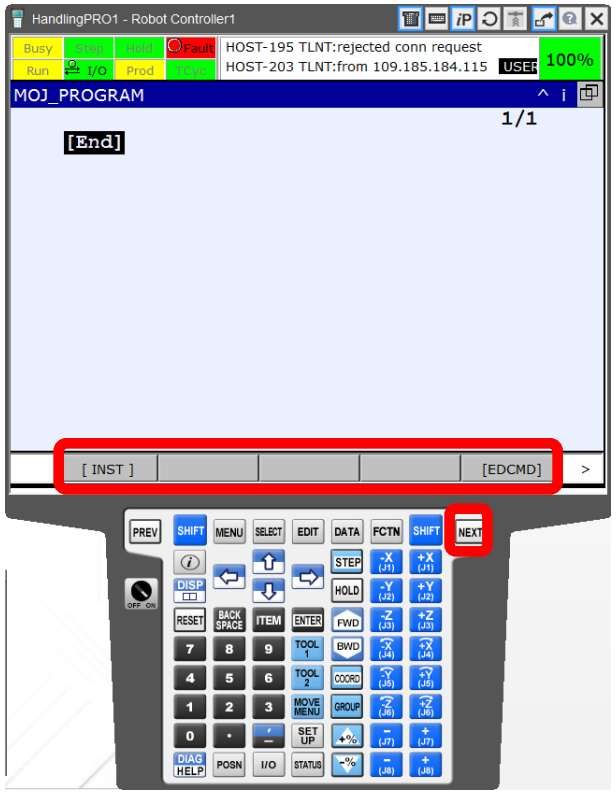
LEARNING / PROGRAMMING PROCESS ON UE (TEACH PENDANT)

Learner Centric Advanced Manufacturing Platform

4. Use TF in the program.
   - NEXT (toggle menu bar) / F1 [INST]



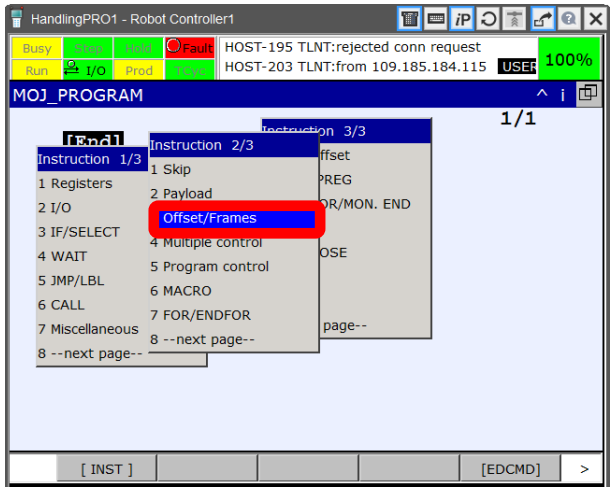


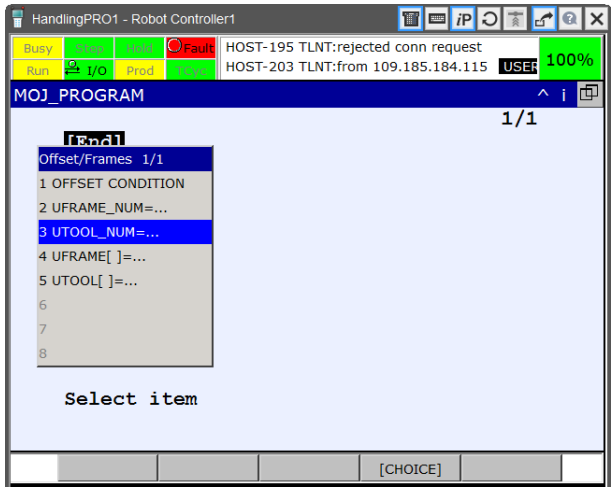


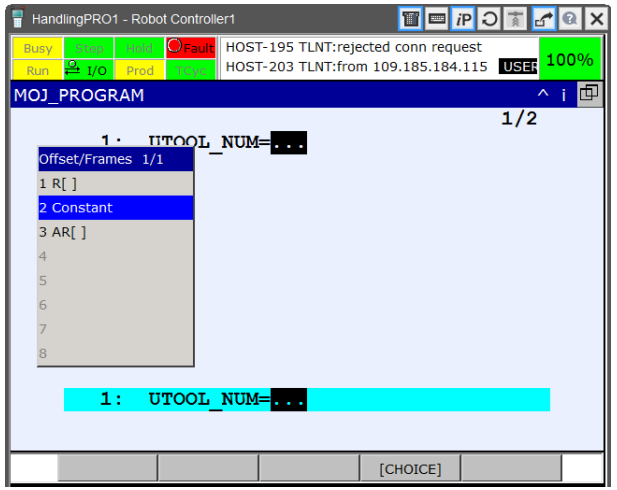
LEARNING / PROGRAMMING PROCESS ON THE UE (TEACH PENDANT)

Learner Centric Advanced Manufacturing Platform

- Offset / Frames + ENTER
- UTOOL_NUM=… + ENTER
- Constant + ENTER
- enter TF number + ENTER



