LCAMP

Learner Centric Advanced Manufacturing Platform

PROGRAMIRANJE FANUC INDUSTRIJSKIH ROBOTOV

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.

ROBOTSKI PROGRAM

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**

PROGRAMIRANJE FANUC INDUSTRIJSKIH ROBOTOV

Learner Centric Advanced Manufacturing Platform

UČENJE/PROGRAMIRANJE ROBOTA

- robotski program
- izbira/odpiranje programa
- parametri programa
- testiranje (T1/T2) programa
- zagon (AUTO) programa
- robotski program
- izdelava robotskega programa
- urejanje robotskega programa
- arhiviranje in restavriranje robotskih programov

PROGRAMIRANJE FANUC INDUSTRIJSKIH ROBOTOV

Learner Centric Advanced Manufacturing Platform

FAZE PROGRAMIRANJA

Programiranja se lotimo podobno kot razvoj drugih izdelkov

- razvoj izdelka

- učenje / programiranje industrijskega robota

PROGRAMIRANJE FANUC INDUSTRIJSKIH ROBOTOV

Learner Centric Advanced Manufacturing Platform

FAZE PROGRAMIRANJA

Projektna naloga

- je tekstovni opis, z grafičnimi prilogami (skice, fotografije …), avtomatiziranega procesa, da programer do podrobnosti razume avtomatizirani proces
- predvidoma jo napiše naročnik

Načrtovanje

- je koračni opis postopka (tekstovno in/ali grafično), ki korak za korakom reši dani problem
- tekstovni opis
- diagram poteka (flow chart)

Kodiranje in ko/dokumentiranje

- diagram poteka zapišete v zaporedje ukazov
- komentarji in opisi za poznejše razumevanje programa in navodila končnim uporabnikom

FAZE PROGRAMIRANJA

Testiranje in odpravljanje napak

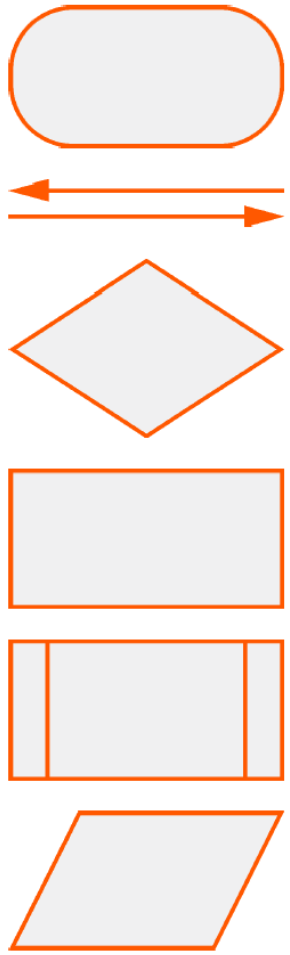
- testiranje je preverjanje delovanja programa pred avtomatskim izvajanjem programa
- je zelo pomembno in mora biti izvedeno, da zagotovimo varnost ljudi in preprečimo poškodbe opreme

Izdelek

- delujoči (komentirani) program za avtomatizirani proces vključno z dokumentacijo

NAČRTOVANJE PROGRAMIRANJA

Diagram poteka



Beginn und Ende eines Prozesses oder Programms

Verknüpfung von Anweisungen und Operationen

Verzweigung

Allgemeine Anweisungen im Programmcode

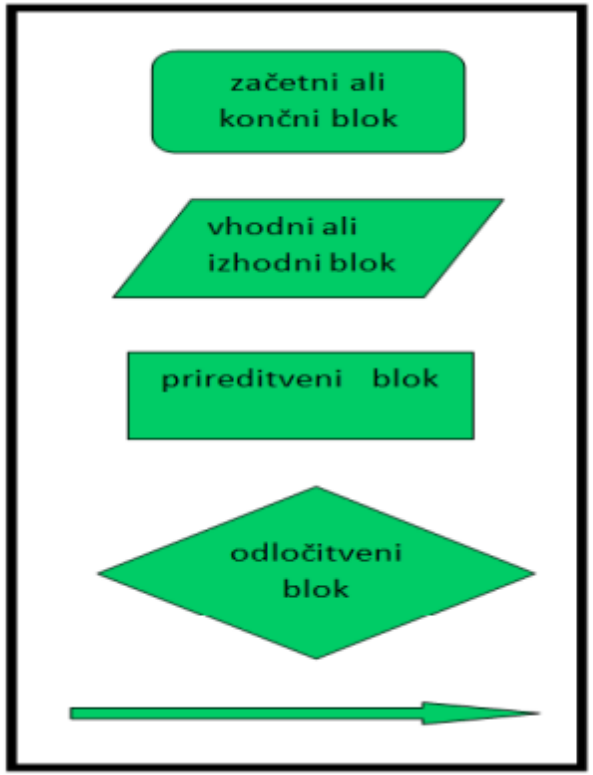
Unterprogrammaufruf

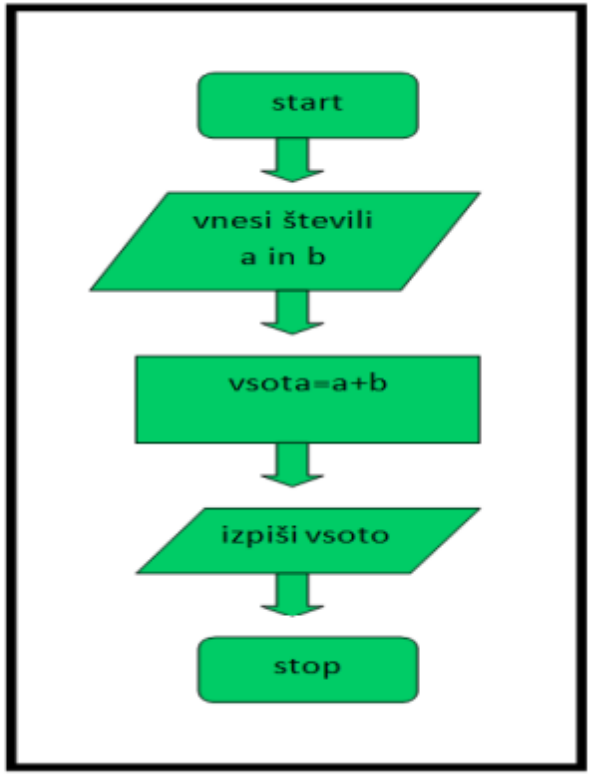
Ein-/Ausgabe Anweisung

PROGRAMIRANJE FANUC INDUSTRIJSKIH ROBOTOV

NAČRTOVANJE PROGRAMIRANJA

- Diagram poteka (Flow Chart)
- Je grafično prikazan algoritem, torej na človeku razumljiv način opisan potek dogodkov.
- Uporablja se pri načrtovanju programov in tudi pri opisovanju ostalih dejavnosti (npr. Domači zdravnik v knjižni obliki, Občinska navodila za pridobitev dovoljenj itd).
- Pametno se je lotiti programskega problema na način diagrama poteka, da si razjasnite izvajanje po korakih in predvidite, kaj bo moral narediti računalnik.



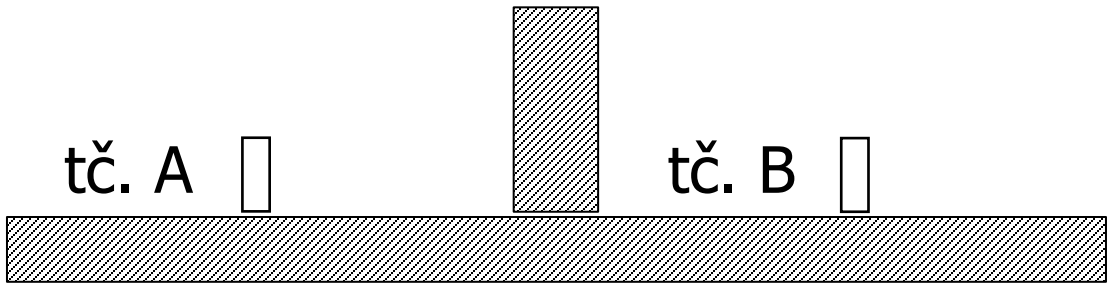


NAČRTOVANJE PROGRAMIRANJA

- Programer mora do podrobnosti razumeti problem, za katerega piše program. Njegovo delo lahko razčlenimo v naslednja opravila:
- Analiza problema je razvoj postopka, ki korak za korakom reši dani problem. Temu postopku pravimo algoritem.
- Organizacija programa. Algoritem organizirate v zaporedje operacij, pri čemer uporabljate diagrame poteka. Delo lahko opravljate s pomočjo osebnega računalnika, če imate orodje za risanje diagramov poteka.
- Kodiranje. Diagram poteka zapišete v zaporedje ukazov – program s pomočjo urejevalnika teksta za osebni računalnik. Shranite ga v t. i. izvorni datoteki.
- Povezovanje in prevajanje. Program povežete z drugimi deli programa, ki ste jih napisali že v preteklosti, ali pa s programi, ki jih dobite v knjižnici (napisal jih je proizvajalec prevajalnika in povezovalnika). Povezavo opravi povezovalnik – linker. Celoten program prevedete s prevajalnikom – compilerjem v obliko, ki jo razume uP. Povezovalnik in prevajalnik sta programski opremi za osebni računalnik.
- Vstavljanje programa v mikroračunalnik opravite s pomočjo programatorja, ki ga priključite na osebni računalnik s pripadajočo programsko opremo.
- Preizkušanje. Program preizkusite tako, da z izvajanjem na mikroprocesorju ugotovite ali da ustrezne rezultate. V ta namen izdelate testno kartico. Napake, ki jih pri tem odkrijete, odpravite tako, da se vrnete na kodiranje in ponovite postopek.
- Dokumentiranje. Za kasnejšo dodelavo ali spremembo programa je potrebno program dokumentirati. Dokumentacijo s potrebnimi komentarji uredite pri vseh opravilnih točkah (tehnični opis, poročilo, elaborat).

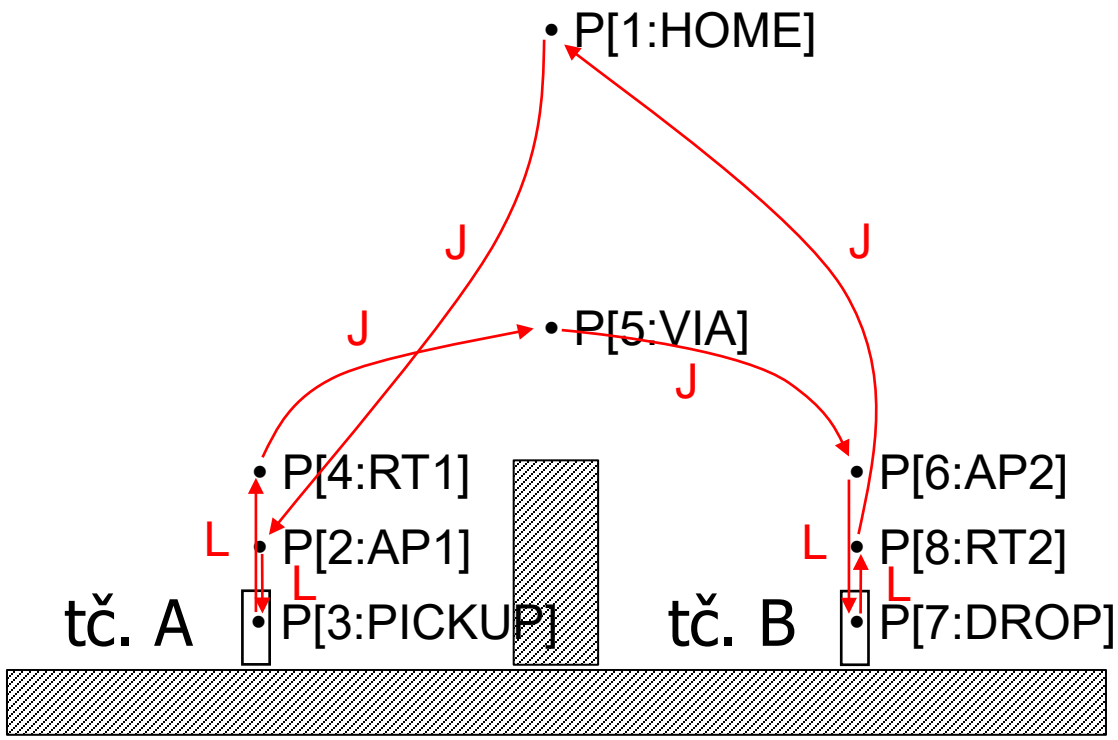
PRIMER FAZ PROGRAMIRANJA

1. Projektna naloga
   - Izdelajte robotsko aplikacijo, ki bo pobrala izdelek in ga odložila na drugi strani ovire.



PRIMER FAZ PROGRAMIRANJA

2. Načrtovanje (tekstovni/grafični opis in/ali diagram poteka)
   - grafični opis



PRIMER FAZ PROGRAMIRANJA

2. Načrtovanje (tekstovni/grafični opis in/ali diagram poteka)
   - tekstovni opis
     - za orodje uporabi prijemalo,
     - uporabi osnovni delovni prostor
     - uporabi splošno polovično hitrost
     - TCP s hitrim gibom natančno postavi v začetno točko
     - TCP s hitrim gibom potuje proti/mimo točk-i/e približevanja
     - z linearnim gibom in primerno hitrostjo TCP natančno postavi v točko pobiranja
     - zapri prijemalo
     - TCP z linearnim gibom potuje proti/mimo točk-i/e vračanja
     - s hitrim gibom prestavi izdelek preko ovire na drugo stran
     - …

PRIMER FAZ PROGRAMIRANJA

2. Načrtovanje (tekstovni/grafični opis in/ali diagram poteka)
   - diagram poteka

ROBOTSKI PROGRAM

- **učenje/programiranje** robota **pomeni**,
  - da **robota ročno vodimo** v ciljno točko in položaj oz. koordinate TCP **shranimo**; v tej točki **tudi nastavimo, kaj naj robot počne** …
  - robota **ročno vodimo v naslednjo točko** …

ROBOTSKI PROGRAM

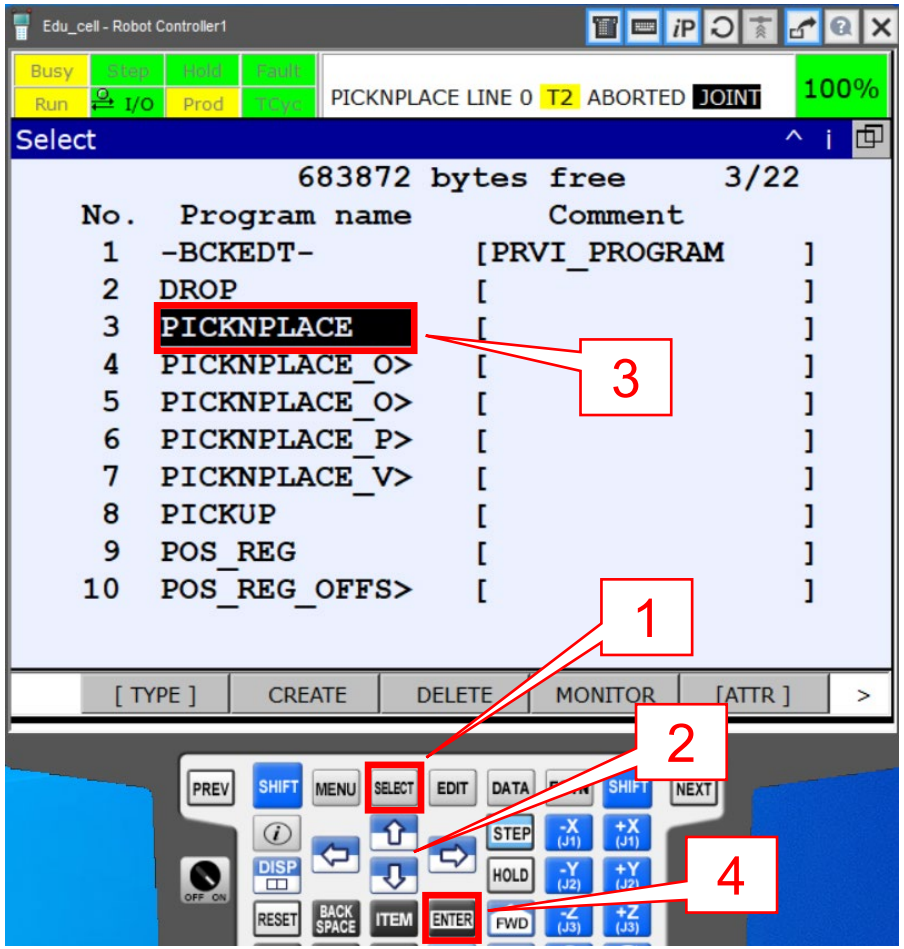
- **nastavitev začetnih vrednosti**
  - **TF** – orodje, ki ga uporabljamo (določa lokacijo TCP in orientacijo orodja),
  - **UF** – „prostor“, v katerem programiramo
  - **payload** – dimenzije, masa/vztrajnost in težišče orodja z/brez izdelkov
  - **registri** oz. spremenljivke …
- **zaporedje točk – položaja in orientacije TCP (orodja oz. TF) v UF in**
- uporaba **drugih ukazov**
  - LBL/JMP LBL,
  - IF/SELECT,
  - FOR/ENDFOR,
  - WAIT, CALL …

PRIMER ROBOTSKEGA PROGRAM

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

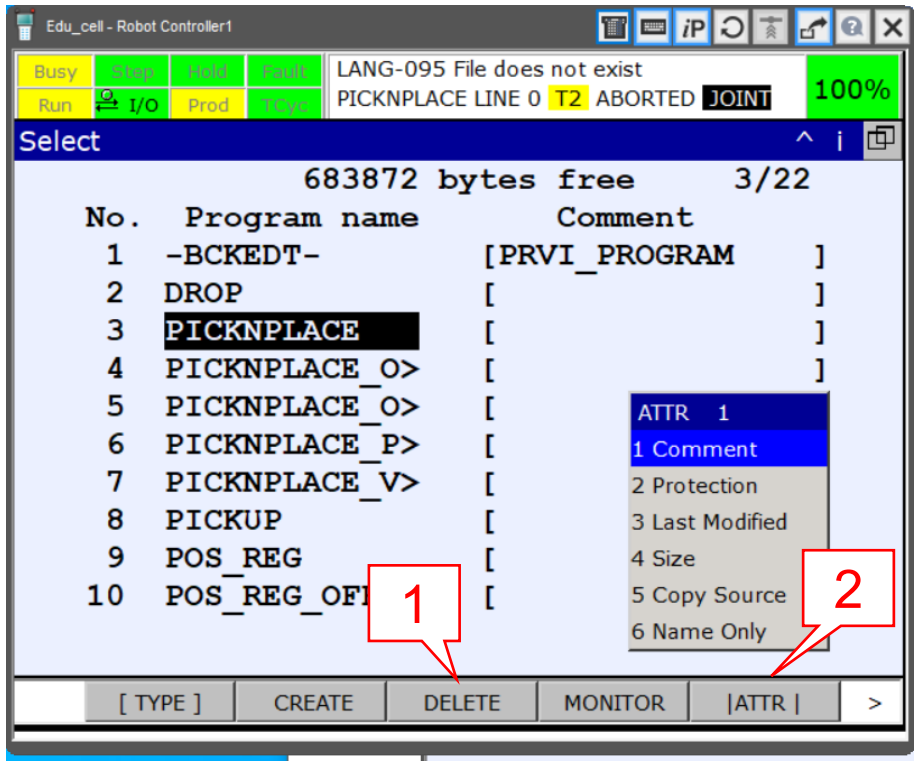
IZBIRA oz. ODPIRANJE PROGRAMA

1. na UE kliknite SELECT,
2. na seznamu s puščicami ↑ ↓
3. poiščite želeni program in
4. izbor potrdite z ENTER



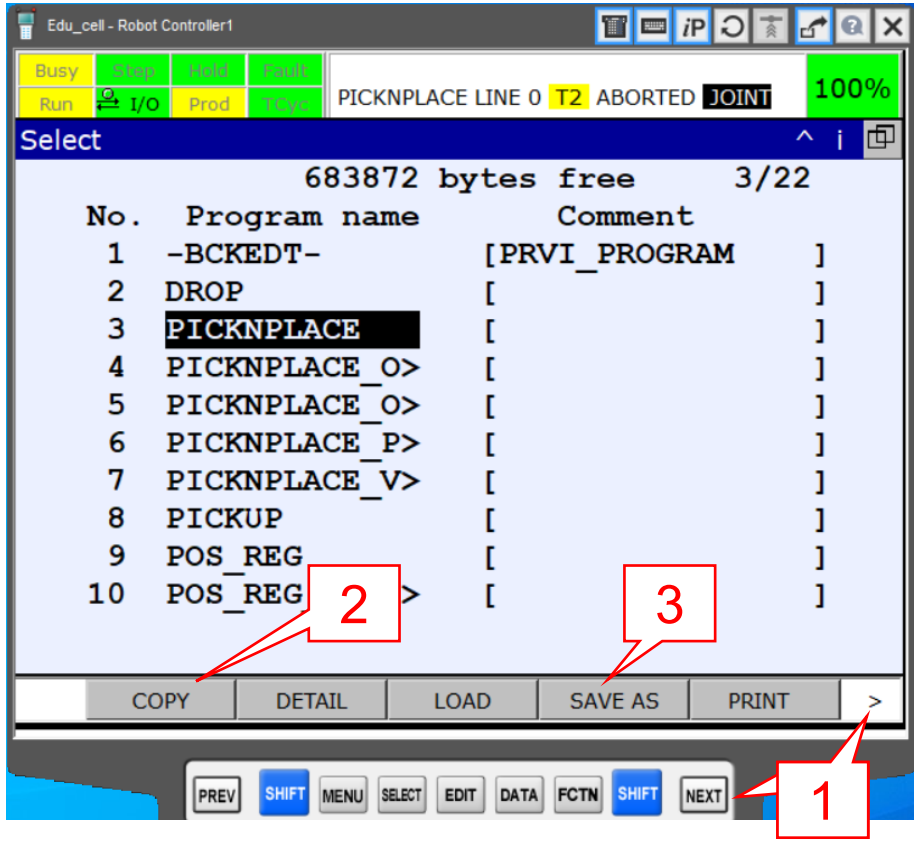
PARAMETRI PROGRAMA

- v oknu izbire (Select) lahko program
  1. s klikom na F3 DELETE izbrišete
  2. s klikom na F5 [ATTR] pogledate njegove parametre
     1. oz. dodate komentar (Comment)
     2. zaščita (Protection)
     3. nazadnje shranjen (Last Modified)
     4. velikost (Size)
     5. izvor (Copy Source)
     6. prikažete samo ime (Name Only)