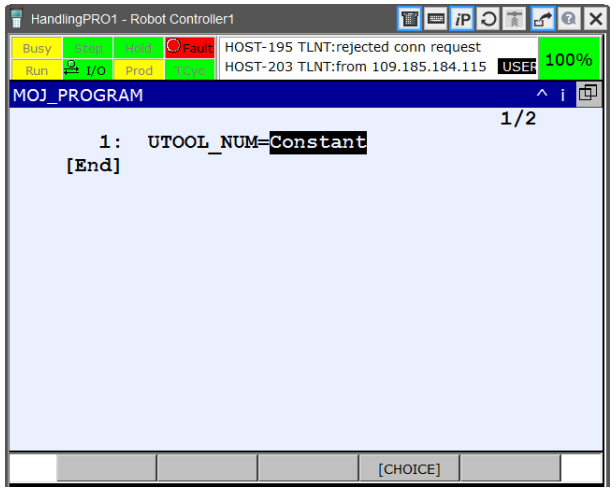


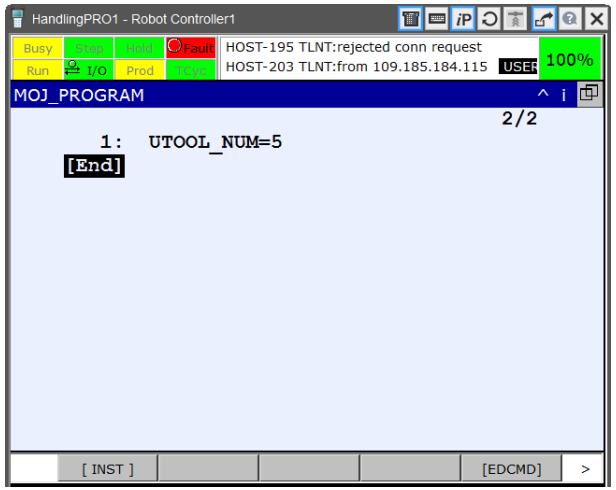


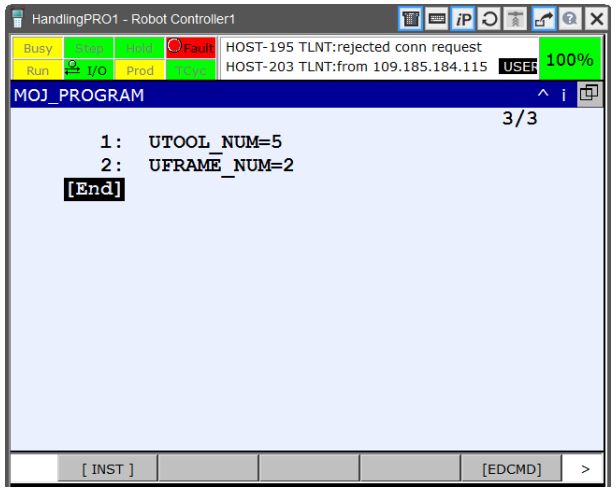
LEARNING / PROGRAMMING PROCESS ON THE UE (TEACH PENDANT)

Learner Centric Advanced Manufacturing Platform

5. Use UF in the program
   - same procedure as in point 4, except that you select UFRAME_NUM=…
6. Specify other settings or initial values in the program (payload , override , register layout, etc.), if necessary.



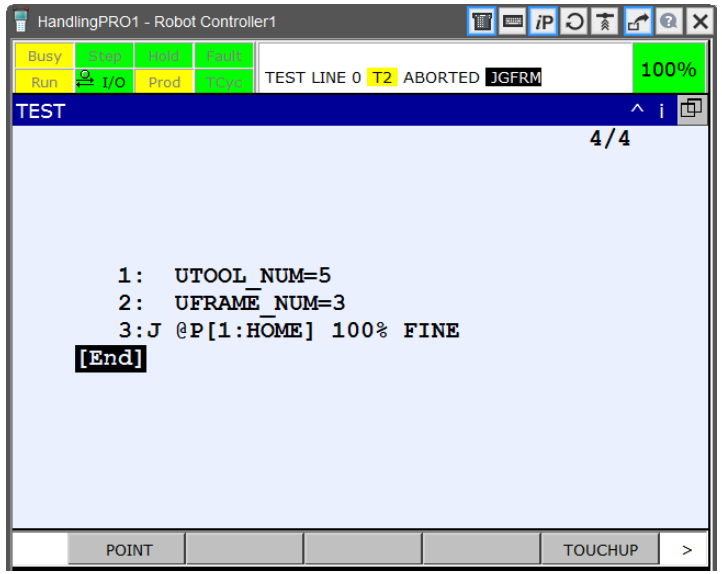


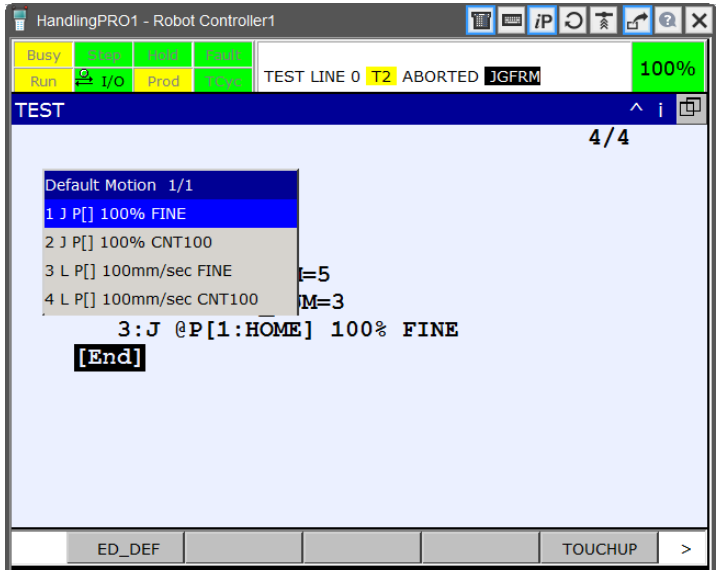


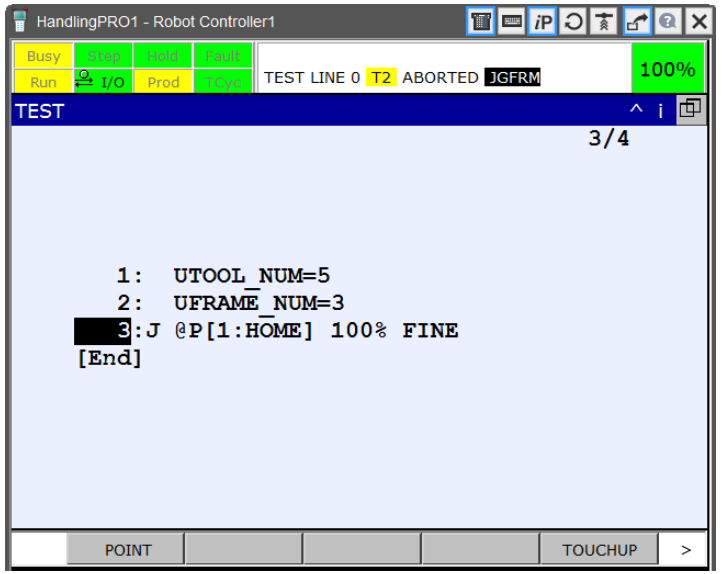
LEARNING / PROGRAMMING PROCESS ON UE (TEACH PENDANT)