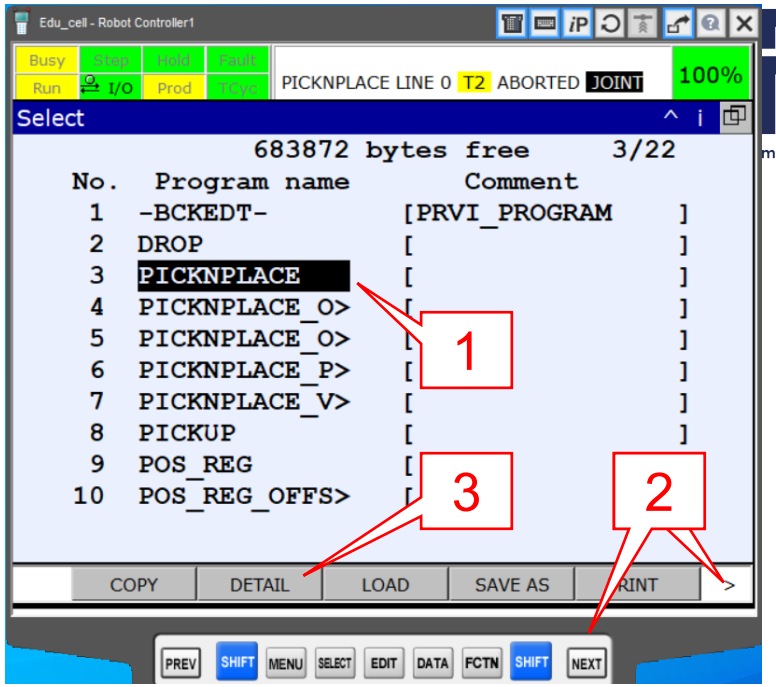
PARAMETRI PROGRAMA

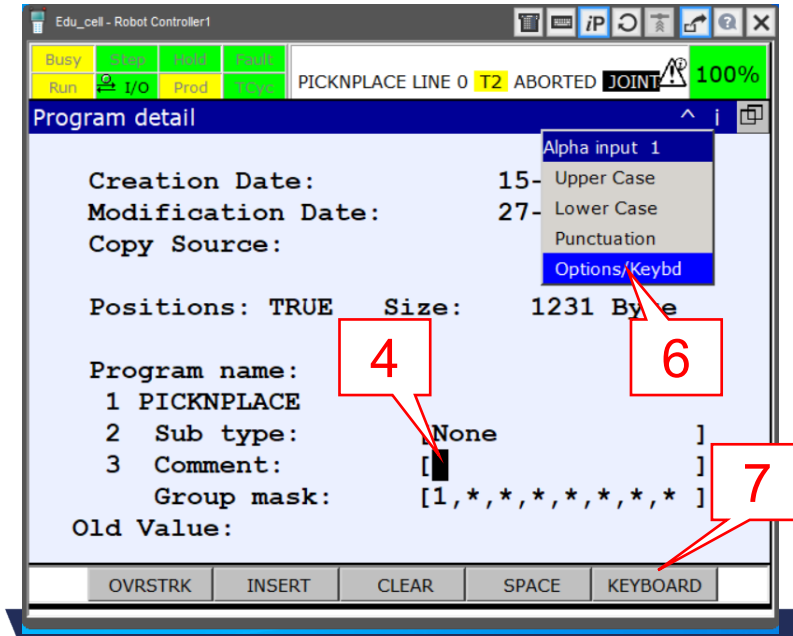
- s klikom na NEXT ali oznako > lahko program:
  1. s klikom na F1 COPY kopirate ali
  2. s klikom na F4 SAVE AS shranite pod drugim imenom

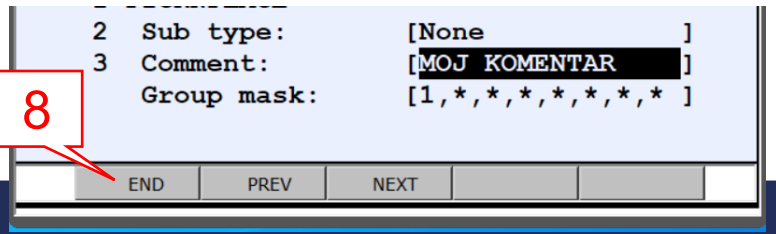


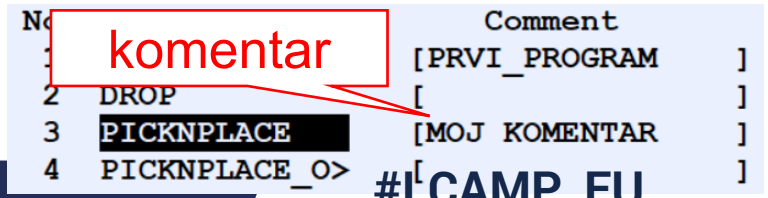
PARAMETRI PROGRAMA

- komentiranje programa
  1. na seznamu označite želeni program
  2. s klikom na NEXT ali oznako > preklopite na
  3. F2 DETAIL in kliknite nanj
  4. na seznamu se pomaknite na Comment
  5. kliknite ENTER
  6. izberite Option/Keybd
  7. kliknite F5 KEYBOARD in vpišite komentar
  8. končate z ENTER in F1 END









TESTIRANJE (T1/T2) ROBOTSKEGA PROGRAMA

- testiranje je **preverjanje delovanja programa** pred avtomatskim, izvajanjem programa
  - je zelo pomembno in **mora** biti izvedeno, da zagotovimo varnost ljudi in preprečimo morebitne poškodbe opreme
  - **koračni način** (STEP) 

    - izvajanje od trenutne programske vrstice, vrstico za vrstico
  - **kontinuirani način** (CONT) 

    - izvajanje programa od trenutne programske vrstice do konca programa, oznake [END] oz. dokler ne pride do napake/prekinitve testiranja

TESTIRANJE IN ZAGON

testiranje (in iskanje napak) na robotu:

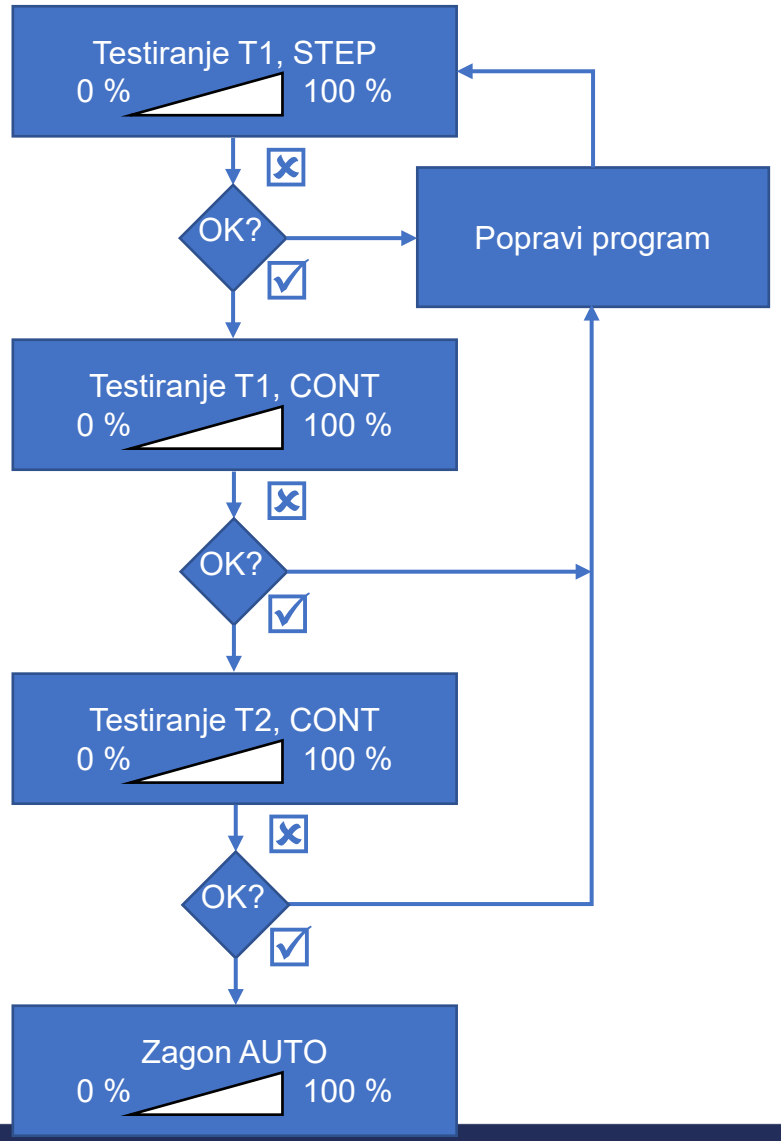
1. T1, koračni način, nizka hitrost
2. T1, kontinuirani način, nizka/srednja/visoka hitrost
3. T2, kontinuirani način, nizka/srednja/visoka hitrost

testiranje (in iskanje napak) v RoboGuide-u:

1. T2, koračni način, nizka hitrost
2. T2, kontinuirani način, nizka/srednja/visoka hitrost

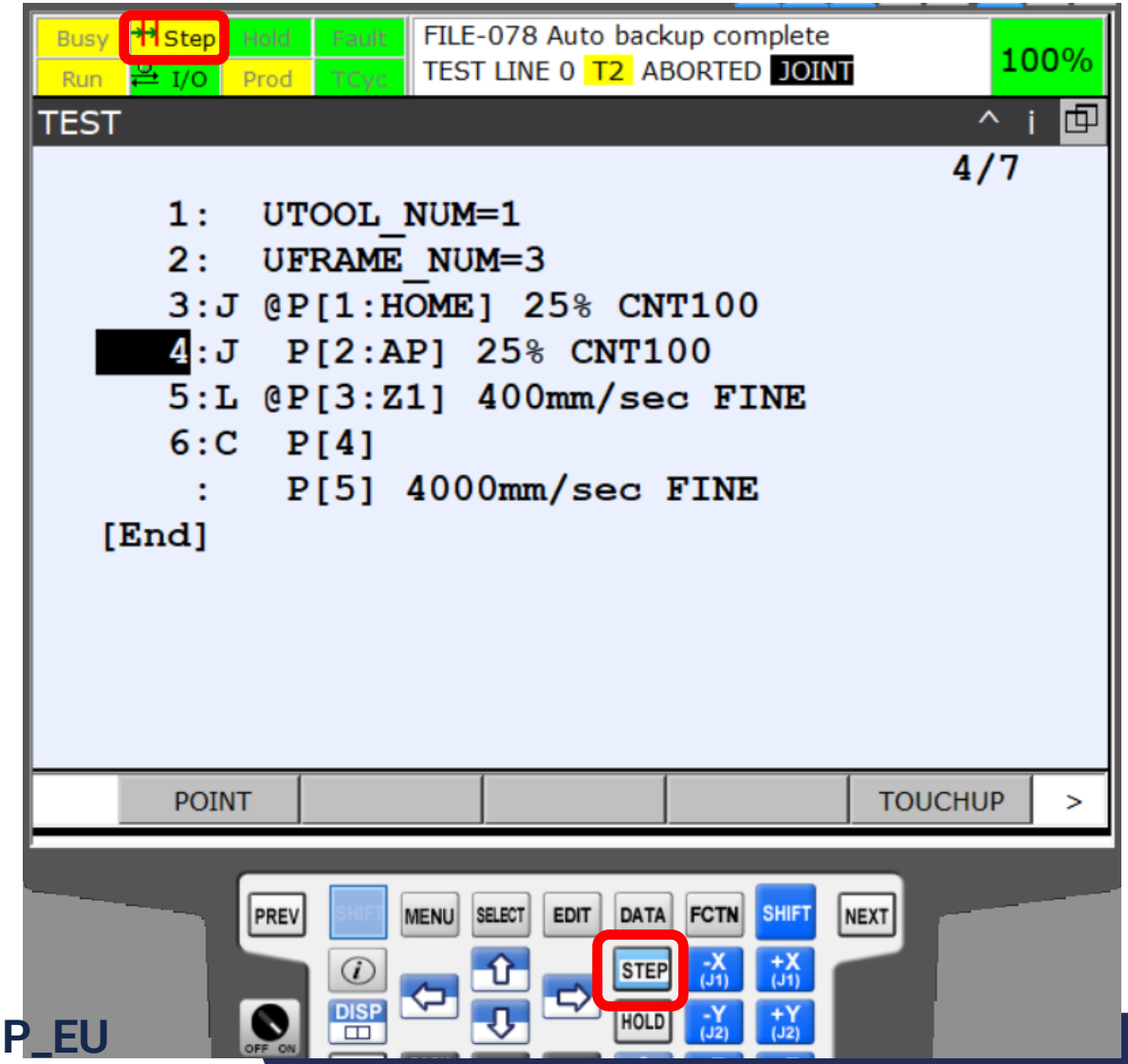
zagon:

1. AUTO način, programirana hitrost



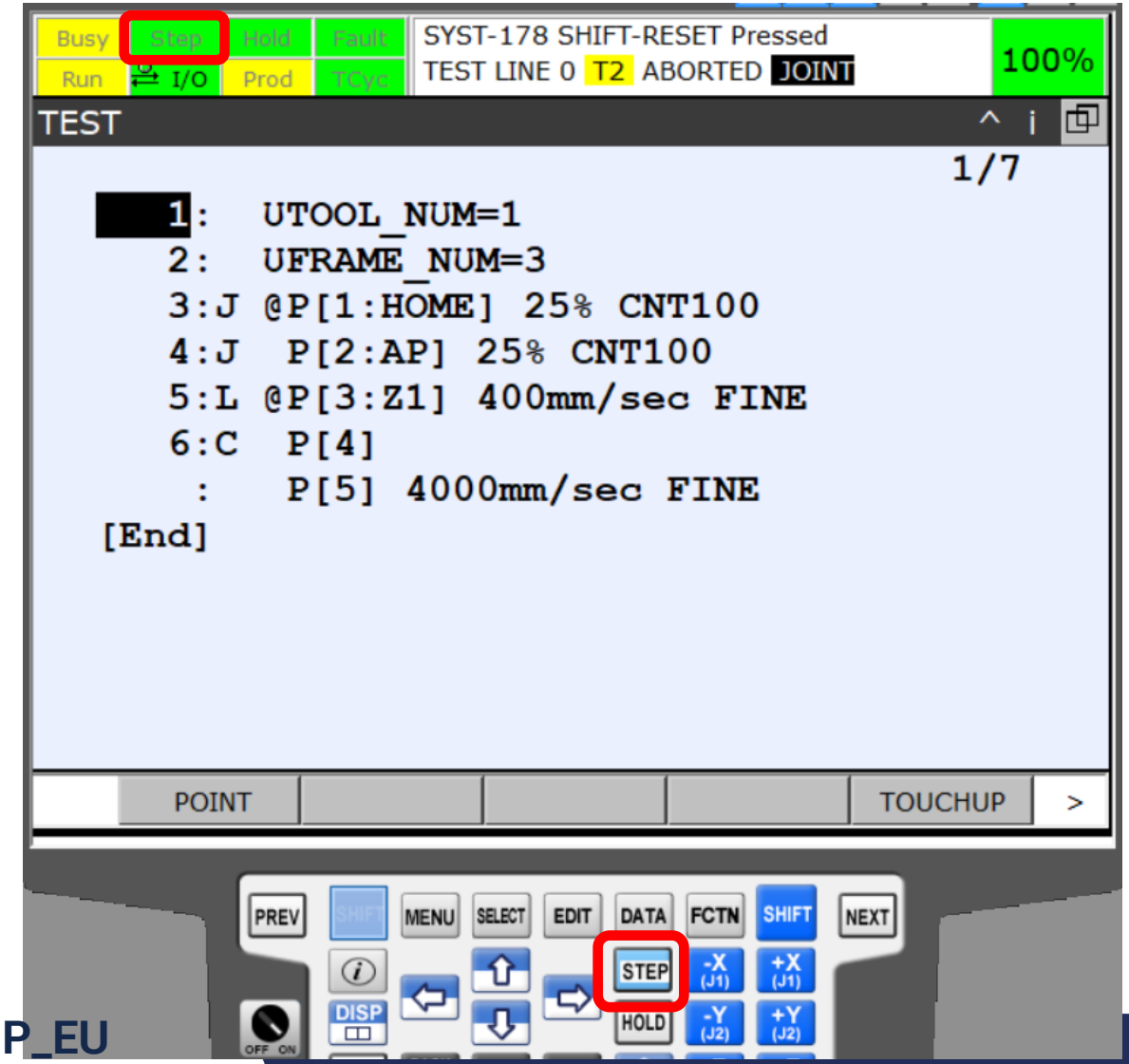
TESTIRANJE (T1)

- **koračni način** (STEP)
  - na krmilniku nastavite T1
  - omogočite učno enoto (ON)
  - izberite program (SELECT …)
  - postavite se v želeno programsko vrstico
  - vključite STEP (ikona na lučki STEP)
  - nastavite nizko hitrost
  - aktivirajte dovolilno tipko in izbrišite napake (RESET)
  - s pritiski na kombinacijo SHIFT + FWD/BWD testirajte programsko vrstico po vrstici naprej/nazaj
  - znak @ označuje trenutno programsko vrstico oz. v kateri točki programa se nahaja TCP



TESTIRANJE (T1, T2)

- **kontinuirani način**
  - na krmilniku nastavite T1 oz. T2
  - postavite se v želeno programsko vrstico
  - izključite STEP (brez ikone na lučki STEP)
  - nastavite nizko/srednjo/visoko hitrost
  - aktivirajte dovolilno tipko in izbrišite napake (RESET)
  - s pritiskom na kombinacijo SHIFT + FWD testirajte program od izbrane programske vrstice do konca [End] oz. do morebitne prekinitve/napake
  - znak @ označuje trenutno programsko vrstico oz. v kateri točki programa se nahaja TCP



ZAGON PROGRAMA (AUTO)

- po potrebi izberite program (SELECT …)
- postavite se v želeno programsko vrstico (predvidoma v prvo)
- na krmilniku nastavite način delovanja na AUTO
- onemogočite učno enoto (OFF)
- izbrišite napake (RESET)
- na krmilniku s pritiskom na tipko CYCLE START zaženite program
- znak @ označuje trenutno programsko vrstico oz. v kateri točki programa se nahaja TCP

PONOVITE

Learner Centric Advanced Manufacturing Platform

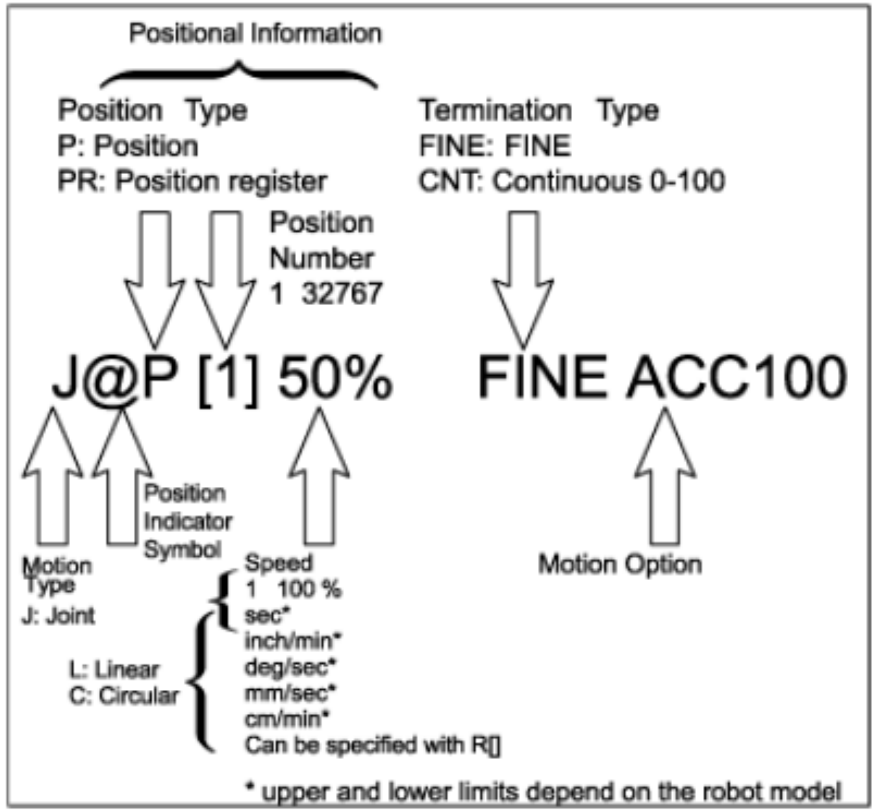
1. Kaj pomeni učiti oz. programirati robota? Kaj je robotski program?
2. Pojasnite kaj v robotskem programu določite z ukazoma UTOOL_NUM in UFRAME_NUM.
3. Kaj pomeni izbrati robotski program? Kaj lahko s programom naredimo v oknu izbire (Select)?
4. Pojasnite razliko med testiranjem in zagonom programa. Zakaj je testiranje programa zelo pomembno?
5. Katere načine testiranja poznamo in pojasnite pravilni postopek testiranja programa.
6. Opišite koračni način testiranja programa.
7. Opišite kontinuirani način testiranja programa.
8. Opišite zagon programa.

UKAZ ZA GIBANJE

- primer ukaza za gibanje:

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

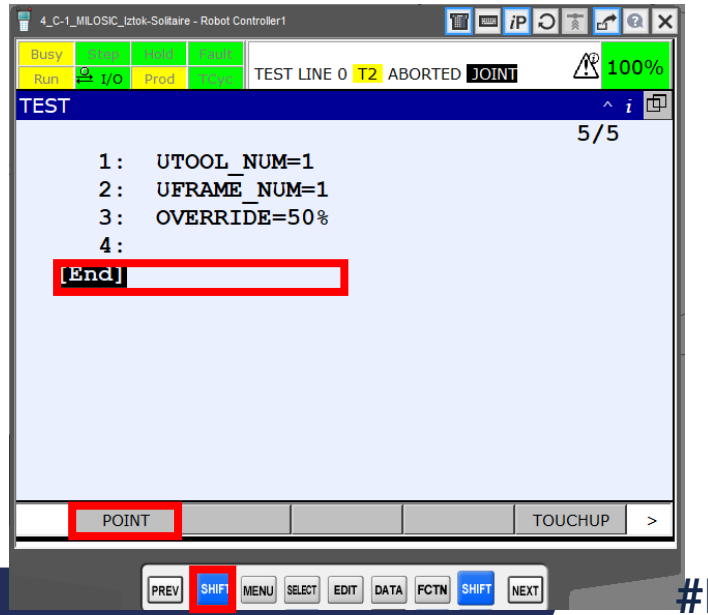
a) način gibanja (J, L, C)
b) oznaka lokacije orodja (TCP)
c) točka, položaj in orientacija TCP (P[1], PR[2,1])
d) hitrost (mm/sec, %, deg/sec, sec …)
e) natančnost, terminacija (FINE, CNT)
f) dodatni parameter/i gibanja

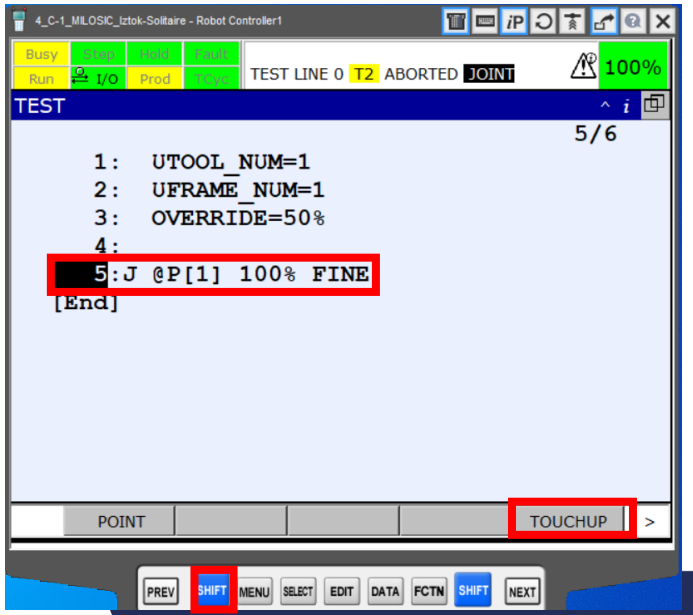


UKAZ ZA GIBANJE

- SHIFT + F1 POINT – v želeni vrstici tvorite ukaz za gibanje
- SHIFT + F5 TOUCHUP – v vrstici z ukazom za gibanje popravite koordinate TCP in UF ter TF

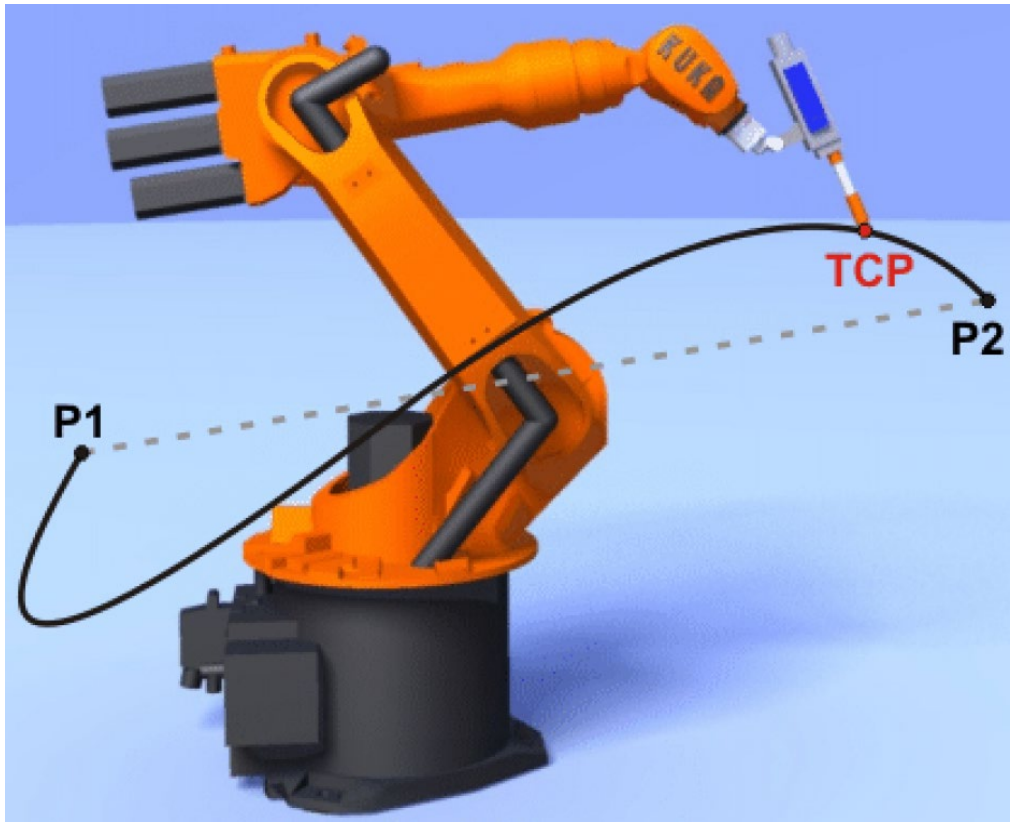
VEDNO SE PRVO S TCP POSTAVITE V ŽELENO LOKACIJO IN ŠELE POTEM IZVEDITE ENEGA OD ZGORNJIH UKAZOV!!!





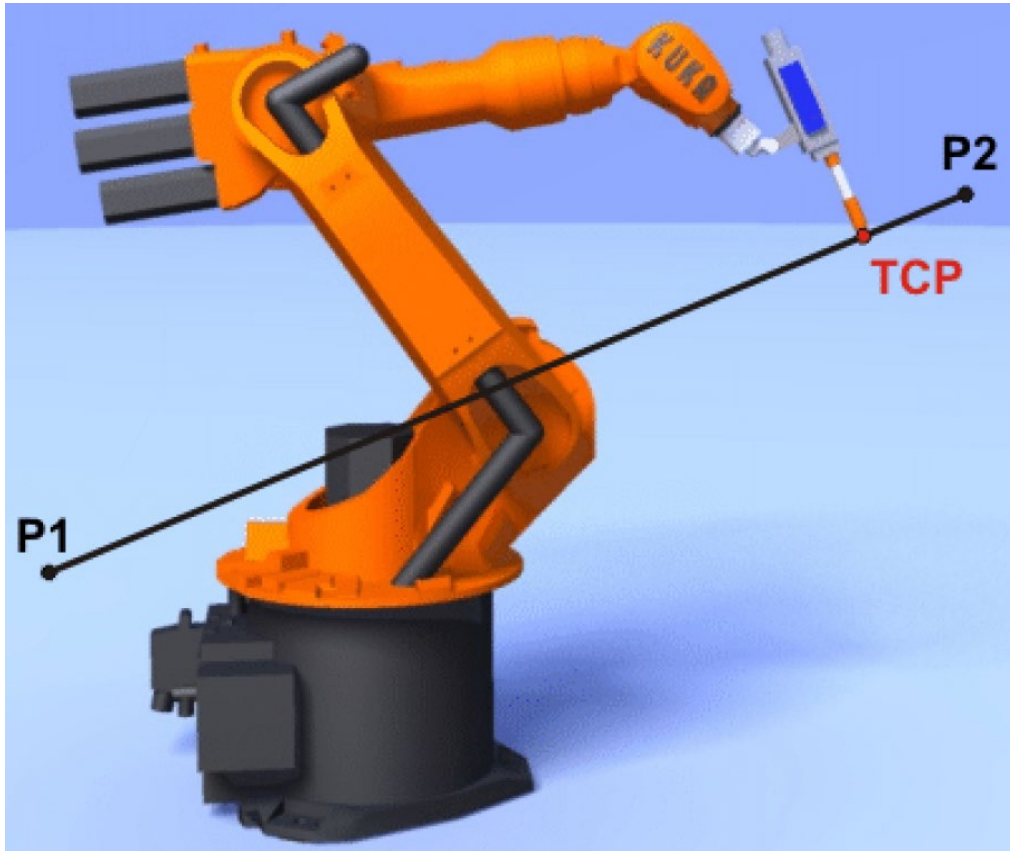
a) NAČIN GIBANJA

- **prosti gib J (joint)**
- robot sam izračuna najhitrejšo pot oz. trajektorijo
- je najhitrejši in najenostavnejši gib za robota
- **pot gibanja, hitrost in orientacija orodja (TCP) v naprej niso znani**, zato ga je potrebno s pazljivostjo uporabiti
- **osno značilno gibanje**



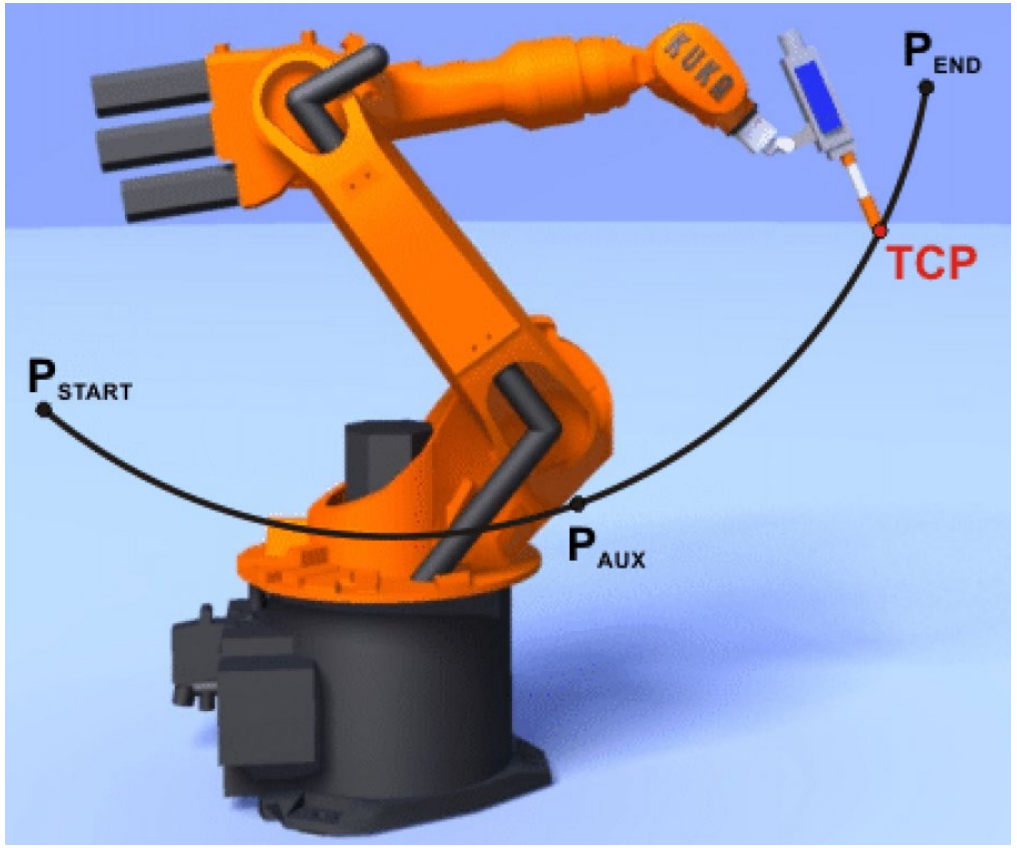
a) NAČIN GIBANJA

- **linearni gib L (linear)**
- orodje (TCP) se linearno giblje iz začetne v končno točko
- **pot gibanja, hitrost in orientacija orodja (TCP) so v naprej znani**
- **gibanje TCP po poti**



a) NAČIN GIBANJA

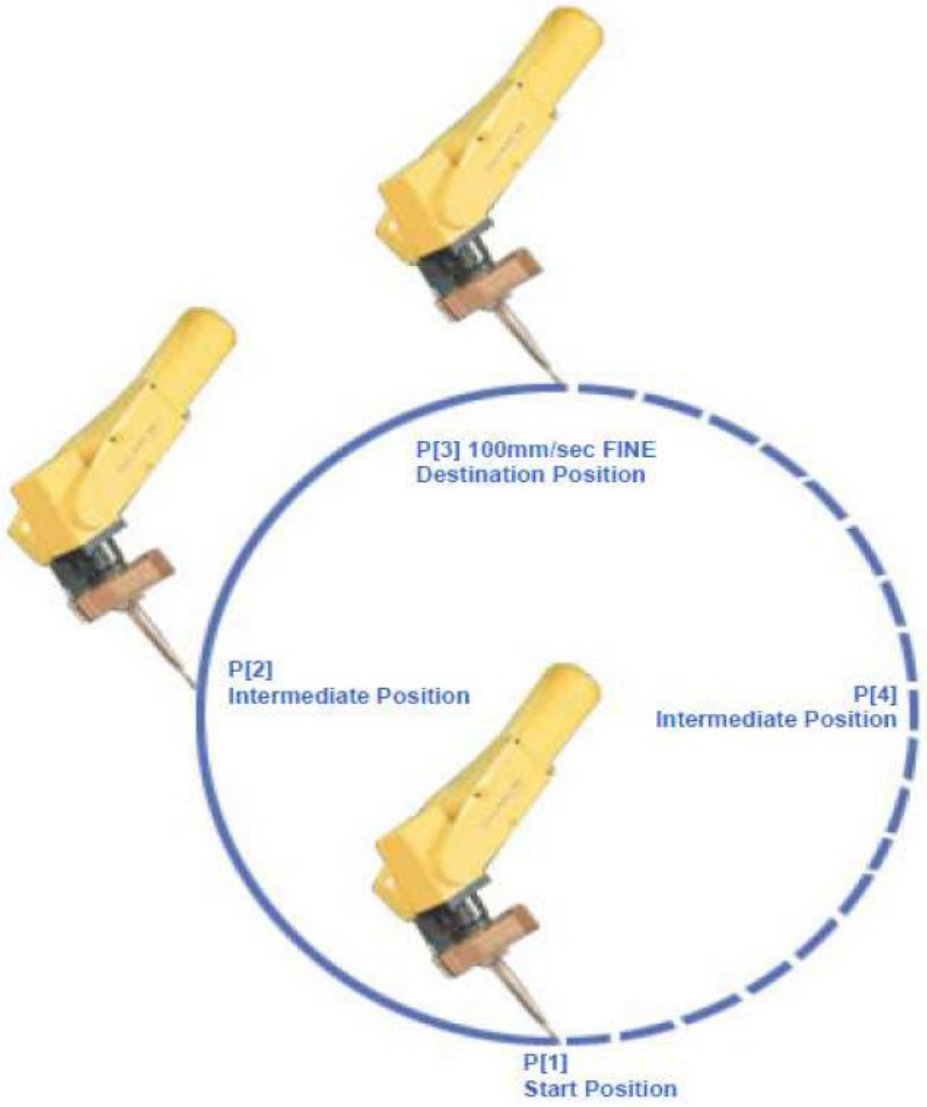
- **krožni gib C (circular)**
- orodje (TCP) se giblje v loku iz začetne točke P[1] skozi vmesno točko P[2] v končno točko P[3]
- **pot gibanja, hitrost in orientacija orodja (TCP) so v naprej znani**
- **gibanje TCP po poti**



a) NAČIN GIBANJA

- **krožni gib C (circular)**
- z enim ukazom lahko izvedemo samo pol krožnice – za celotno krožnico potrebujemo dva polkrožna giba
- ukaz za krožni gib vsebuje vmesno P[2] in končno točko P[3], začetno točko P[1] vsebuje predhodni ukaz

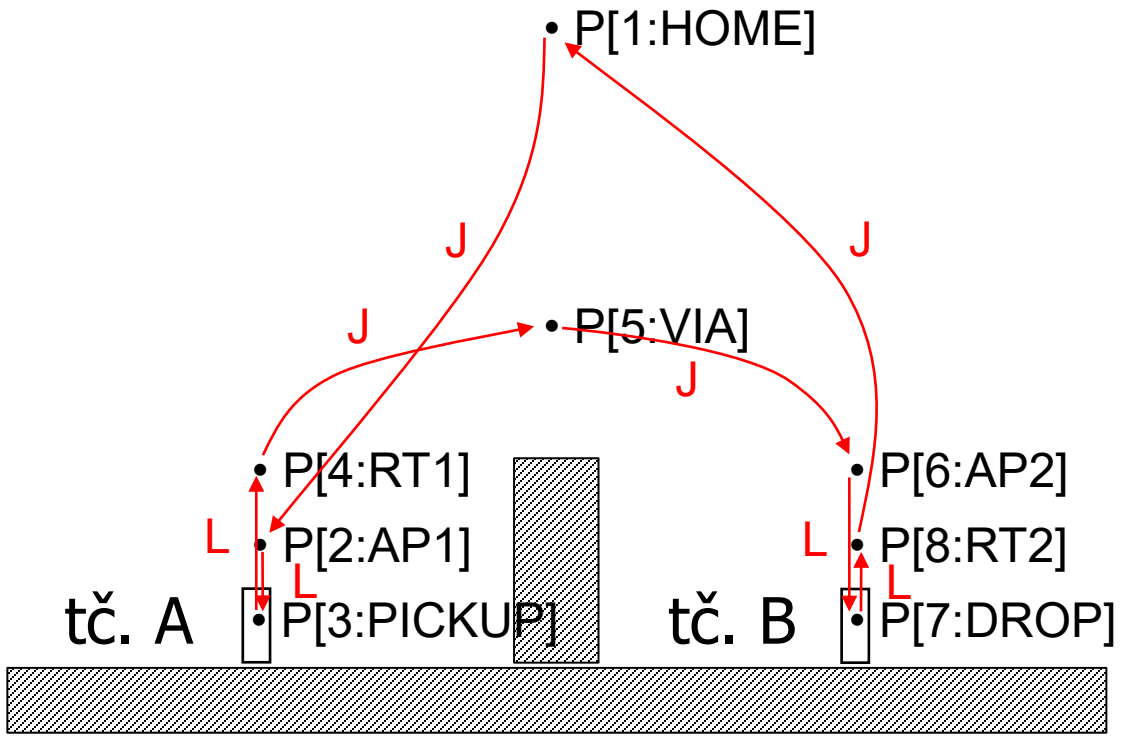
```
1: J P[1] 100% FINE
2: C P[2]
   :    P[3] 100mm/sec FINE
3: C P[4]
   :    P[1] 100mm/sec FINE
```



b) OVIRE

Zgled:

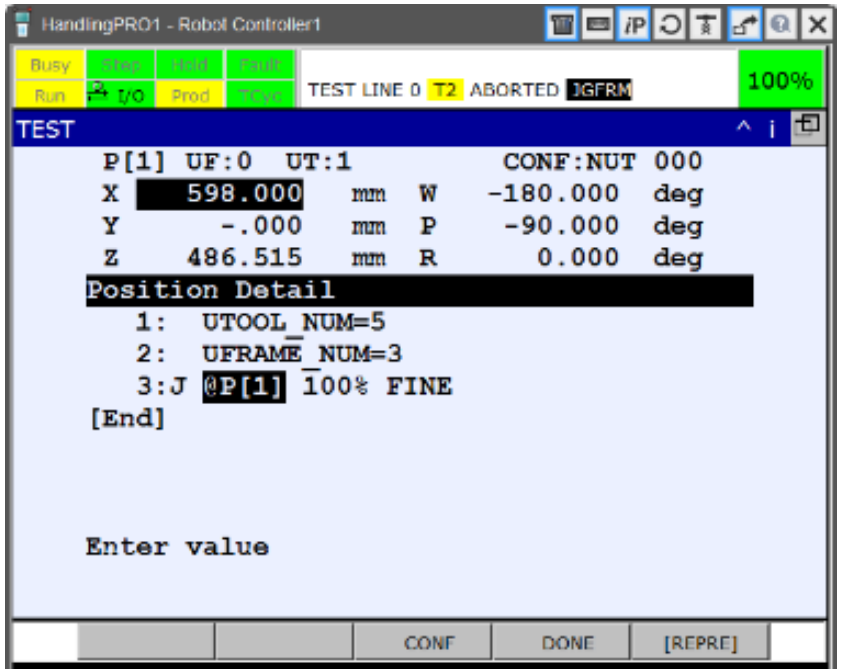
- z robotom želite prenesti predmet iz tč. A preko ovire v tč. B
- robot prične v izhodišču in se tja tudi vrne
- narišite vse točke TCP
- pojasnite/napišite gibe, ki bi jih uporabili