Learner Centric Advanced Manufacturing Platform

7. Place the tool (TCP) at the desired location and save it (repeat for all points) and set the necessary parameters:
   - SHIFT + F1 POINT or
   - F1 POINT → select the appropriate movement command + ENTER.
   - SHIFT + F5 TOUCHUP, we correct the coordinates (X, Y, Z, W, P, R, CONF or J1, J2, J3, J4, J5, J6) and UT and UF of an already saved location.



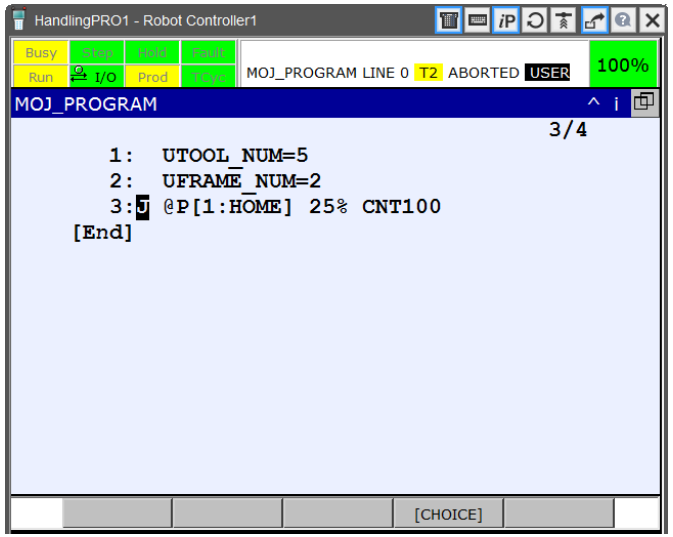


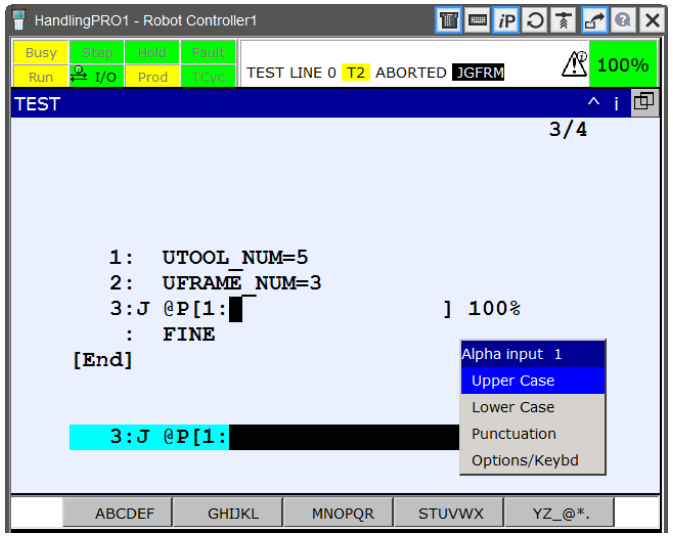


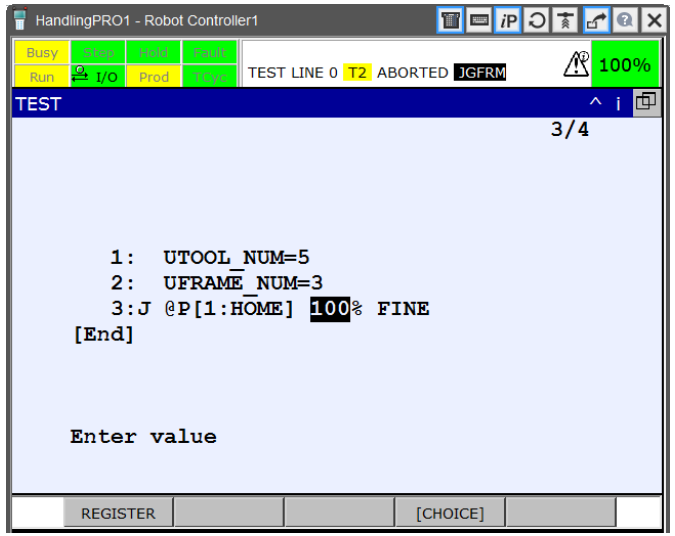
LEARNING / PROGRAMMING PROCESS ON UE (TEACH PENDANT)

Learner Centric Advanced Manufacturing Platform

- set movement mode, F4 [CHOICE] (J, L, C)
- enter the point name, P[1], ENTER, enter the name, ENTER and/or manually correct the coordinates
- enter the speed in % (J) or in mm/s (L and C)
- set the parameter FINE / CNT (F4 [CHOICE]) and/or , ACC, AP_LD, RT_LD, CR ...
- if necessary, use other commands: LBL/JMP LBL , IF / SELECT, FOR /ENDFOR , WAIT, CALL …







LEARNING / PROGRAMMING PROCESS ON UE (TEACH PENDANT)

Learner Centric Advanced Manufacturing Platform

8. Test the program in T1 in STEP mode, slow speed, and debug if necessary.
9. Once all errors have been corrected and the program is working as instructed, test it again in T1 in continuous mode, slow/medium/high speed. If errors are found, correct them and repeat steps 8 and 9.
10. Test the program in T2 in continuous mode, slow/medium/high speed. If you find any errors, fix them and repeat the testing (T1 -> T2, points 8, 9 and 10) until the program works as required.
11. You can run the program in automatic mode.