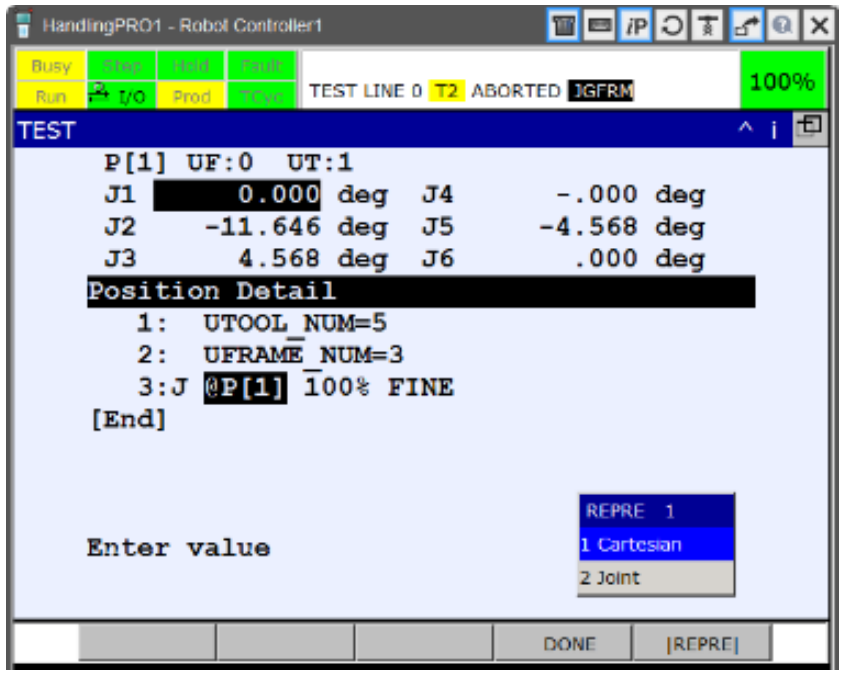
c) POLOŽAJ TCP IN ORIENTACIJA ORODJA

- v točki P[ ] je shranjen položaj in orientacija TCP v:
  1. **kartezični obliki**: X, Y, Z, W, P, R, UTOOL in UFRAME ter CONFIG **ali**
  2. **osno značilni obliki**: J1, J2, J3, J4, J5, J6, UTOOL in UFRAME

Kartezična oblika koordinat

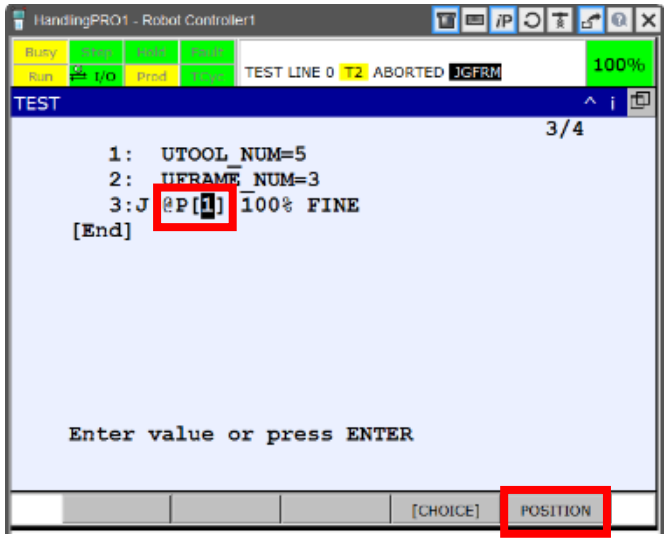


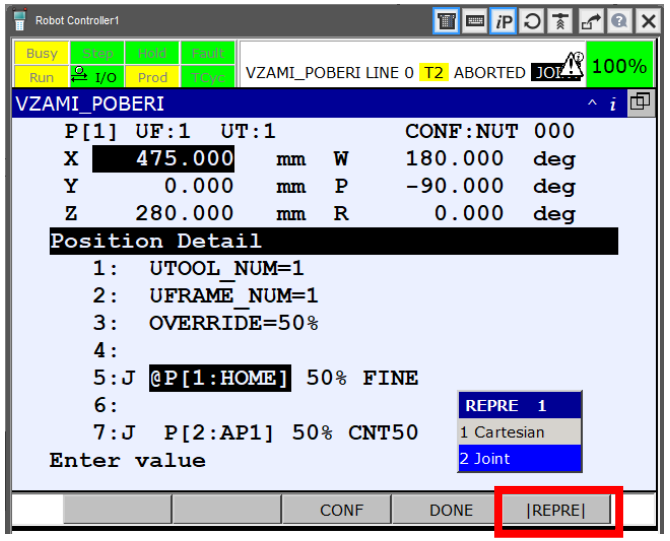
Osno značilna oblika koordinat

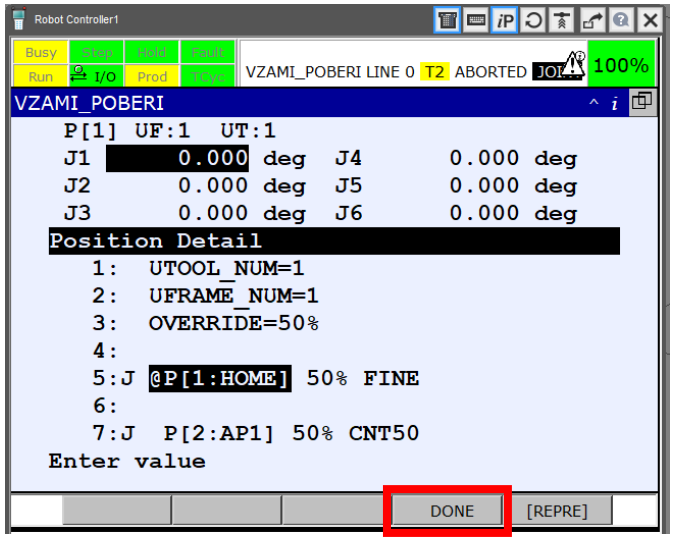


c) POLOŽAJ TCP IN ORIENTACIJA ORODJA

- **koordinate** točke P[ ] **pogledate/spremenite:**
  - **z vnašalko se postavite na točko in kliknete F5 [POSITION]**
  - **F5 [REPRE]**, ustrezno preklopite med
    - **osnimi koordinatami**, **Joint** (J1, J2, J3, J4, J5, J6) ali
    - **kartezičnimi koordinatami**, **Cartesian** (X, Y, Z, W, P, R)
  - vpišite/spremenite želene koordinate in potrdite s F4 DONE







c) POLOŽAJ TCP IN ORIENTACIJA ORODJA

- **konfiguracija** točke

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.

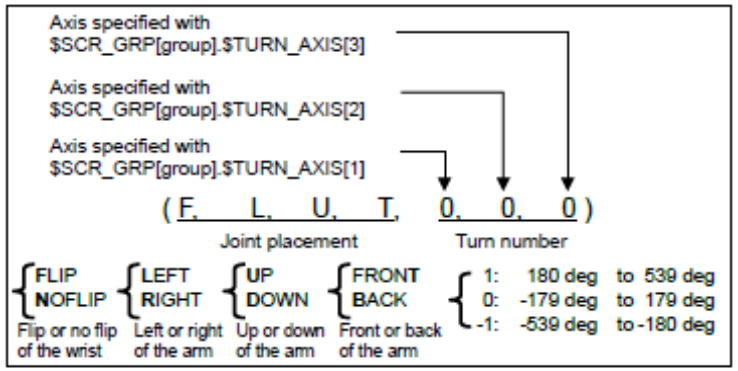


Fig. 4.3.2 (c) Configuration

- **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.

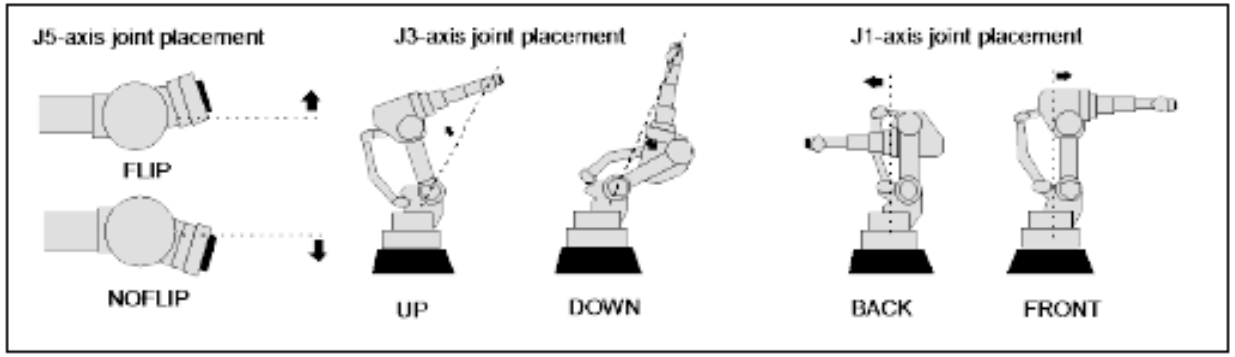


Fig. 4.3.2 (d) Joint placement

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) POLOŽAJ TCP IN ORIENTACIJA ORODJA

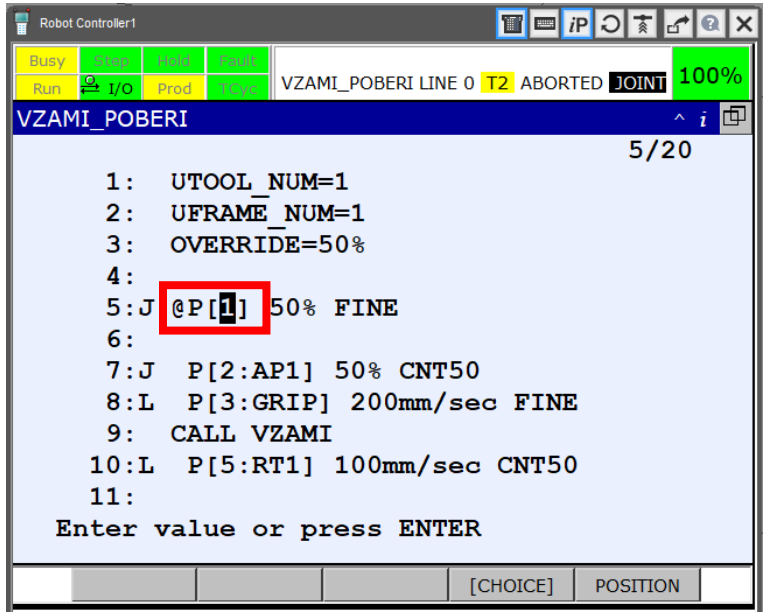
- **konfiguracija** točke

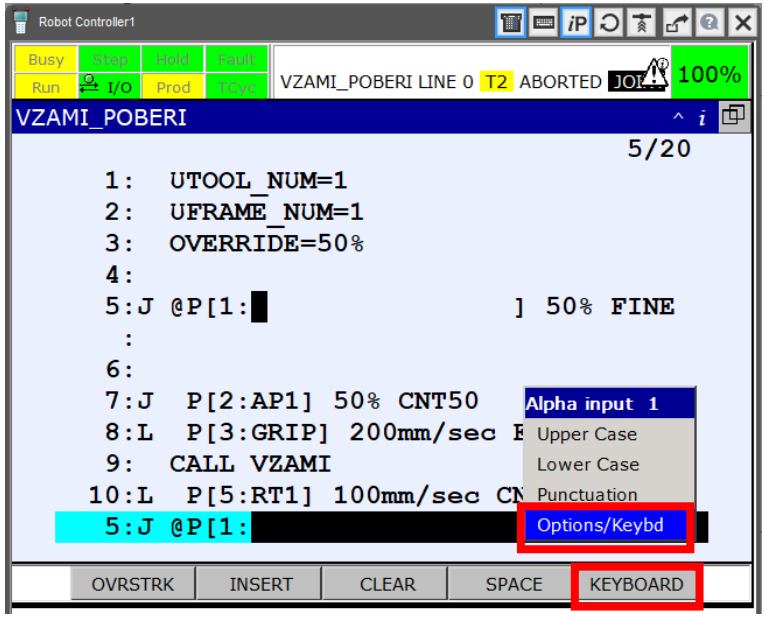
Cartesian coordinate system reference

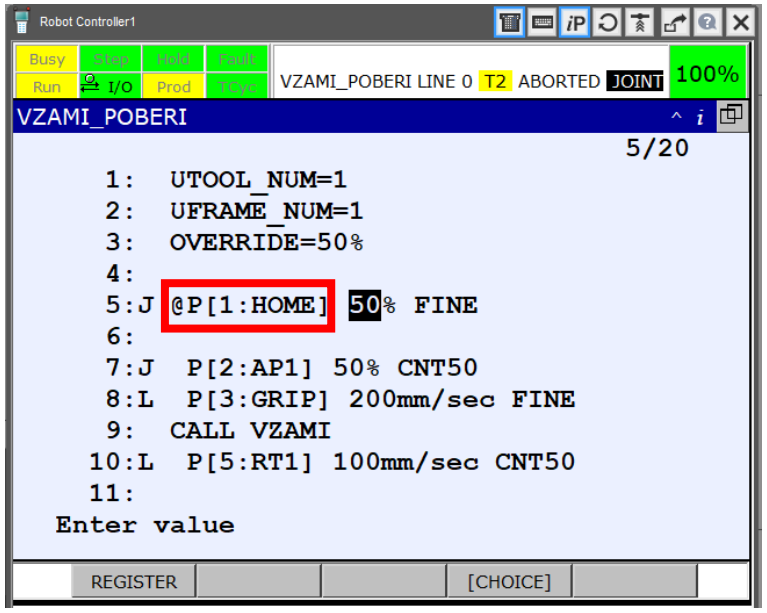
In playback of position data consisting of Cartesian coordinates, a Cartesian coordinate system reference checks the coordinate system number of a Cartesian coordinate system to be used.
If the coordinate system number (a number from 0 to 10 for the tool coordinate system, and a number from 1 to 9 for 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) POLOŽAJ TCP IN ORIENTACIJA ORODJA

- **ime** točke P[ ] **vpišete/spremenite:**
  - z vnašalko se postavite na točko P[ ] in kliknite ENTER
  - iz menija izberite Options/Keybd in kliknite F5 KEYBOARD
  - s tipkovnico vpišite/spremenite ime točke in potrdite z ENTER

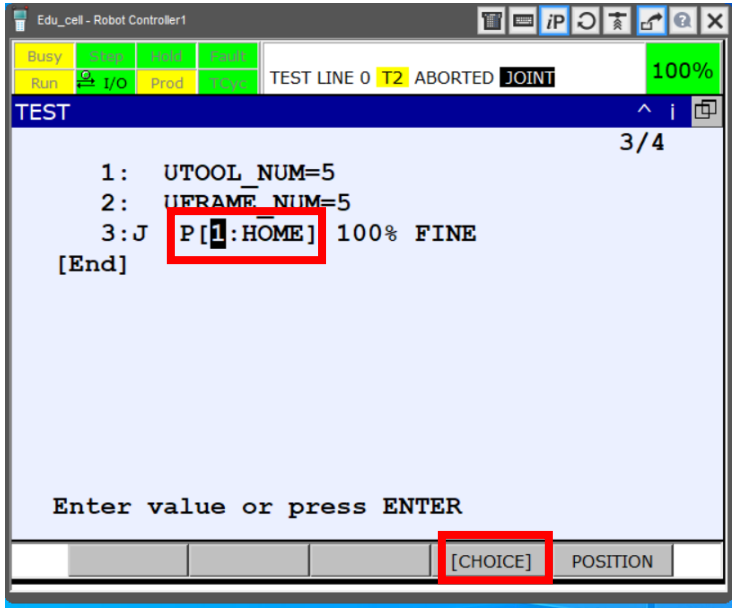


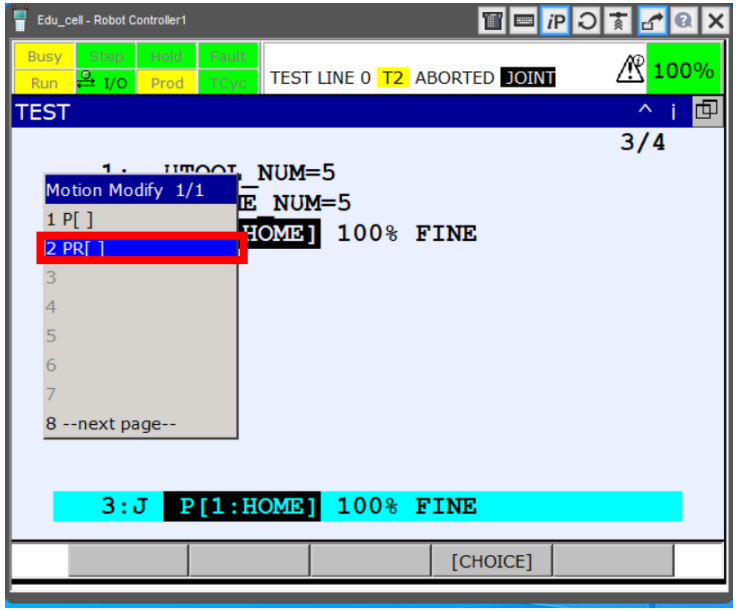


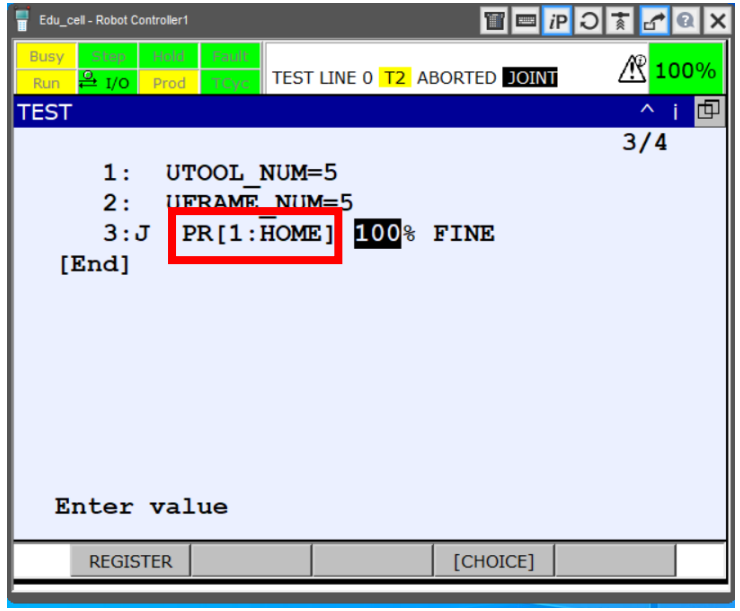


c) POLOŽAJ TCP IN ORIENTACIJA ORODJA

- točka TCP je tudi lahko shranjena v pozicijskem registru PR[ ]
  - **s koordinatami** v PR[ ] lahko **operiramo** (+, -, *, / …) oz. **programsko spreminjamo**
    - na točki P[ ] kliknete F4 CHOICE
    - izberete PR[ ] in potrdite z ENTER

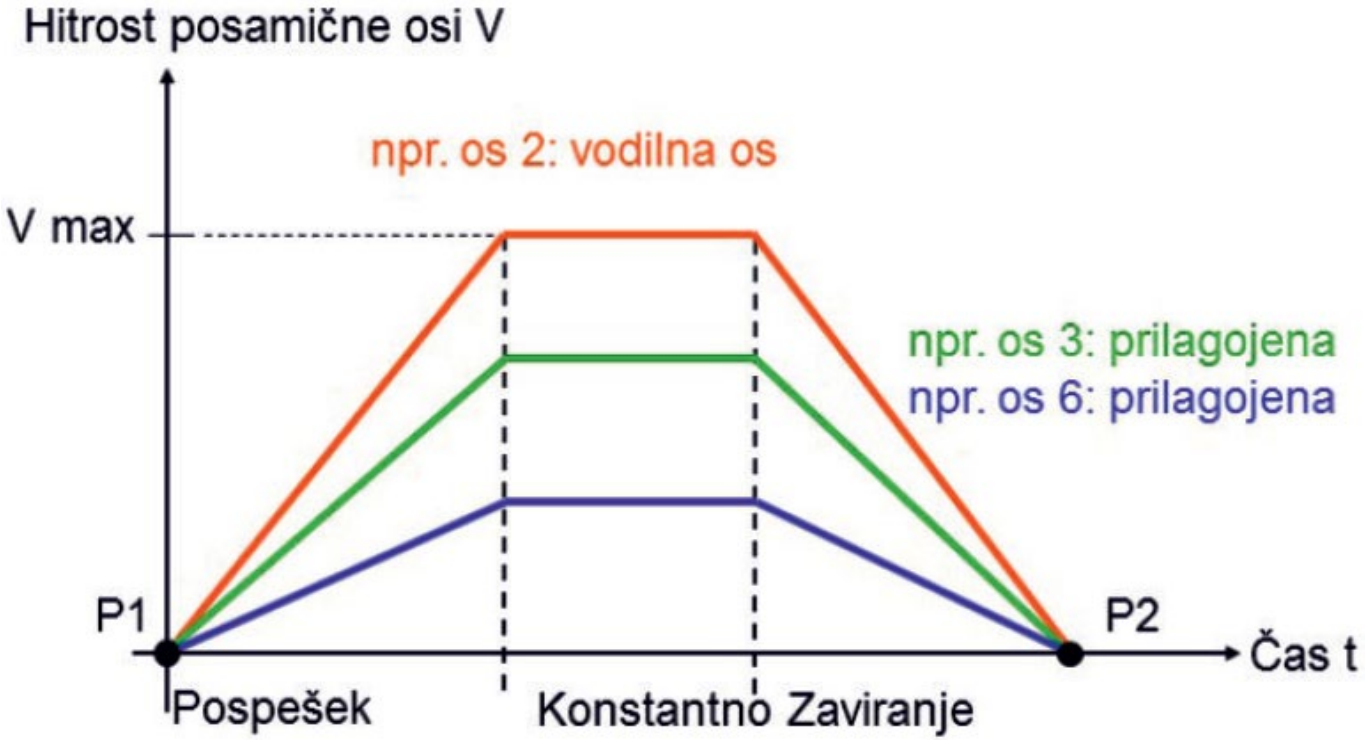






d) HITROST

- JOINT (%, sec)
  - ni možno vpisati večje hitrosti od maksimalne hitrosti robota
  - hitrost vodilne osi – os, ki potrebuje največ časa, da doseže ciljno točko