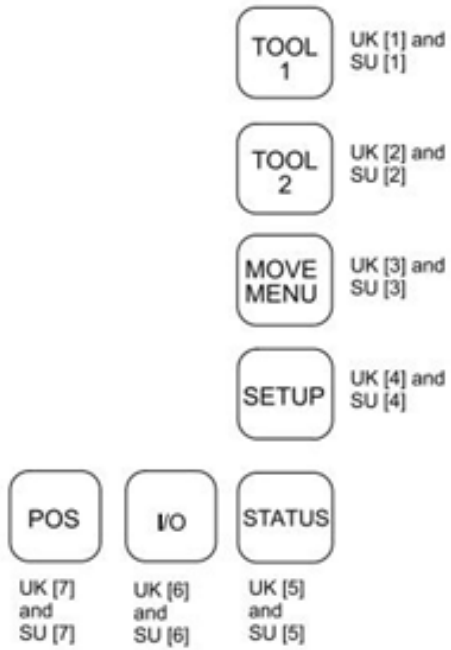
MANUAL RECEIVER ACTIVATION

- SHIFT + TOOL 1 (white key, right of 9)

SHIFT + TOOL 1 – gripper open/close

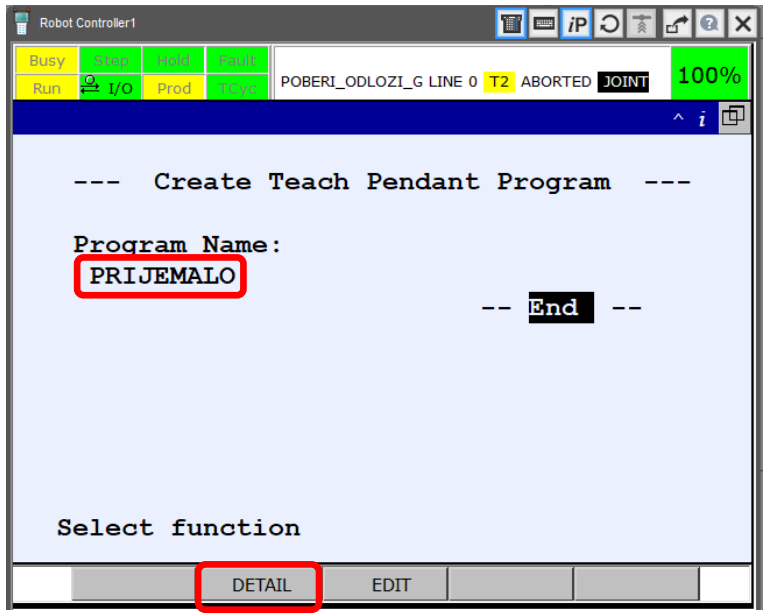
SETTING MANUAL RECEIVER ACTIVATION

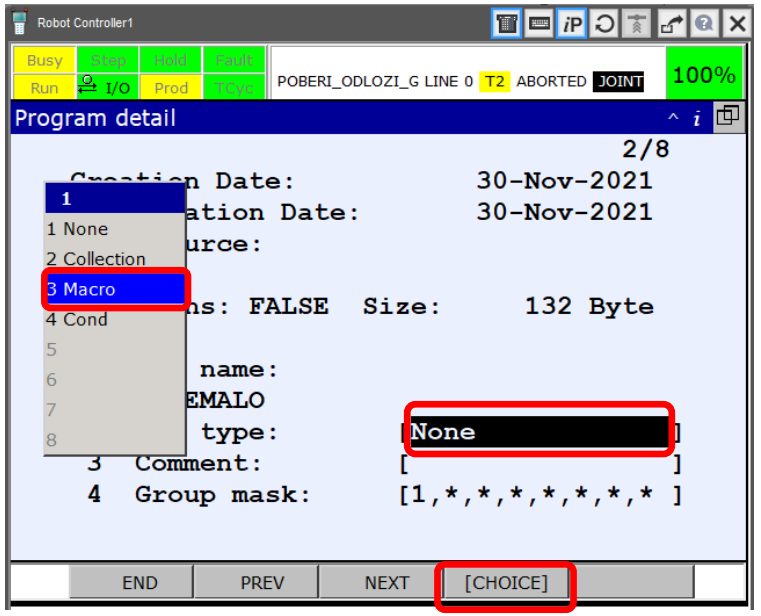
- create a macro
  - A macro is similar to a robot program, except it does not contain movement commands.
- macro can be assigned to user keys (white unlabeled keys)

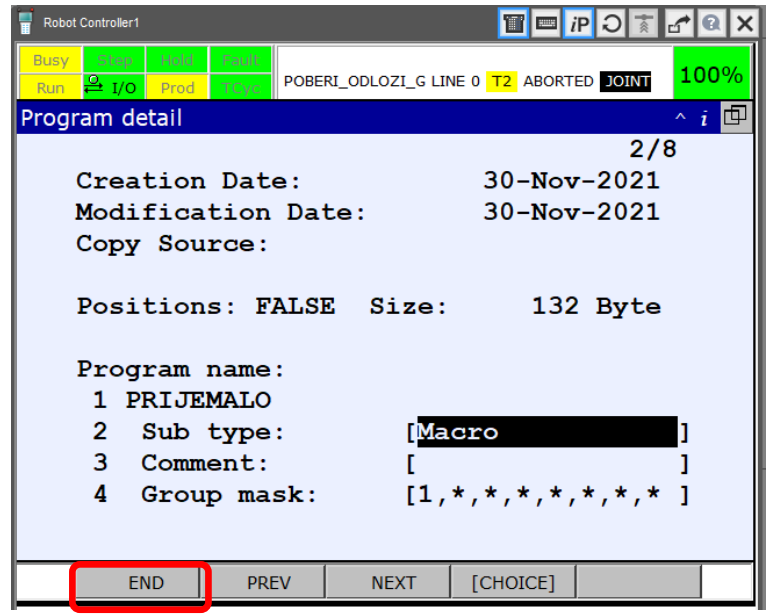


SETTING MANUAL RECEIVER ACTIVATION

- creating a macro (similar to a robot program):
  - on the UE click F2 CREATE
  - enter the macro name and confirm with ENTER
  - click F2 DETAIL and in the Sub type selection : click F4 [CHOICE]
  - select 3 Macro in the menu and confirm with F1 END