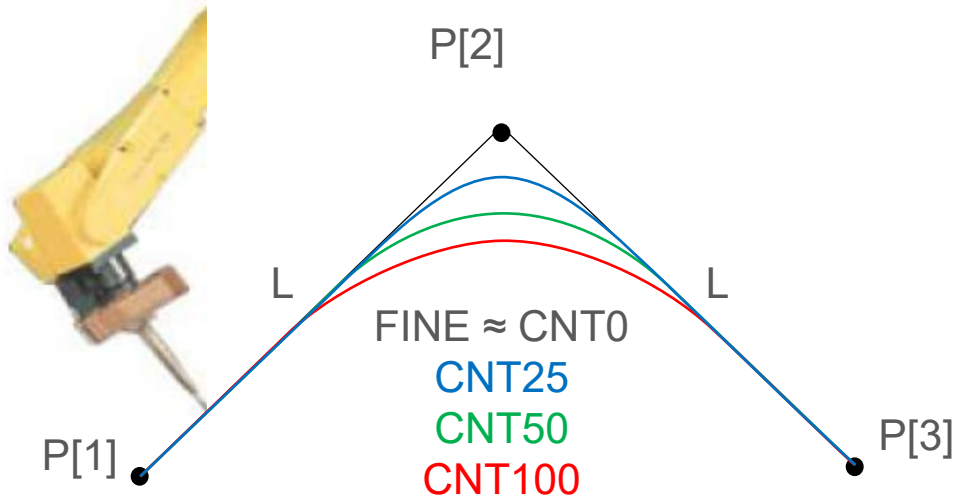


d) HITROST

- LINEAR, CIRCULAR (**mm/sec**, cm/min, inch/min, deg/sec, sec)
  - ni možno vpisati večje hitrosti od maksimalne hitrosti robota
  - hitrost TCP
  - v določenih primerih TCP ne doseže zahtevane hitrosti, npr. varilne aplikacije, ko robotska roka naredi velik gib za majhno rotacijo orodja

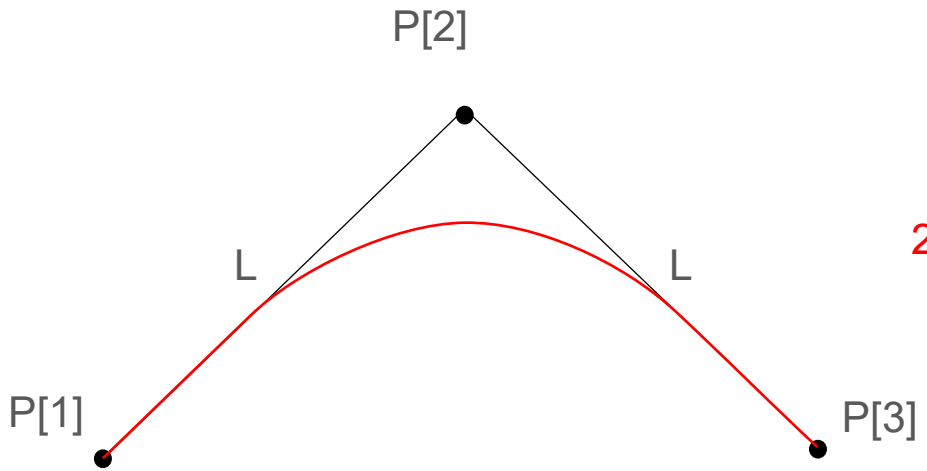
e) NATANČNOST, TERMINACIJA

- določa, kako natančno TCP doseže točko
  - **FINE**
    - **natančno** doseže točko in se v njej **TCP ustavi**
  - **CNT** – CONTINUOUS
    - določimo za koliko se **TCP približa točki, TCP se ne ustavi**
    - CNT nastavimo vrednost med 0 in 100
      - CNT0 ≈ FINE, TCP je najbližje točki (**največje zaviranje**)
      - CNT100, TCP se točki približa za toliko, da se lahko gib izvede s polno hitrostjo oz. brez zaviranja

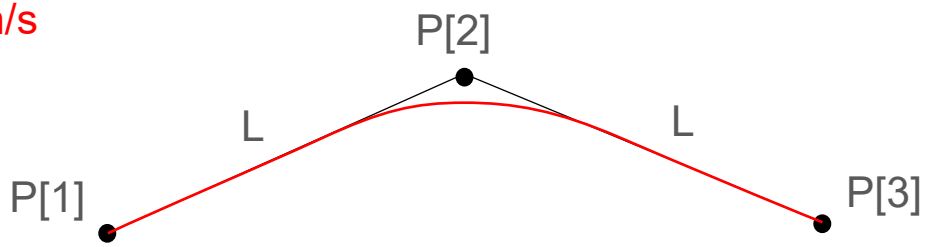


e) NATANČNOST, TERMINACIJA

- Zgled 1

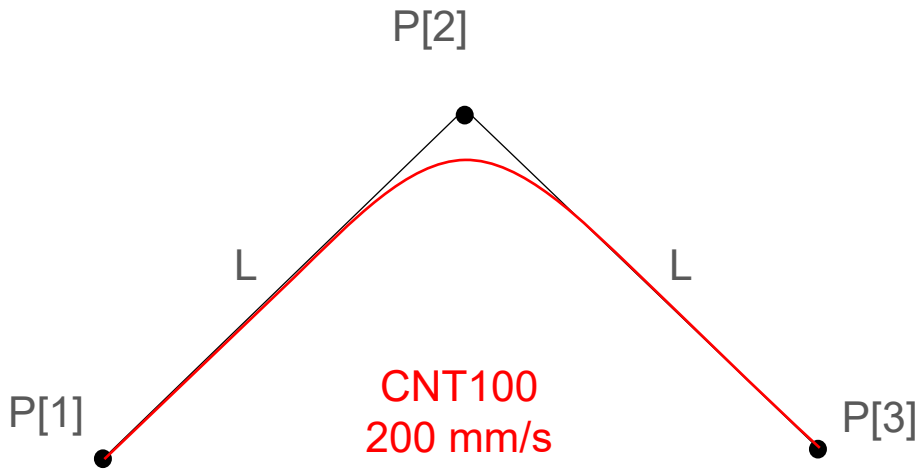


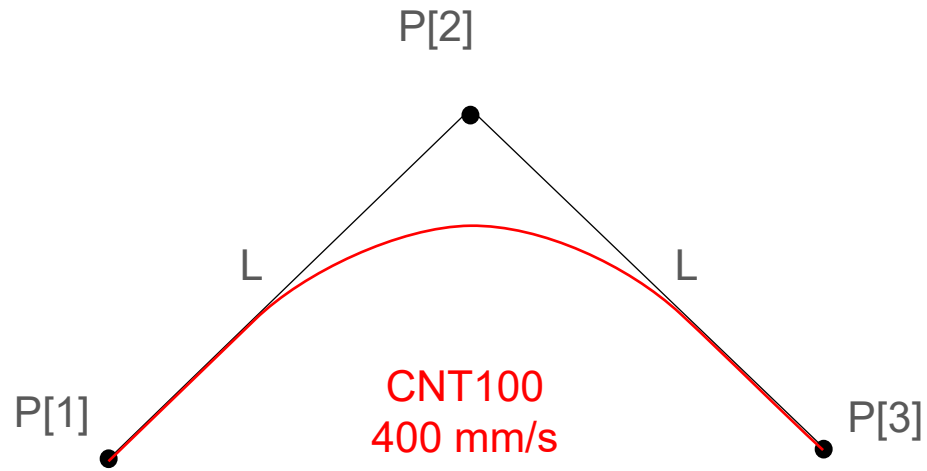
CNT100
200 mm/s



e) NATANČNOST, TERMINACIJA

- Zgled 2



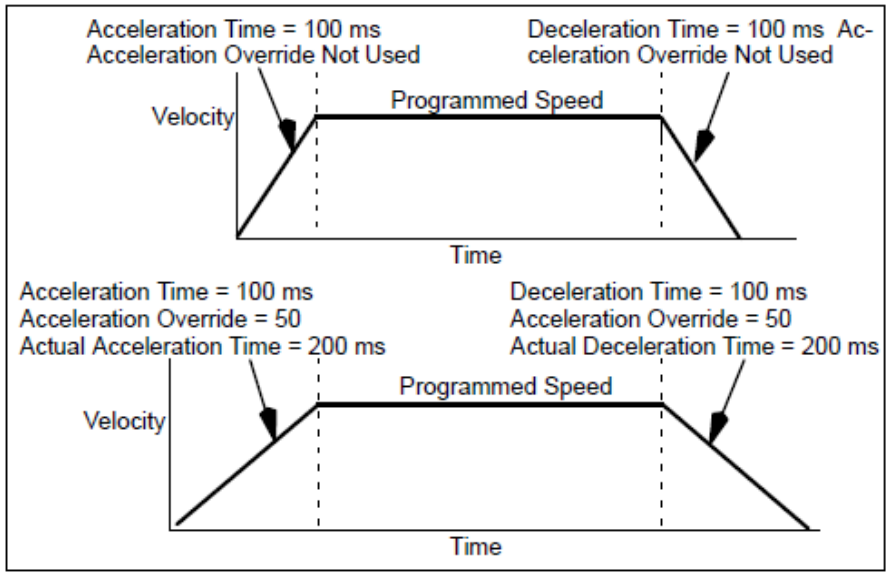


e) NATANČNOST, TERMINACIJA

- prednosti:
  - dosežemo večje hitrosti gibanja robota – skrajšamo delovni cikel
  - zmanjšamo ali odstranimo vibracije robota
  - daljša življenjska doba robota
  - robot mora „plesati kot balerina“ …

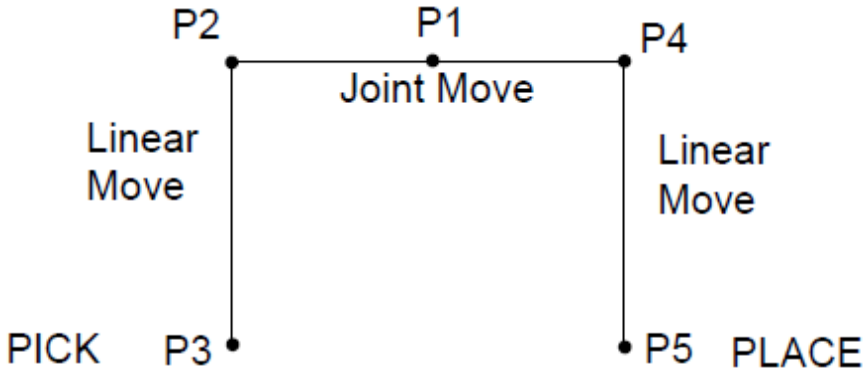
e) DODATNI PARAMETER GIBANJA: ACC

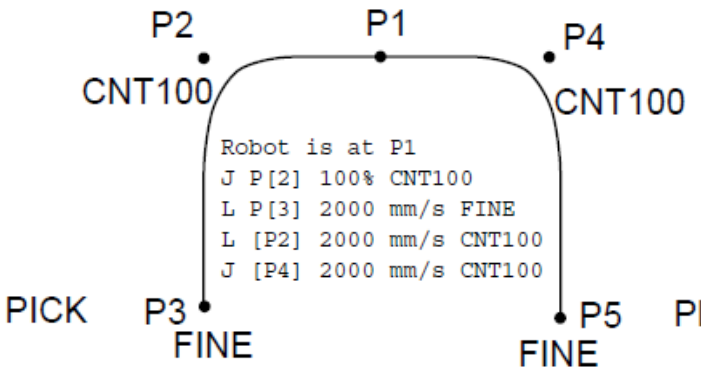
- ACC50
- je vrednost v % med 20 % in 150 %
- vrednost 50 % pomeni, da bo robot potreboval 2 × več časa za pospeševanje (accelerate) in zaviranje (decelerate)
- vrednost nad 100 % je potrebno uporabljati s pazljivostjo, saj pomeni večje obremenitve za robota

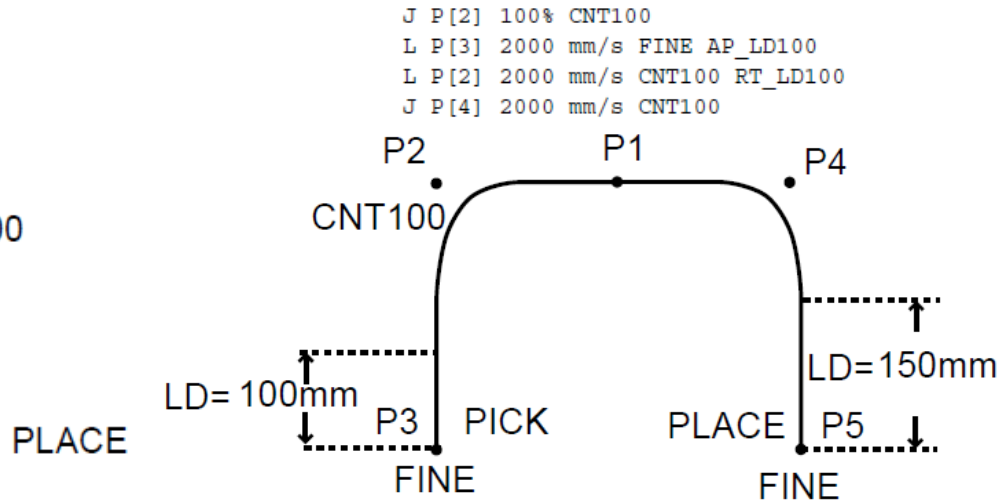


e) DODATNI PARAMETER GIBANJA: AP_LD, RT_LD

- **AP_LDXX** (APproach_LinearDistance), **RT_LDXX** (ReTrieve_LinearDistance), XX v mm, R[]
- primerno za aplikacijo pobiranje in odlaganje (pick & place)
- TCP potuje iz P1 skozi P2 do P3 (pobere) in nazaj skozi P2 in P4 do P5 (odloži), slika levo
- da dosežemo krajši čas cikla, uporabimo parameter CNT, slika sredina
  - težava: ne vemo koliko imamo linearnosti nad P3 oz. P5
  - odprava: lahko eksperimentiramo z različnimi CNT oz. dviganjem/spuščanjem P2 in P4
- bolje: uporabimo parameter LinearDistance, slika desno

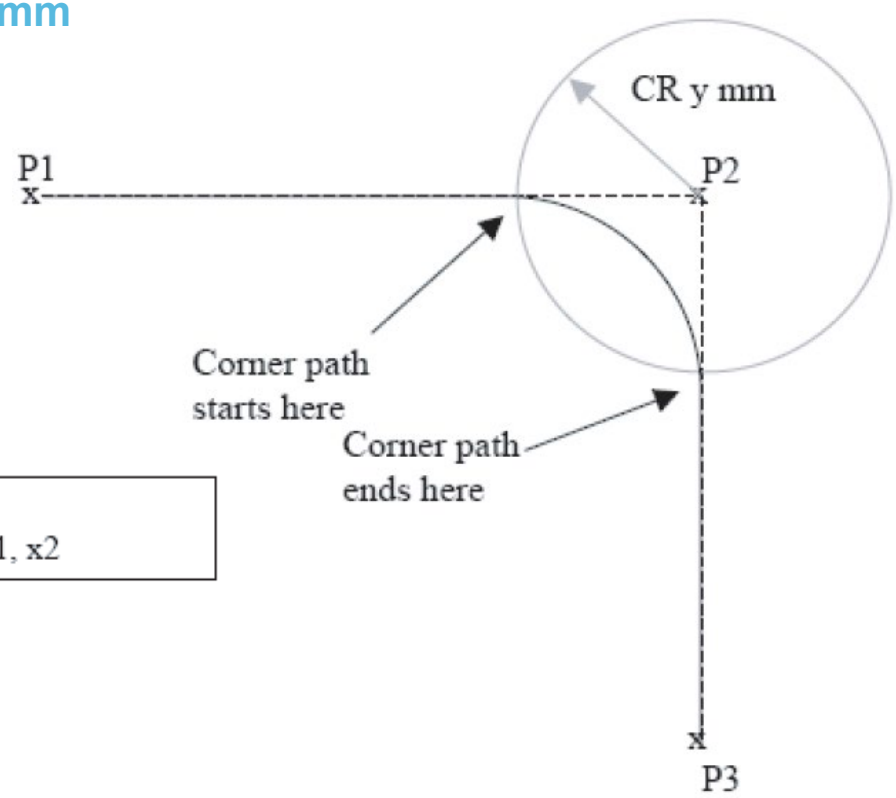




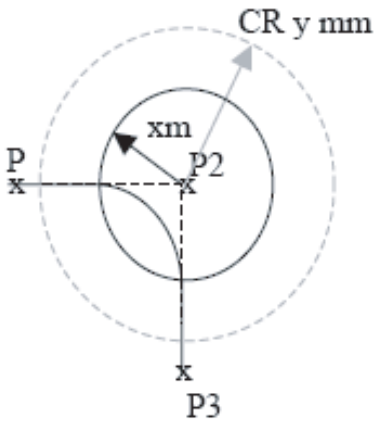


e) DODATNI PARAMETER GIBANJA: CRy

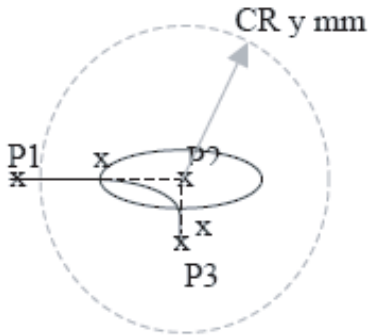
- **CRy** (Corner Region Termination Type), **y podan v mm med 0 in 1000 mm**
- je podprt samo na L in C gibih (L P[2] 100 mm/sec CRy)
- je vrednost, koliko mm pred točko P[2] TCP zapusti pot, naredi krožni lok in se zopet vrne na programirano pot
- y je lahko največ polovico segmenta med P [1] in P[2] oz. med P [2] in P[3]