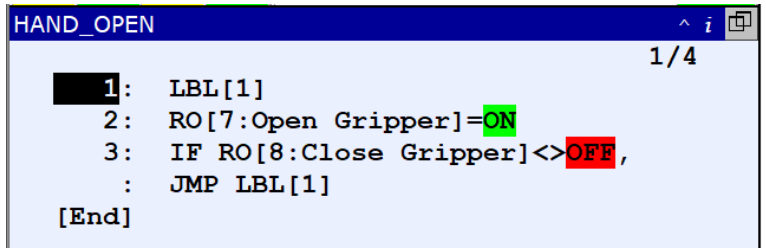


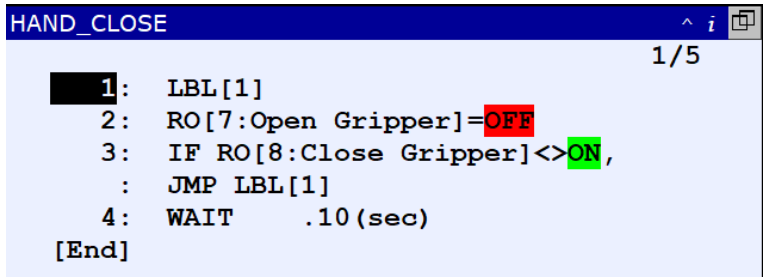


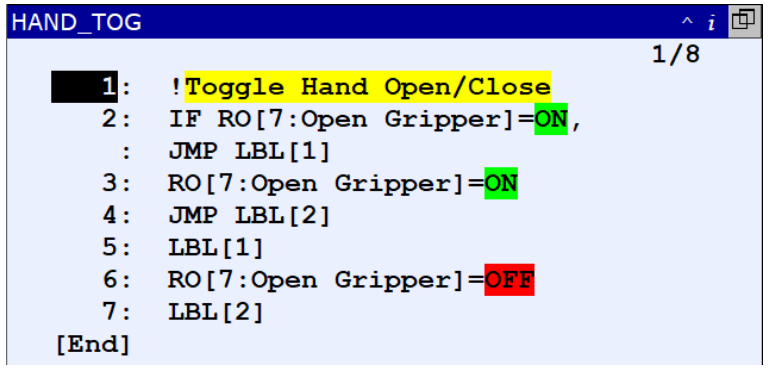


SETTING MANUAL RECEIVER ACTIVATION

- In the program editor, write the appropriate code to activate the gripper
- example:
  - HAND_OPEN – macro for opening the gripper
  - HAND_CLOSE – macro for closing the gripper
  - HAND_TOG – macro for toggling the gripper open/closed

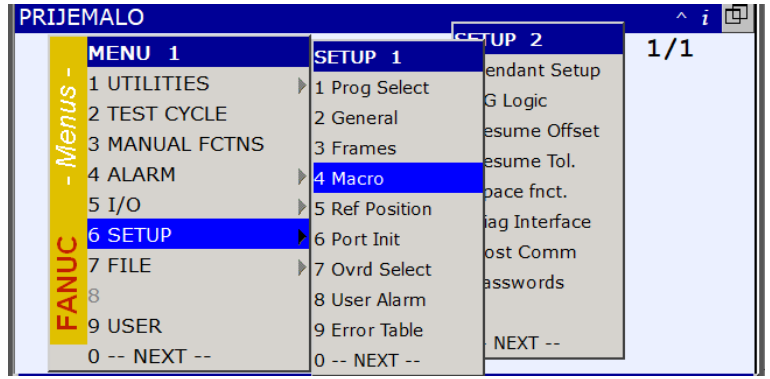


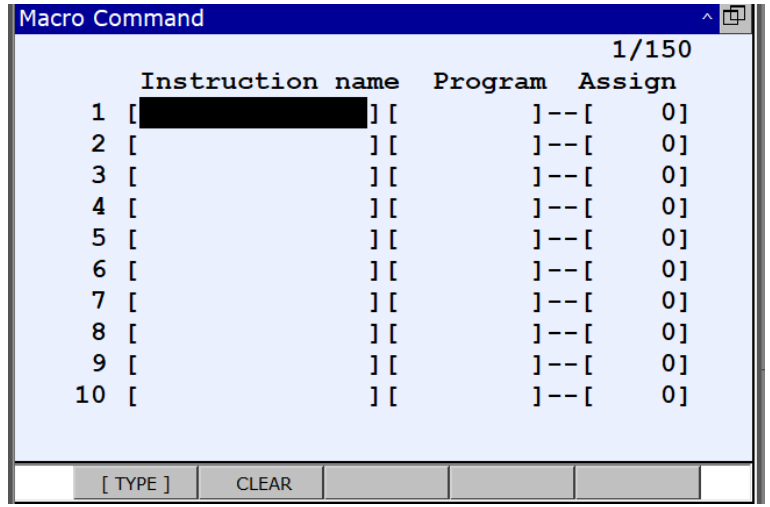


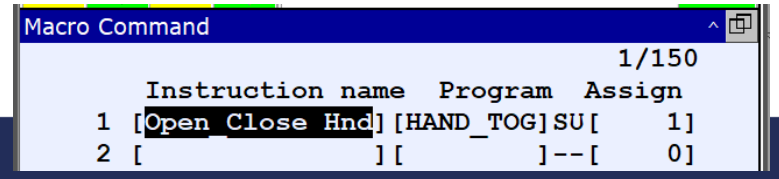


SETTING MANUAL RECEIVER ACTIVATION

- assigning macros to user keys
  - on the UE, click MENU/6 SETUP/4 Macro
  - in the Instruction name field, click ENTER and enter the name of the command and confirm with ENTER
  - in the Program field, click F4 CHOICE and find the previously created macro
  - in the - - field, click F4 CHOICE and select 2 UK or 3 SU
    - UK – User Key
    - SU – SHIFT + User Key
  - in the Assign field , enter the user key number (UK or SU)
- Test the activation of the gripper by clicking on UK or SU