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

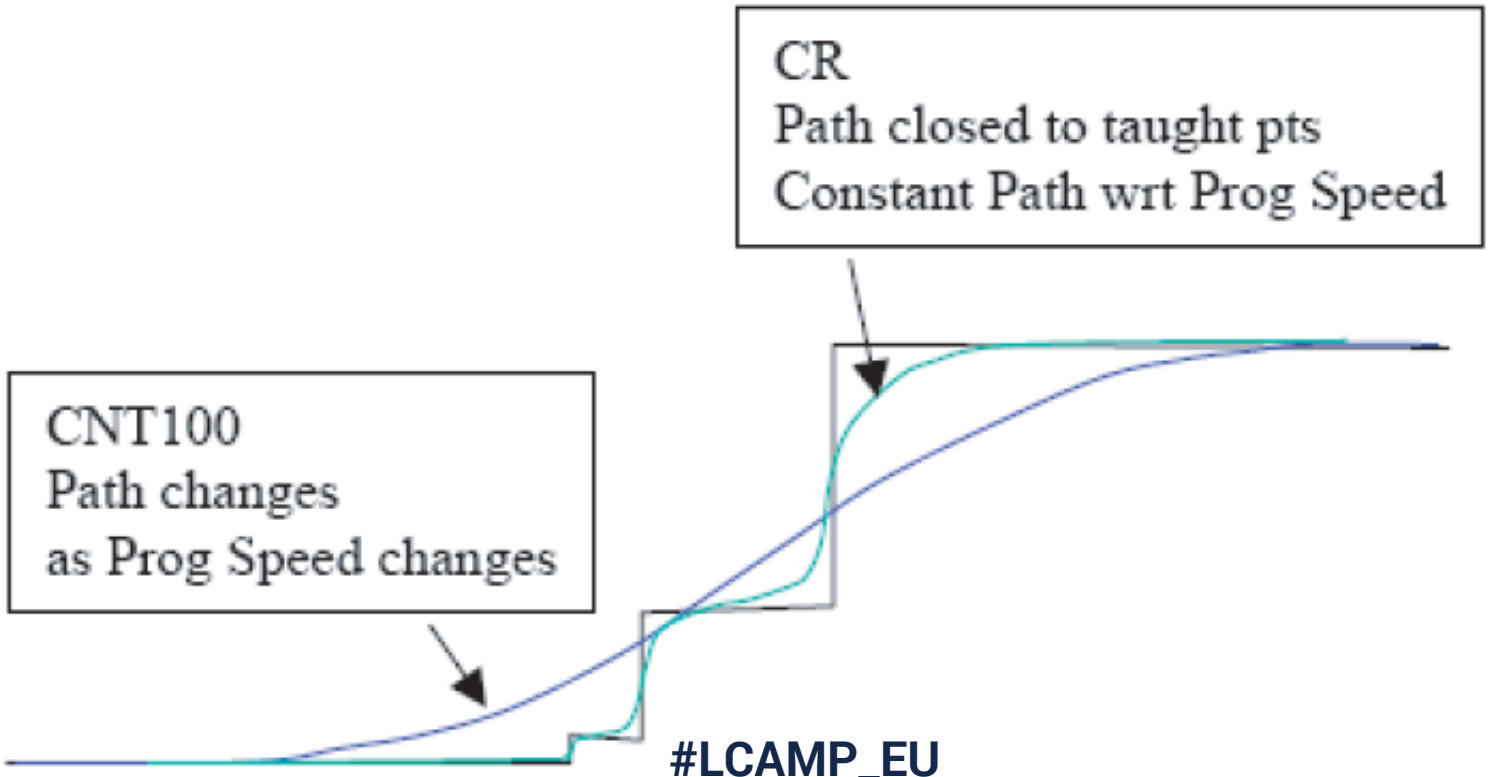


Unequal segment lengths case
Actual CR = half segment distances x1, x2

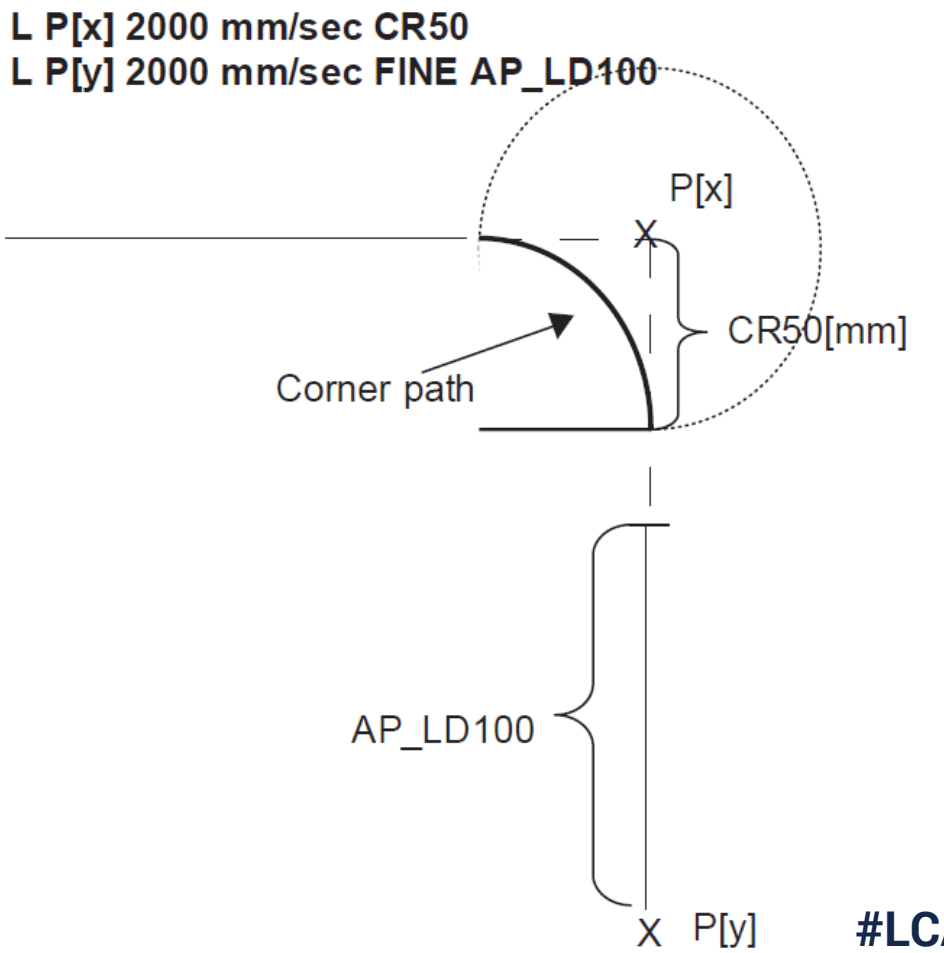


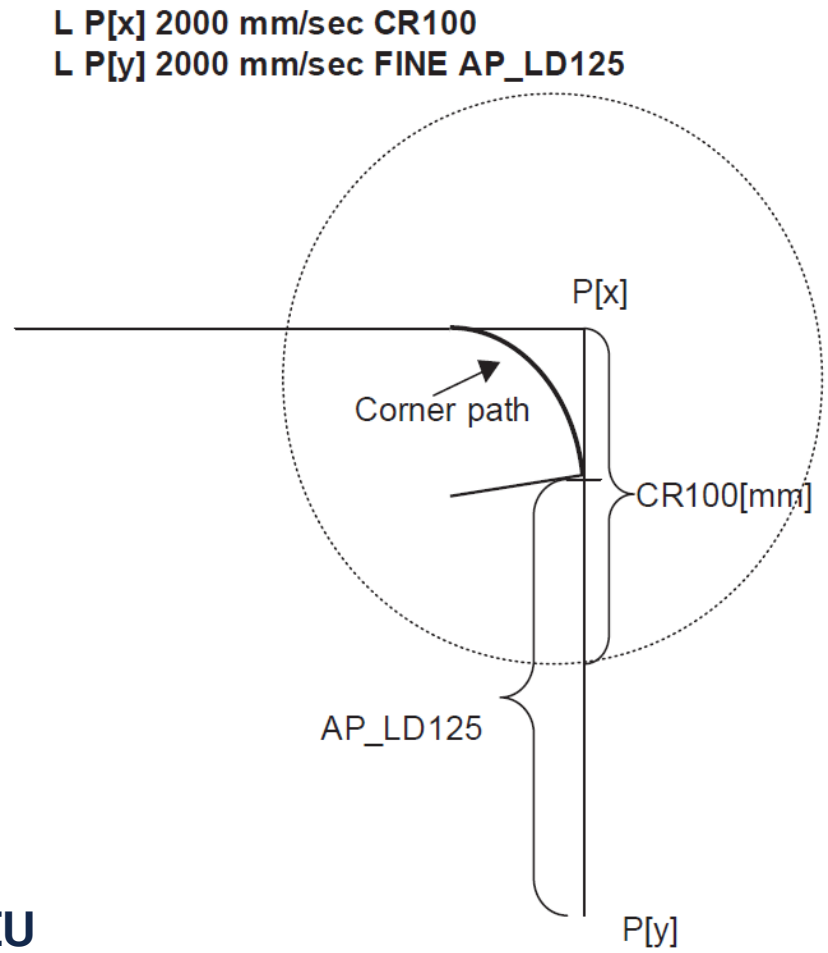
e) PRIMERJAVA PARAMETROV GIBANJA: CNT in CR

- oddaljenost od točk **se pri CNT spreminja** s sprogramirano hitrostjo
- oddaljenost od točk **se pri CR ne spreminja** s sprogramirano hitrostjo



e) UPORABA PARAMETROV GIBANJA: AP_LD in CR





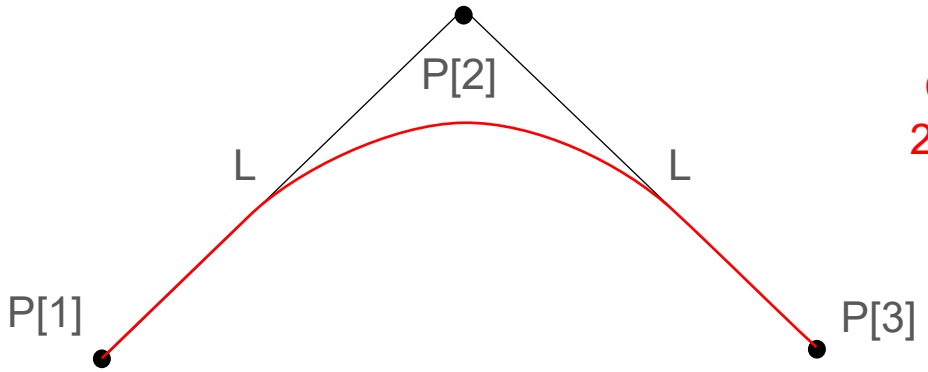
PONOVITE

Learner Centric Advanced Manufacturing Platform

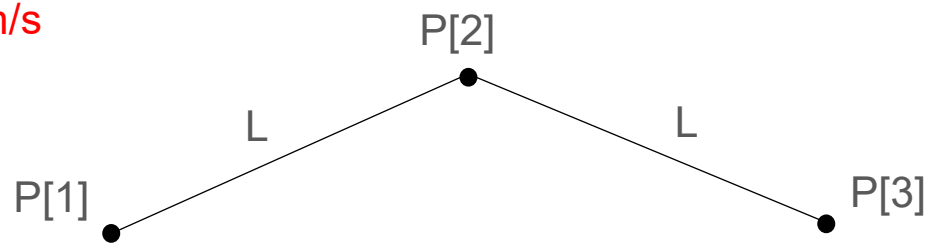
1. Kaj je ukaz za gibanje in kako ga tvorimo? Kako v ukazu za gibanje spremenimo koordinate v trenutni položaj TCP-ja?
2. Opišite prosti gib.
3. Opišite linearni gib.
4. Opišite krožni gib.
5. Kako je lahko shranjena točka programa? Kako pogledamo koordinate točke?
6. Na kakšen način so lahko v točki P[ ] shranjene koordinate? Kako med njimi preklopimo?

PONOVITE

7. Kako točki dodamo ime? Zakaj točke poimenujemo?
8. Kaj je prednost, če je točka shranjena v PR[ ]?
9. Pojasnite, na kakšen način je podana hitrost giba in na kaj se nanaša ta hitrost (kaj potuje s to hitrostjo)?
10. Opišite natančnost oz. terminacijo točke. Kaj so prednosti uporabe parametra CNT?
11. Narišite pot gibanje TCP-ja.

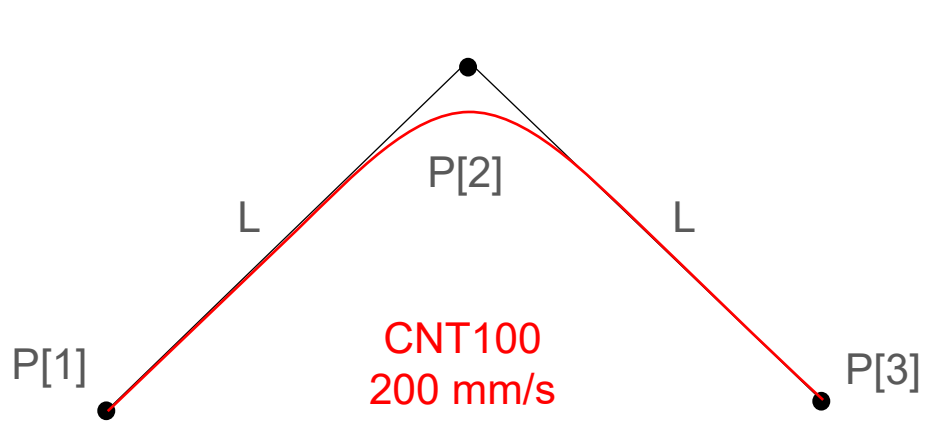


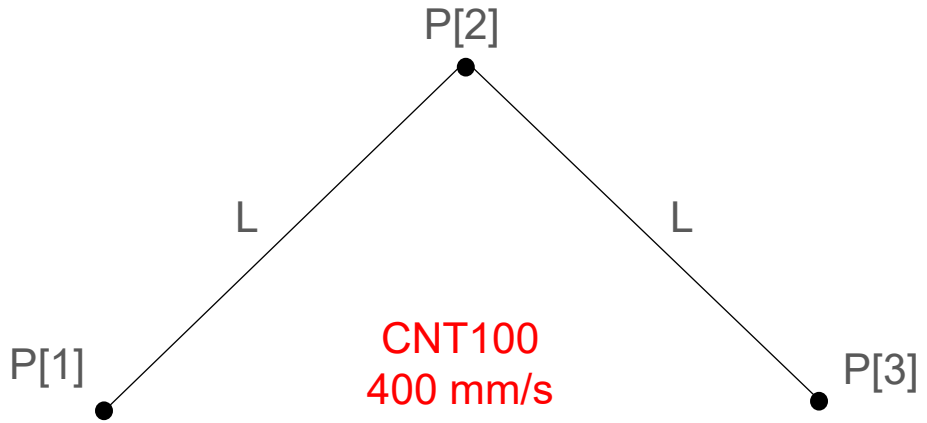
CNT100
200 mm/s



PONOVITE

12. Narišite pot gibanje TCP-ja.





13. Opišite dodatni parameter ACC.

OPIS UČENJA / PROGRAMIRANJA NA UE (TEACH PENDANT)

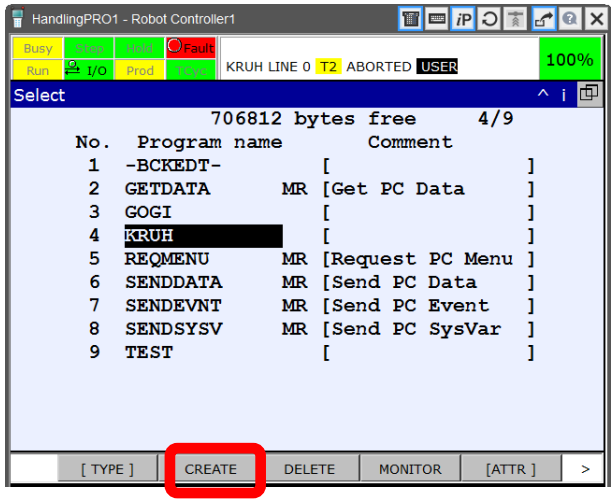
Learner Centric Advanced Manufacturing Platform

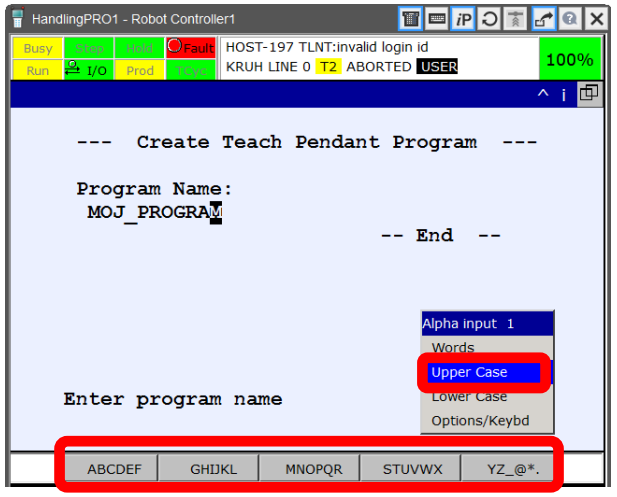
1. Izdelajte nov prazen robotski program.
2. Vstavite orodje (TF oz. TCP), s katerim in prostor (UF), v katerem bo robot programiran.
3. Nastavite začetne vrednosti (spremenljivke – registre, payload, override …)
4. Premaknite TCP v ciljno točko.
5. Vstavite ukaz za gibanje.
6. Nastavite parametre gibanja (način gibanja, točka, hitrost, terminacija …).
7. V tej točki nastavite potrebne aktivnosti (zapri/odpri prijemalo, preveri stanje vhodov in ustrezno ukrepaj, aktiviraj izhode …)
8. Ponavljajte korake 4 do 7, po potrebi tudi 2 (v primeru zamenjave orodja in/ali prostora).

POSTOPEK UČENJA / PROGRAMIRANJA NA UE (TEACH PENDANT)

Learner Centric Advanced Manufacturing Platform

1. Nastavite KS orodja (metoda 3 oz. 6 točk) – TOOL FRAME (TF), (če je potrebno).
2. Nastavite uporabniški KS (metoda 3 točk) – USER FRAME (UF), (če je potrebno).
3. Izdelajte nov, prazen program s smiselnim imenom.
   - SELECT / F2 CREATE
   - Izberete UPPER CASE in s tipkami F1 do F5 napišete ime programa, ENTER
   - s tipko F3 EDIT odprete urejevalnik programa

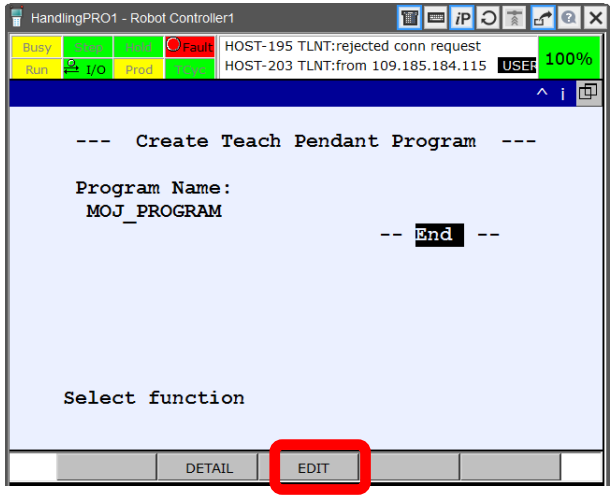


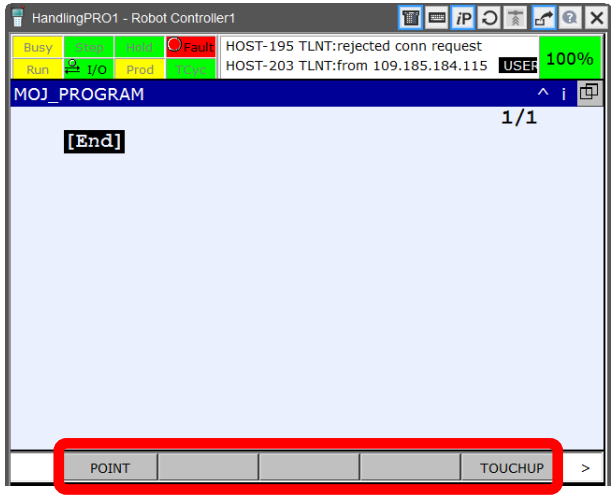


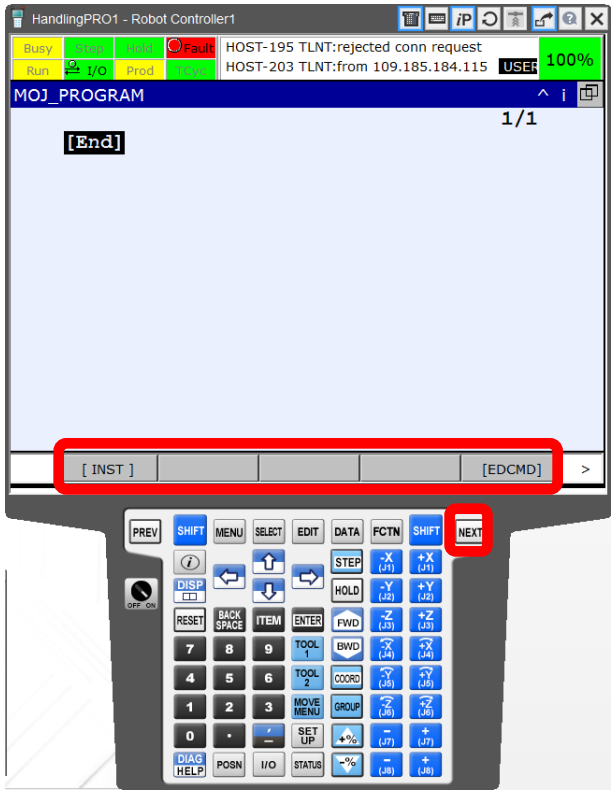
POSTOPEK UČENJA / PROGRAMIRANJA NA UE (TEACH PENDANT)

Learner Centric Advanced Manufacturing Platform

4. V programu uporabite TF.
   - NEXT (preklopite menijsko vrstico) / F1 [INST]

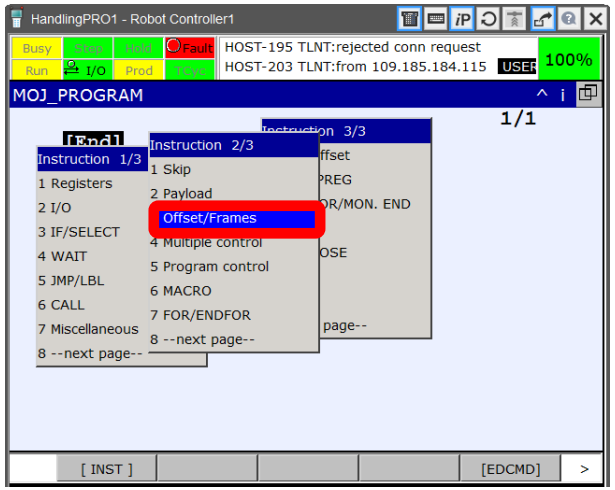


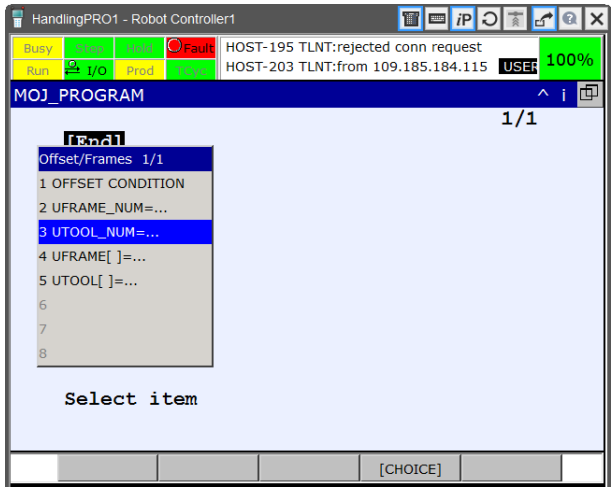


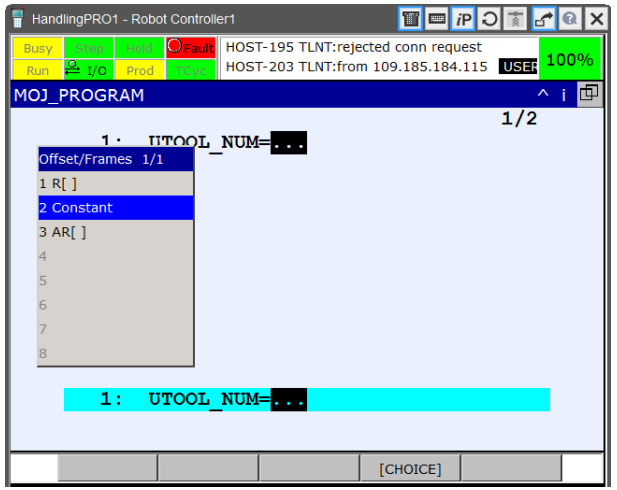


POSTOPEK UČENJA / PROGRAMIRANJA NA UE (TEACH PENDANT)