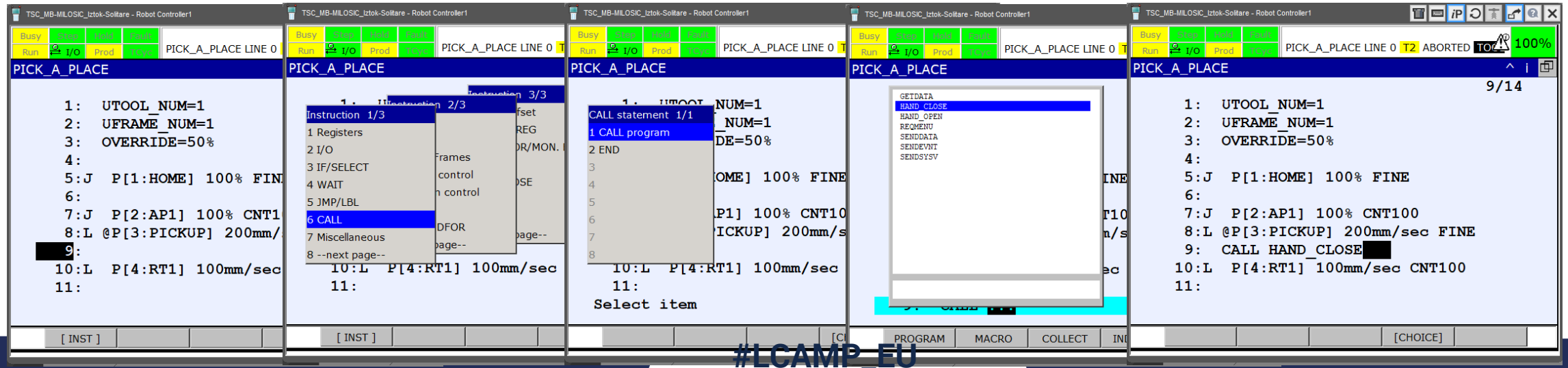






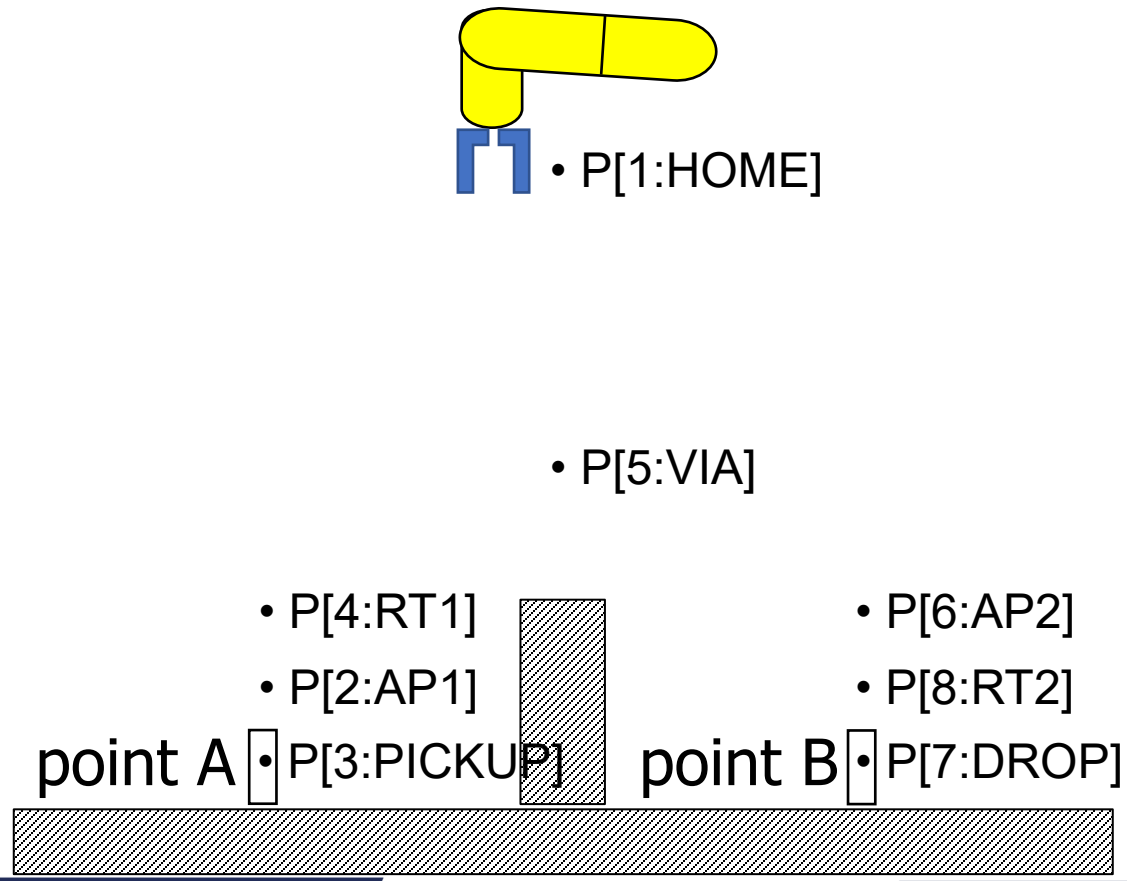
SOFTWARE ACTIVATION OF THE RECEIVER

- In the program, we activate the gripper by calling the subroutine in the corresponding program line.
  - click F1 [INST] and select 6 CALL/1 CALL program
  - click F2 MACRO and find the appropriate macro and confirm with ENTER



PROGRAMMING GUIDELINES

- We plan TCP points or locations sensibly :



PROGRAMMING GUIDELINES

Learner Centric Advanced Manufacturing Platform

- **We plan TCP points or locations sensibly** :
  - **starting point** (HOME) **P[1]**
    - at this point TCP starts and eventually returns to it or waits for the next command
    - is suitably close and at the same time far enough away that the robot does not interfere with other activities
  - **target (end) points** (PICKUP, DROP, WELD …) **P[3] and P[6]**
    - operating points (picking/placing, welding, gluing, etc.)
  - **prefixes** (APPROACH) **P[2] and return points (** RETREAT) **P[5]**
    - are perpendicular to the target points and
    - in them the tool is properly oriented
  - **intermediate points** (VIA) **P[4]**
    - to avoid obstacles
    - for more appropriate TCP movement
  - the points **– we give them meaningful names**

PROGRAMMING GUIDELINES

- at **the beginning of the program always** :
  - set KS (TF and UF),
  - we set up payloads – PAYLOAD
  - set speed, OVERRIDE
  - we set initial values, declare variables (registers)...
- For **better program clarity, we leave blank lines** between individual program parts.
- **We comment on the program meaningfully.**
- **longer programs are divided into** smaller, meaningful **subprograms**

PROGRAMMING GUIDELINES

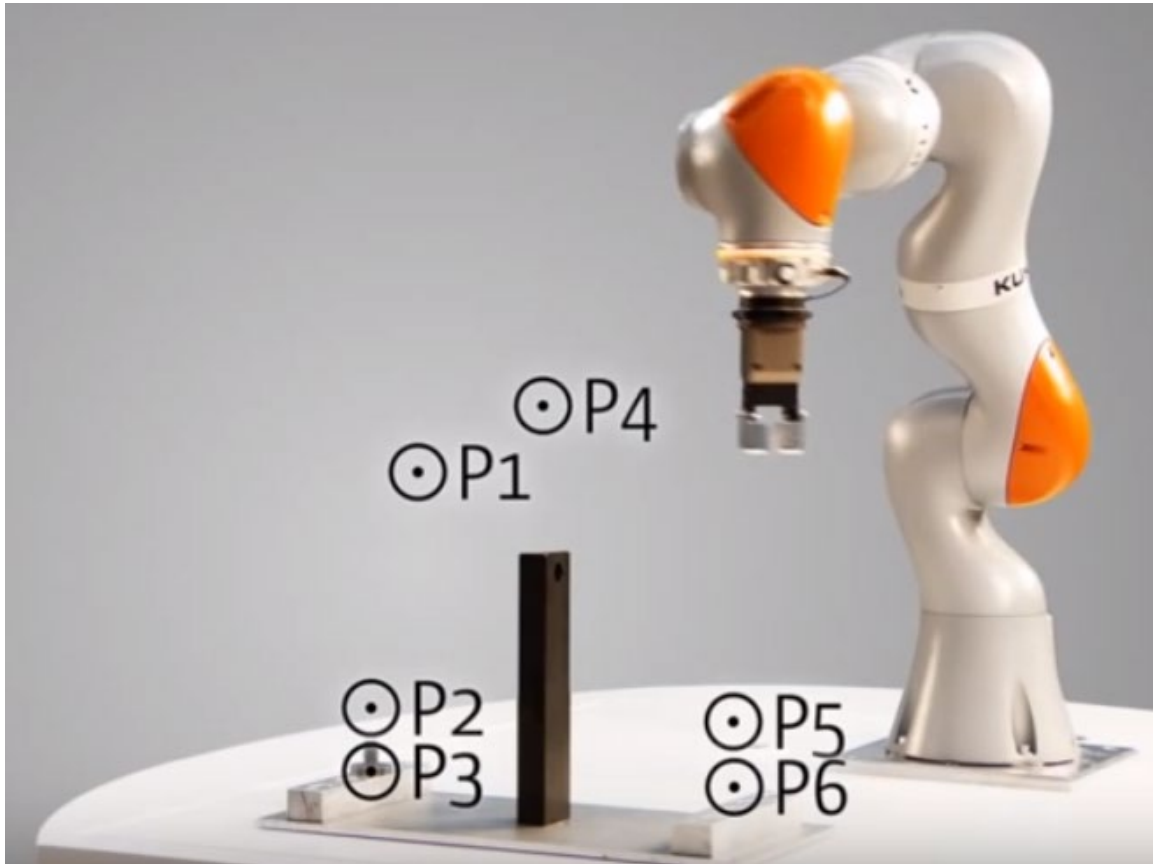
- we use **free motion (J)** – axis movement, for faster operation and **shorter cycle times**
- linear and circular movements (L, C) are slower; they are used where the TCP must travel along a precisely planned path – movement along a path (picking/placing, welding, gluing, etc.)
- the pre-points (AP) and return points (RT) as close as possible to the target points (PICKUP, DROP, WELD, etc.)
- The FINE parameter is used only in target points (PICKUP, DROP, WELD, etc.), in intermediate points (VIA) we use the CNT parameter

CYCLE TIME OPTIMIZATION

- appropriate movements
  - use as many free (J) moves as possible, except where you must use L or C moves (TCP movement along the path)
- moving points
  - the HOME starting position should be as close as possible to the workspace, but far enough away so as not to interfere with other activities
  - Perform slow movements (L and C) at the shortest possible distances - move points AP, RT as close as possible to picking, placing, gluing, welding, etc.
- speed of movement
  - use the highest possible movement speeds that the technological process allows
- FINE/CNT setting
  - use as many and as high a value for the CNT parameter as possible, except where the TCP must definitely stop FINE (picking, depositing, start/end of gluing, welding, etc.)
- ACC setting
  - increase/decrease the acceleration and deceleration of the robot movement accordingly

LEARNING / PROGRAMMING

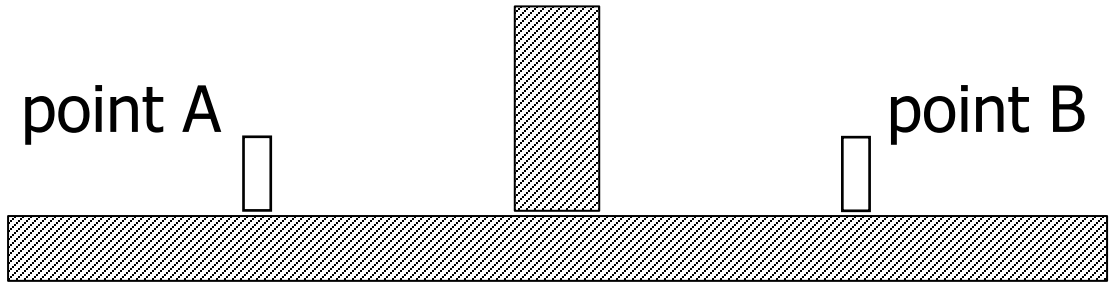
- example of learning (programming) a collaborative robot