- Offset/Frames + ENTER
- UTOOL_NUM=… + ENTER
- Constant + ENTER
- vpišete št. TF + ENTER



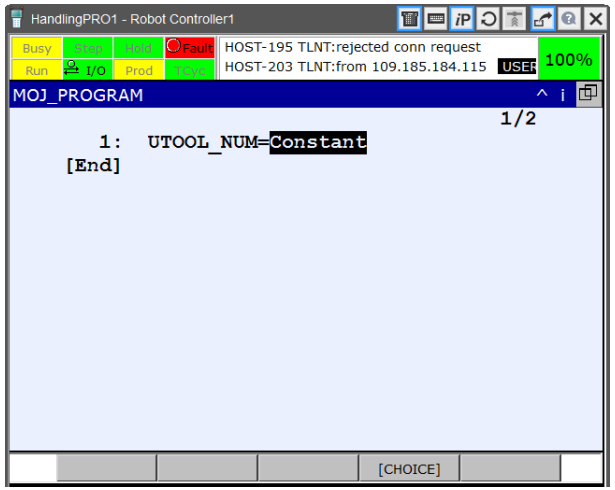


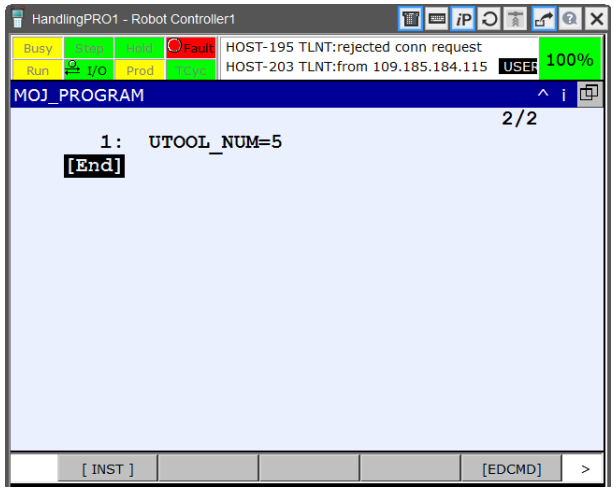


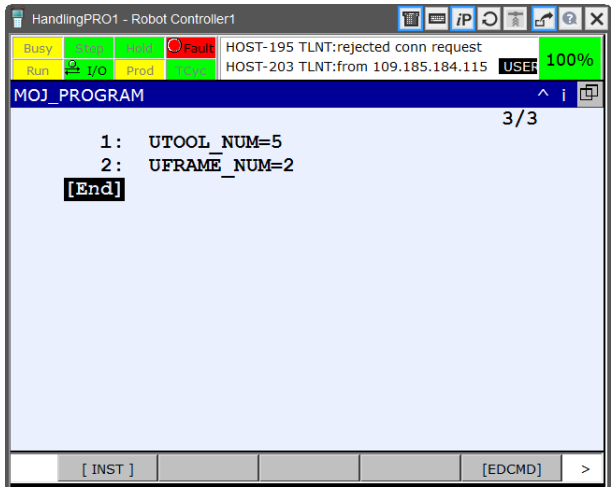
POSTOPEK UČENJA / PROGRAMIRANJA NA UE (TEACH PENDANT)

Learner Centric Advanced Manufacturing Platform

5. V programu uporabite UF
   - enak postopek kot v tč. 4, le da izberete UFRAME_NUM=…
6. V programu določite druge nastavitve oz. začetne vrednosti (payload, override, postavitev registrov …), če je potrebno.



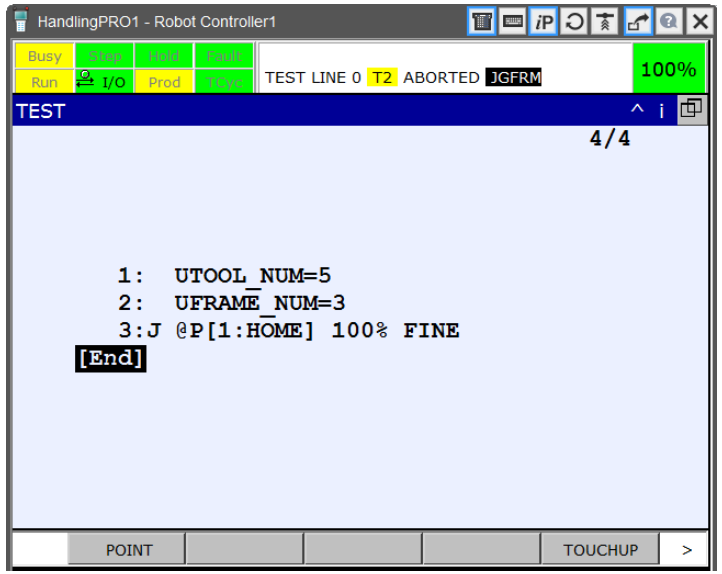


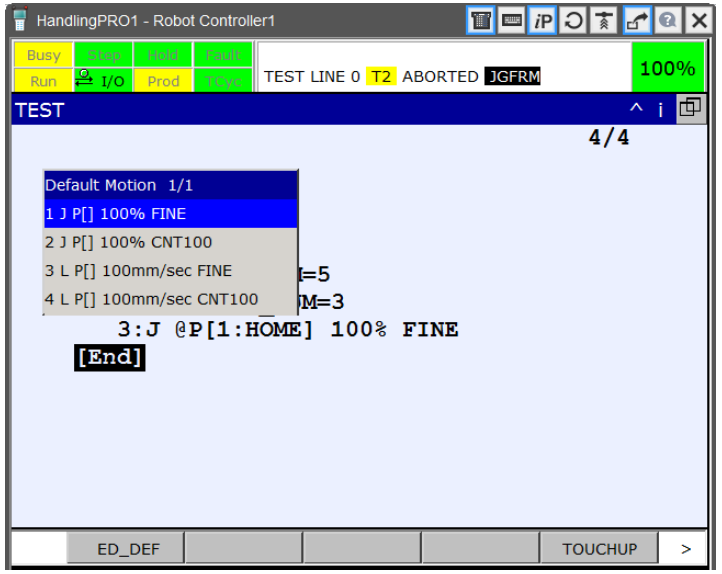


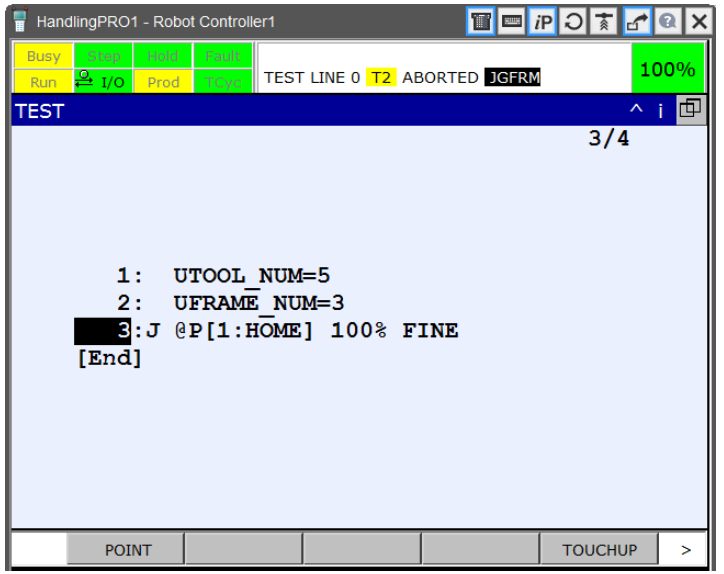
POSTOPEK UČENJA / PROGRAMIRANJA NA UE (TEACH PENDANT)

Learner Centric Advanced Manufacturing Platform

7. Orodje (TCP) postavite na želeno lokacijo in jo shranite (ponavljajte za vse točke) ter nastavite potrebne parametre:
   - SHIFT + F1 POINT ali
   - F1 POINT → izberite ustrezni ukaz za gibanje + ENTER.
   - SHIFT + F5 TOUCHUP, že shranjeni lokaciji popravimo koordinate (X, Y, Z, W, P, R, CONF oz. J1, J2, J3, J4, J5, J6) in UT ter UF.

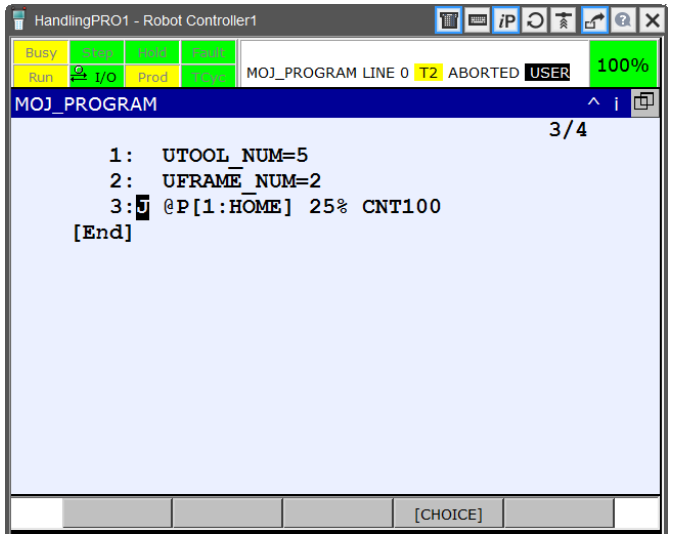


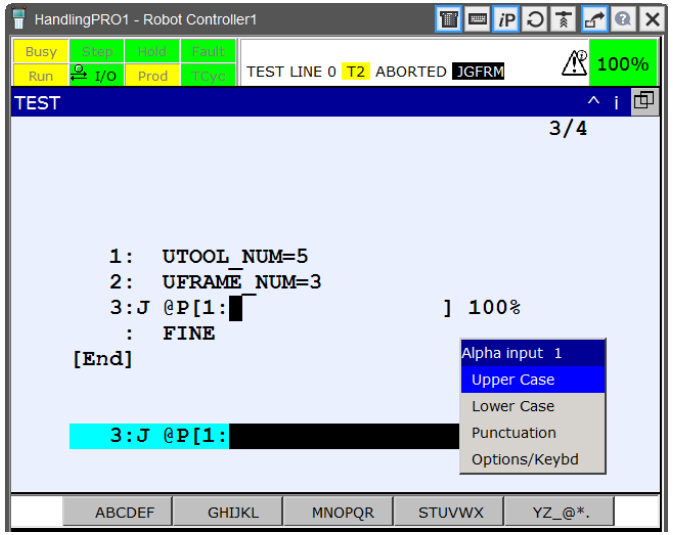


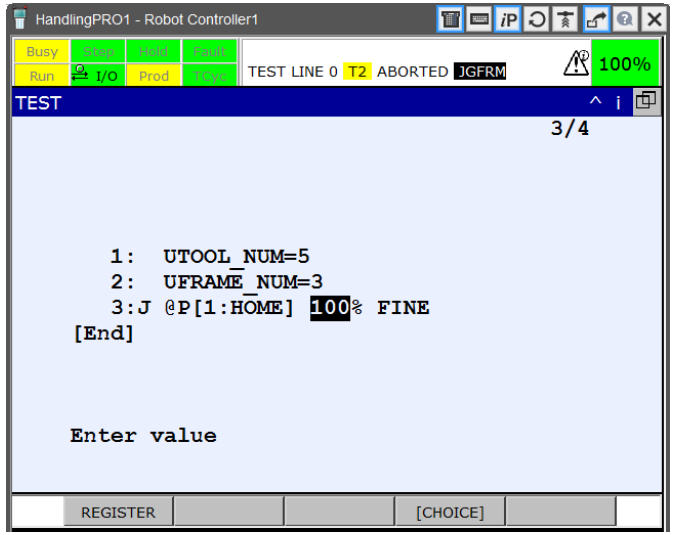


POSTOPEK UČENJA / PROGRAMIRANJA NA UE (TEACH PENDANT)

- nastavite način gibanja, F4 [CHOICE] (J, L, C)
- vpišete ime točke, P[1], ENTER, vpišete ime, ENTER in/ali ročno popravite koordinate
- vpišete hitrost v % (J) oz. v mm/s (L in C)
- nastavite parameter FINE / CNT (F4 [CHOICE]) in/ali , ACC, AP_LD, RT_LD, CR …
- po potrebi uporabite druge ukaze: LBL/JMP LBL, IF/SELECT, FOR/ENDFOR, WAIT, CALL …







POSTOPEK UČENJA / PROGRAMIRANJA NA UE (TEACH PENDANT)

Learner Centric Advanced Manufacturing Platform

8. Testirajte program v T1 v koračnem načinu (STEP), počasna hitrost in po potrebi odpravljajte napake.
9. Ko vse napake odpravite in program deluje po navodilih, ga testirajte še v T1 v kontinuiranem načinu, počasna/srednja/visoka hitrost. Če ugotovite napake, jih odpravite in ponovite točki 8 in 9.
10. Testirajte program v T2 v kontinuiranem načinu, počasna/srednja/visoka hitrost. Če ugotovite napake, jih odpravite in ponovite testiranje (T1 -> T2, tč. 8, 9 in 10), dokler program ne deluje po zahtevah.
11. Program lahko zaženete v avtomatskem režimu.

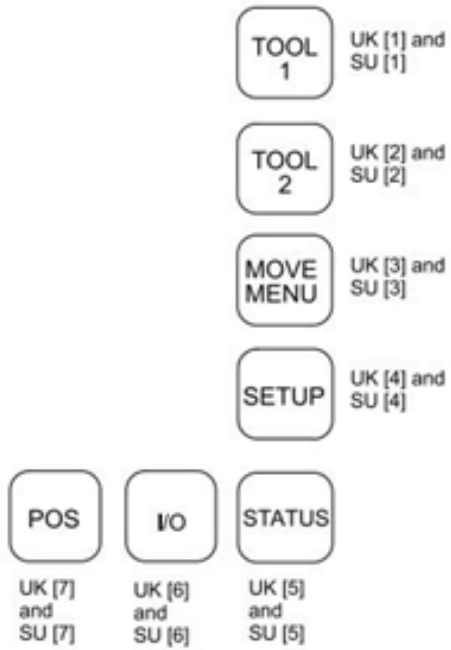
ROČNO AKTIVIRANJE PRIJEMALA

- SHIFT + TOOL 1 (bela tipka, desno od 9)

SHIFT + TOOL 1 – prijemalo odprto/zaprto

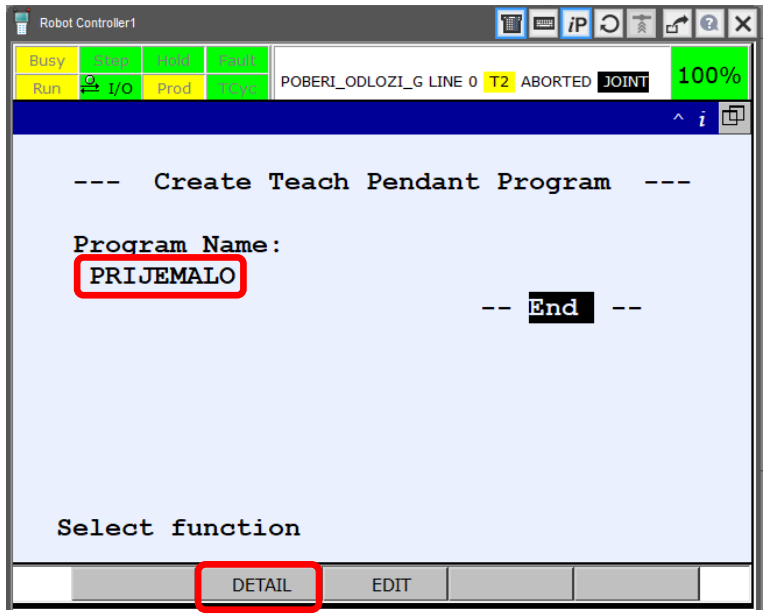
NASTAVITEV ROČNEGA AKTIVIRANJA PRIJEMALA

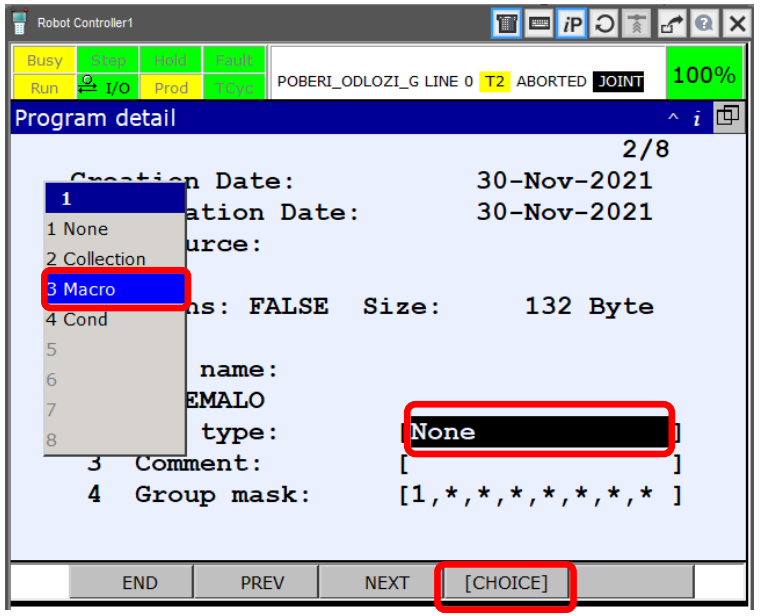
- izdelati makro
  - makro je podoben robotskemu programu, le da ne vsebuje ukazov za gibanje
- makro lahko priredimo uporabniškim tipkam (bele nepopisane tipke)

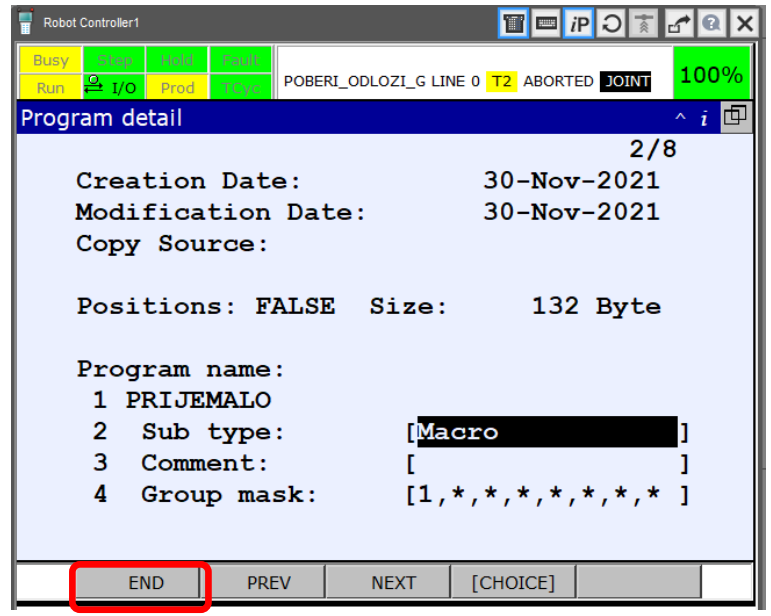


NASTAVITEV ROČNEGA AKTIVIRANJA PRIJEMALA

- izdelava makra (podobno kot robotski program):
  - na UE kliknite F2 CREATE
  - vpišite ime makra in potrdite z ENTER
  - kliknite F2 DETAIL in v izbiri Sub type: kliknite F4 [CHOICE]
  - v meniju izberite 3 Macro ter potrdite s F1 END







NASTAVITEV ROČNEGA AKTIVIRANJA PRIJEMALA

- v programskem urejevalniku napišite ustrezno kodo za aktivacijo prijemala
- primer:
  - HAND_OPEN – makro za odpiranje prijemala
  - HAND_CLOSE – makro za zapiranje prijemala
  - HAND_TOG – makro za preklapljanje prijemala odprto/zaprto

HAND_OPEN 1/4

```
  1:  LBL[1]
  2:  RO[7:Open Gripper]=ON
  3:  IF RO[8:Close Gripper]<>OFF,
   :  JMP LBL[1]
[End]
```

HAND_CLOSE 1/5

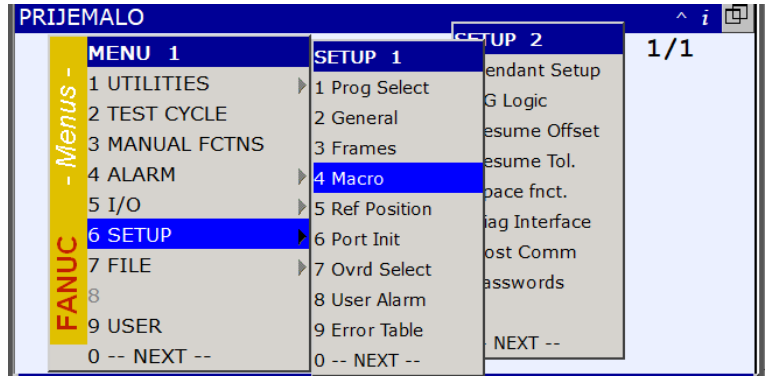
```
  1:  LBL[1]
  2:  RO[7:Open Gripper]=OFF
  3:  IF RO[8:Close Gripper]<>ON,
   :  JMP LBL[1]
  4:  WAIT    .10(sec)
[End]
```