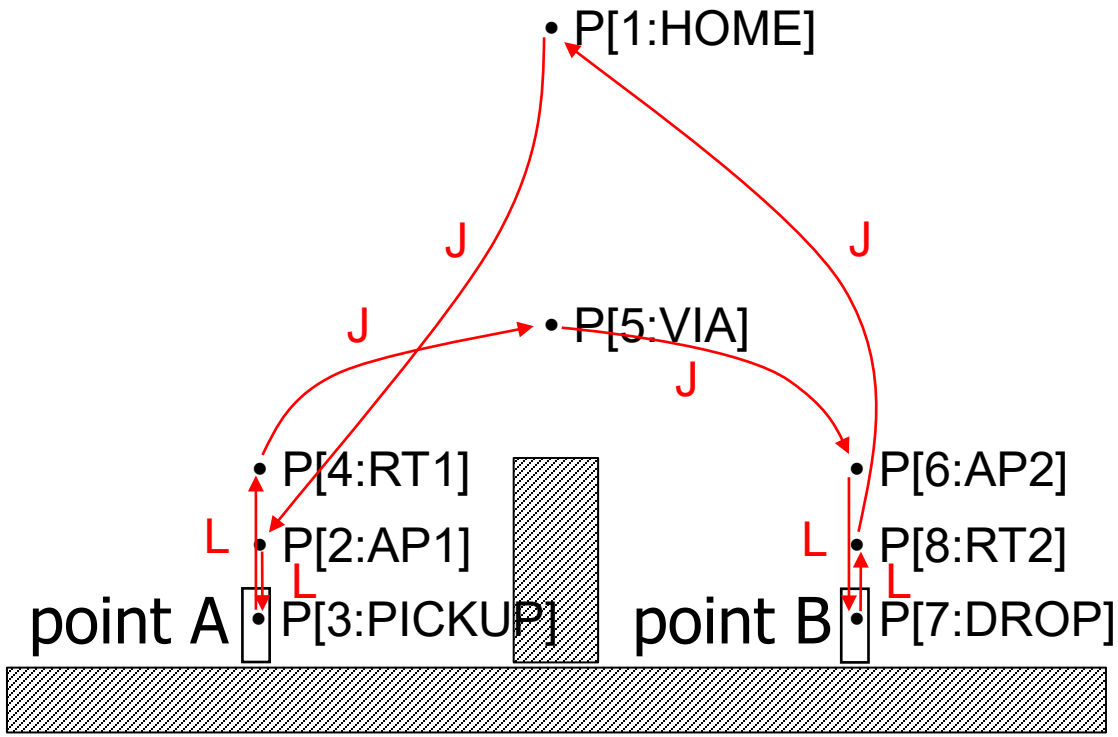
https://www.youtube.com/watch?v=r7gU74Yv9Es

EXAMPLE

- Write a program that will move an object from point A over an obstacle to point B, as shown in the figure. Follow the programming guidelines,
  - use TF 1 and UF 1 and general speed (OVERRIDE) 50%
  - draw and meaningfully name the points,
  - draw the paths of movement and label and explain them,
  - when terminating CNT use the value 50,
  - use 50% speed for J movements
  - for L movements 200 mm/s if the gripper is empty and 100 mm/s if the gripper is full



EXAMPLE

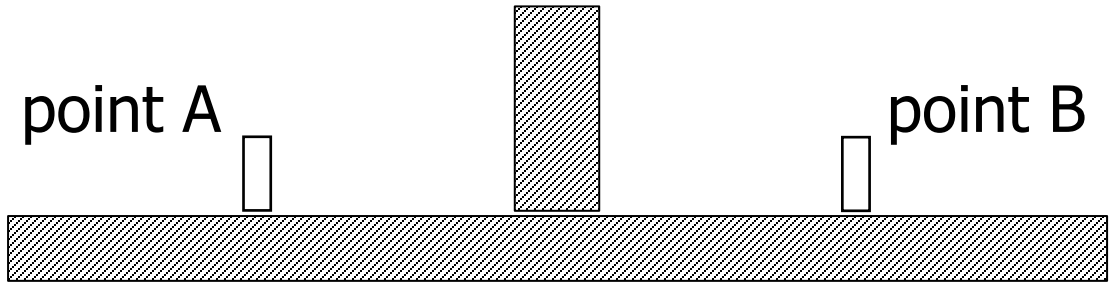

P[1:HOME]
J
J
J
P[5:VIA]
J
P[4:RT1]
P[6:AP2]
L
P[2:AP1]
L
P[8:RT2]
L
L
point A
P[3:PICKUP]
point B
P[7:DROP]

REPEAT

1. Describe a simplified procedure for programming a robot on a learning unit.
2. Explain the guidelines when programming a robot?
3. How do we optimize program cycle time?
4. How to activate the gripper:
   - manually,
   - in the program?
5. What is the setup procedure for manually activating the gripper?

REPEAT

6. Write a program that will move an object from point A over an obstacle to point B, as shown in the figure.
   - follow the programming guidelines
   - use TF 1 and UF 2
   - draw and name the points
   - Draw the paths of movement and explain the movements
   - use 50% speed for J movements
   - for L movements 200 mm/s if the gripper is empty and 100 mm/s if the gripper is full



LCAMP

Learner Centric Advanced Manufacturing Platform

#LCAMP_EU

info@lcamp.eu

www.lcamp.eu

@LCAMP_EU

LCAMP
Learner Centric Advanced
Manufacturing Platform for CoVEs

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.