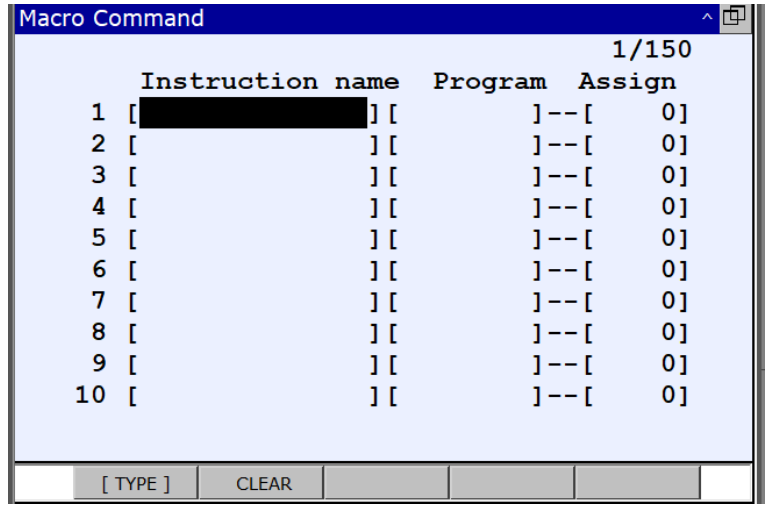
HAND_TOG 1/8

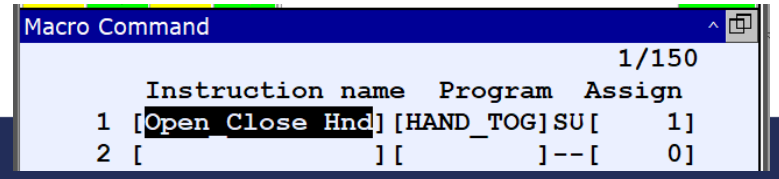
```
  1:  !Toggle Hand Open/Close
  2:  IF RO[7:Open Gripper]=ON,
   :  JMP LBL[1]
  3:  RO[7:Open Gripper]=ON
  4:  JMP LBL[2]
  5:  LBL[1]
  6:  RO[7:Open Gripper]=OFF
  7:  LBL[2]
[End]
```

NASTAVITEV ROČNEGA AKTIVIRANJA PRIJEMALA

- prirejanje makra uporabniškim tipkam
  - na UE kliknite MENU/6 SETUP/4 Macro
  - v polju Instruction name kliknite ENTER in vpišite ime ukaza ter potrdite z ENTER
  - v polju Program kliknite F4 CHOICE in poiščite prej izdelani makro
  - v polju - - kliknite F4 CHOICE in izberite 2 UK ali 3 SU
    - UK – User Key
    - SU – SHIFT + User Key
  - v polju Assign vpišite številko uporabniške tipke (UK ali SU)
- s kliki na UK oz. SU testirajte aktivacijo prijemala

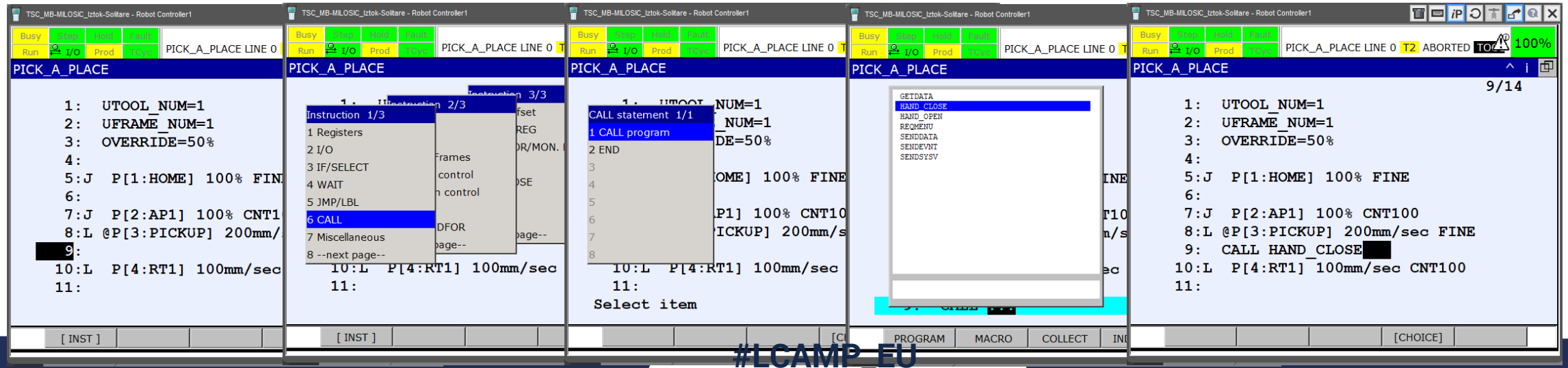






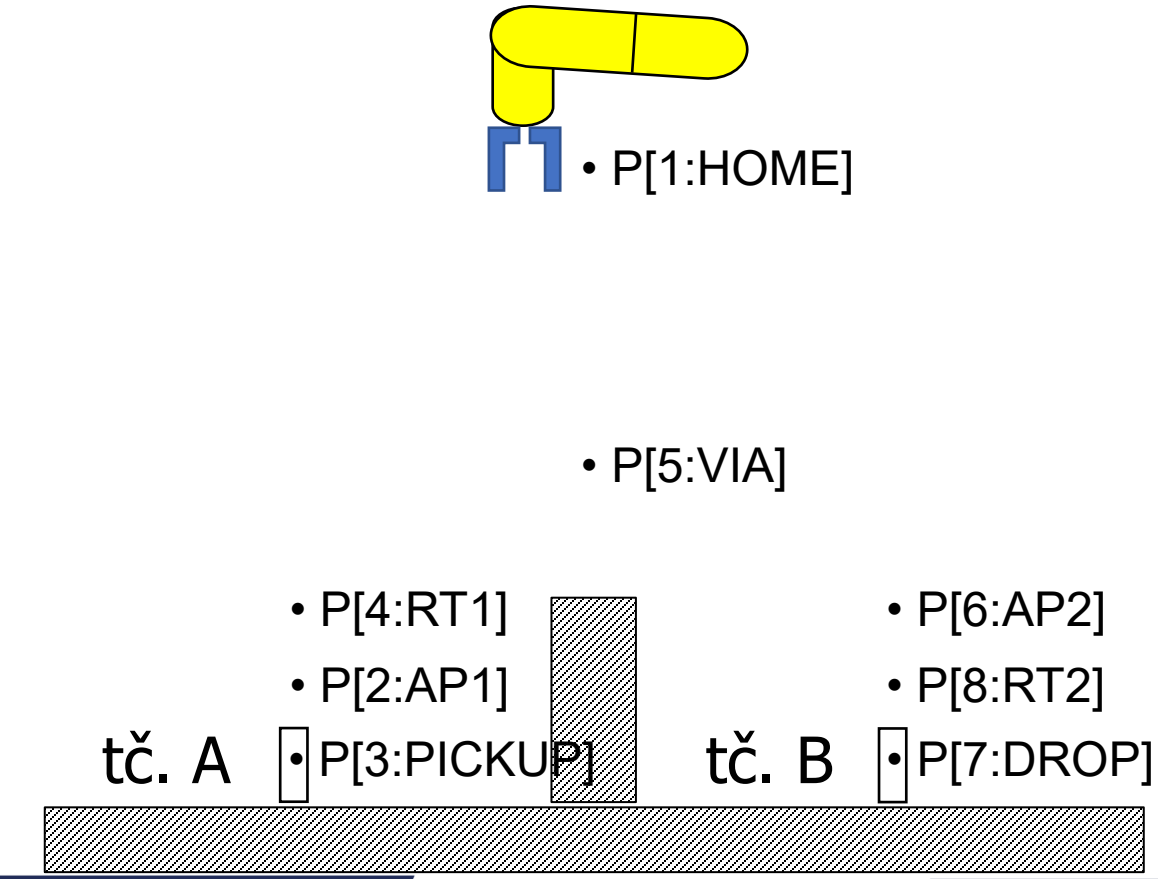
PROGRAMSKO AKTIVIRANJE PRIJEMALA

- v programu aktiviramo prijemalo s klicanjem podprograma v ustrezni programski vrstici
  - kliknite F1 [INST] in izberite 6 CALL/1 CALL program
  - kliknite F2 MACRO in poiščite ustrezni makro ter potrdite z ENTER



SMERNICE PRI PROGRAMIRANJU

- smiselno načrtujemo točke oz. lokacije TCP:



SMERNICE PRI PROGRAMIRANJU

- **smiselno načrtujemo točke oz. lokacije TCP**:
  - **izhodiščno točko** (HOME) **P[1]**
    - v tej točki TCP prične in se na koncu v njo vrne oz. čaka na naslednji ukaz
    - je primerno blizu in hkrati dovolj oddaljena, da robot ne moti ostalih aktivnosti
  - **ciljne (končne) točke** (PICKUP, DROP, WELD …) **P[3] in P[6]**
    - točke obratovanja (prijemanje / odlaganje, varjenje, lepljenje …)
  - **predtočke** (APPROACH) **P[2] in točke vračanja (**RETREAT) **P[5]**
    - so pravokotne na ciljne točke in
    - v njih je orodje ustrezno orientirano
  - **vmesne točke** (VIA) **P[4]**
    - za izogibanje oviram
    - za ustreznejšo gibanje TCP
  - točke **komentiramo – smiselno poimenujemo**

SMERNICE PRI PROGRAMIRANJU

- na **začetku programa vedno**:
  - nastavimo KS (TF in UF),
  - nastavimo obremenitve – PAYLOAD
  - nastavimo hitrost, OVERRIDE
  - postavimo začetne vrednosti, deklariramo spremenljivke (registre) …
- zaradi **boljše preglednosti** programa med posameznimi deli programa **puščamo prazne vrstice**
- **program smiselno komentiramo**
- **daljše programe razdelimo na** manjše, smiselne **podprograme**

SMERNICE PRI PROGRAMIRANJU

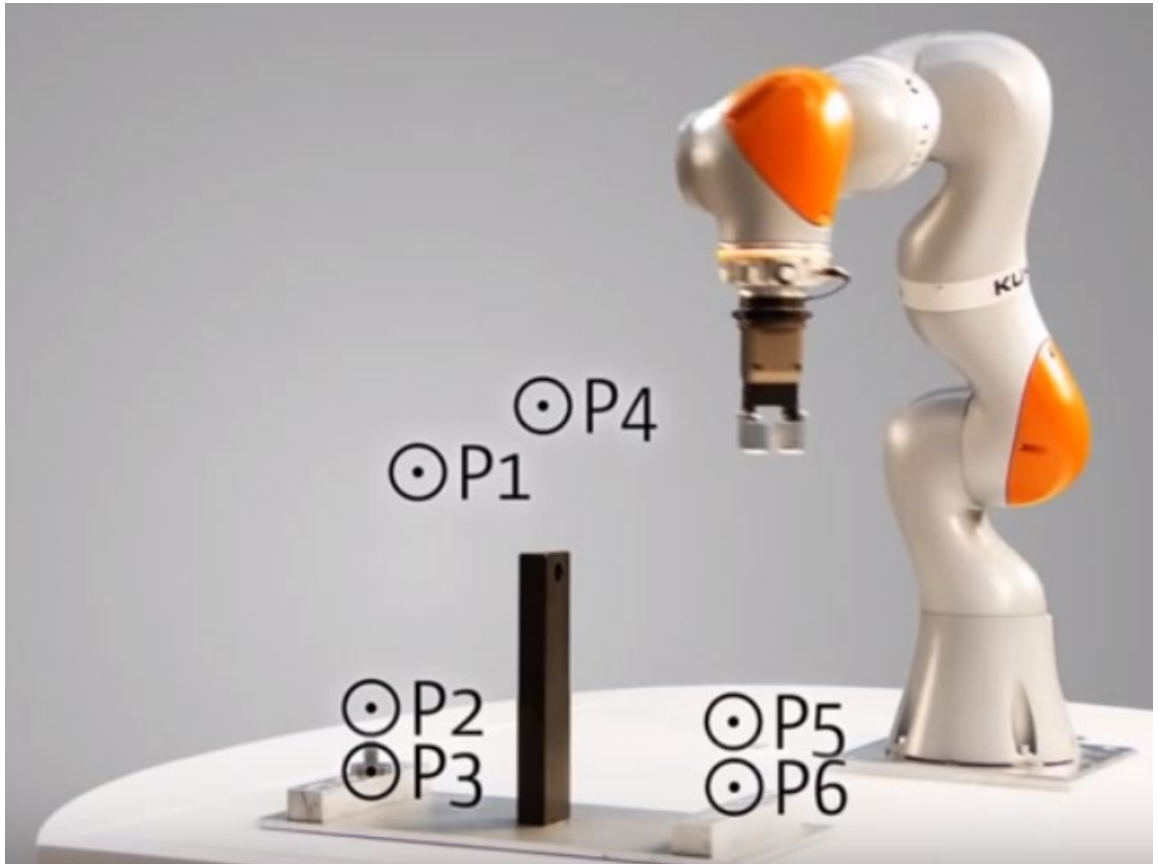
- uporabljamo **proste gibe (J)** – gibanje osi, za hitrejše delovanje in **krajše časovne cikle**
- linearni in krožni gibi (L, C) so počasnejši; uporabljamo jih tam, kjer mora TCP potovati po točno predvideni poti – gibanje po poti (prijemanje/odlaganje, varjenje, lepljenje …)
- predtočke (AP) in točke vračanja (RT) postavljamo čim bližje ciljnim točkam (PICKUP, DROP, WELD …)
- parameter FINE uporabljamo samo v ciljnih točkah (PICKUP, DROP, WELD …), v vmesnih točkah (VIA) uporabljamo parameter CNT

OPTIMIZACIJA ČASA CIKLA

- ustrezni gibi
  - uporabljajte čim več prostih (J) gibov, razen tam, kjer morate uporabiti L ali C gibe (gibanje TCP po poti)
- premikanje točk
  - izhodiščna pozicija HOME naj bo čim bliže delovnemu prostoru, vendar dovolj daleč, da ne moti ostalih aktivnosti
  - počasne gibe (L in C) izvajajte na čim krajših razdaljah – premik točk AP, RT čim bliže pobiranju, odlaganju, lepljenju, varjenju …
- hitrost giba
  - uporabljajte čim višje hitrosti gibov, kot to prenese tehnološki postopek
- nastavitev FINE/CNT
  - uporabljajte čim več in čim višje vrednosti CNT parametra, razen tam, kjer se mora TCP zagotovo ustaviti FINE (pobiranje, odlaganje, pričetek/konec lepljenja, varjenja …)
- nastavitev ACC
  - ustrezno povečajte/zmanjšajte pospešek in pojemek gibanja robota

UČENJE / PROGRAMIRANJE

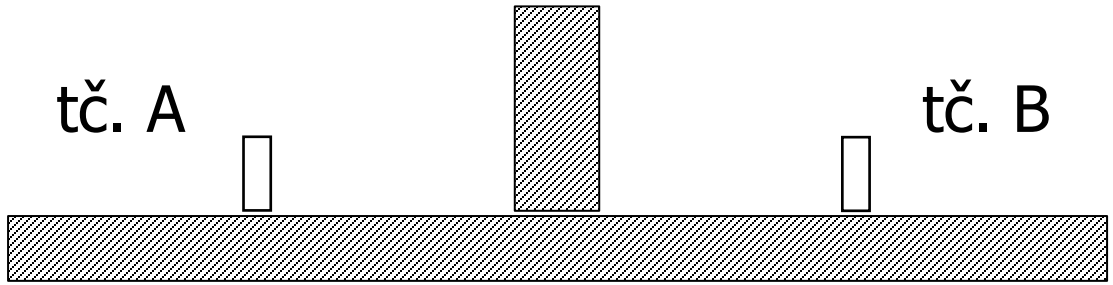
- primer učenja (programiranje) sodelujočega robota



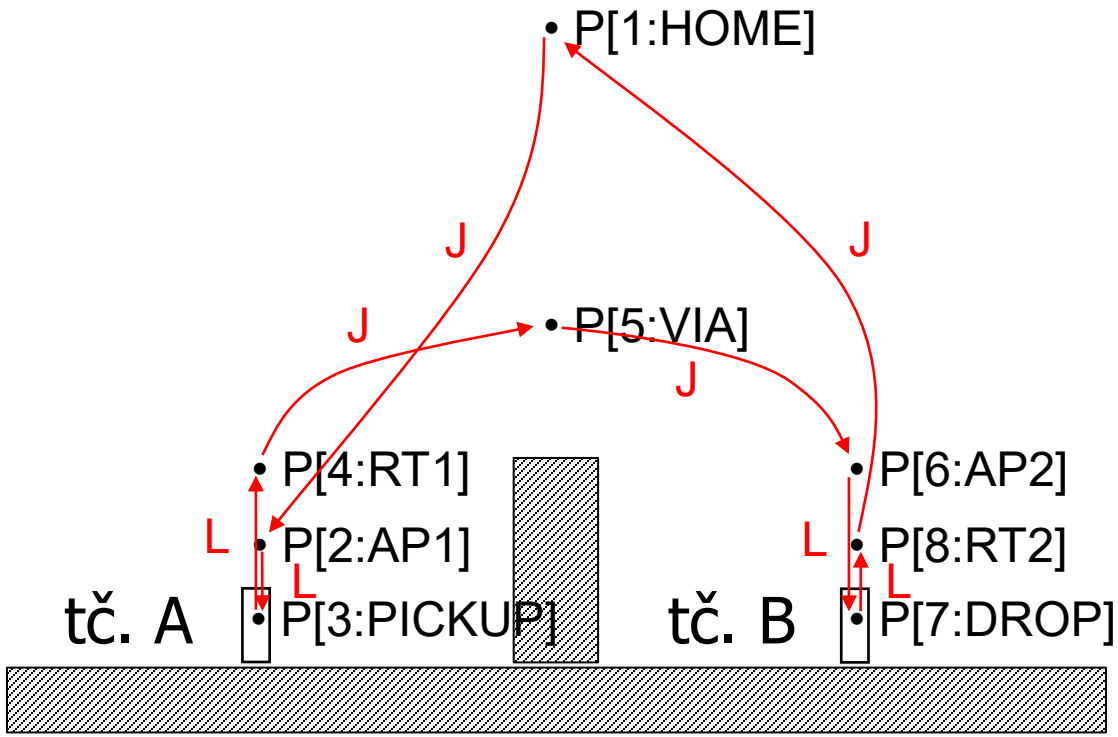
https://www.youtube.com/watch?v=r7gU74Yv9Es

PRIMER

- Napišite program, ki bo prestavil predmet iz točke A preko ovire na točko B, kot je prikazano na sliki. Upoštevajte smernice pri programiranju,
  - uporabite TF 1 in UF 1 ter generalno hitrost (OVERRIDE) 50 %
  - narišite in smiselno poimenujte točke,
  - narišite poti gibanja in jih označite ter pojasnite,
  - pri terminaciji CNT uporabite vrednost 50,
  - pri J gibih uporabite hitrosti 50 %
  - pri L gibih 200 mm/s, če je prijemalo prazno in 100 mm/s, če je prijemalo polno



PRIMER


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
tč. A
P[3:PICKUP]
tč. B
P[7:DROP]

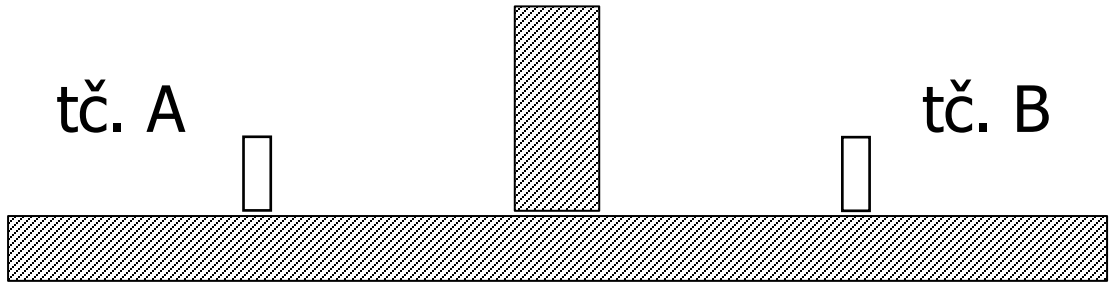
PONOVITE

Learner Centric Advanced Manufacturing Platform

1. Opišite poenostavljen postopek za programiranje robota na učni enoti.
2. Pojasnite smernice pri programiranju robota?
3. Kako optimiziramo čas cikla programa?
4. Kako aktiviramo prijemalo:
   - ročno,
   - v programu?
5. Kakšen je postopek nastavitve za ročno aktiviranje prijemala?

PONOVITE

6. Napišite program, ki bo prestavil predmet iz točke A preko ovire na točko B, kot je prikazano na sliki
   - upoštevajte smernice pri programiranju
   - uporabite TF 1 in UF 2
   - narišite in poimenujte točke
   - narišite poti gibanja in pojasnite gibe
   - pri J gibih uporabite hitrosti 50 %
   - pri L gibih 200 mm/s, če je prijemalo prazno in 100 mm/s, če je prijemalo polno



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.