Laborator 1

Prezentarea sistemului

Celula flexibilă de fabricație

- 5 posturi de lucru
- Bandă conveioare
- Automat programabil

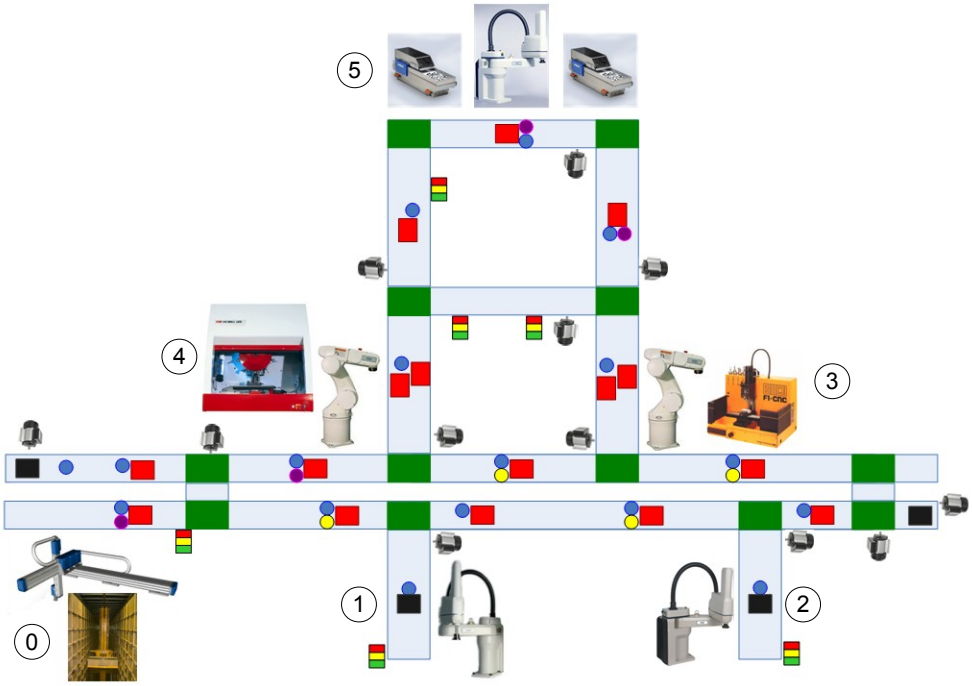

Figura 1.1: *Posturile de lucru*

Posturile de lucru:

Postul 1: Asamblare verticală

- Robot SCARA - Adept Cobra 600TT
- Vedere 2D: AdeptVision
- O cameră fixă și una mobilă

Postul 2: Asamblare verticală

- Robot SCARA - Adept Cobra s600
- Vedere 2D: AdeptSight
- O cameră fixă și una mobilă

Postul 3: Asamblare și prelucrare prin așchiere

- Robot articulat vertical - Adept Viper s650
- Emco F1 CNC - 3 axe
- Vedere 2D: AdeptSight
- O cameră fixă și una mobilă

Postul 4: Asamblare, prelucrare prin așchiere, inginerie inversă

- Robot articulat vertical - Adept Viper s650
- Emco ConceptMill 105 CNC - 4 axe
- Vedere 2D: AdeptSight
- O cameră fixă și una mobilă
- Vedere 3D: Scanner laser montat pe efectorul terminal al robotului

Postul 5: Alimentare cu piese

- Adept Cobra s800
- 2 x Adept AnyFeeder
- Vedere 2D: AdeptSight
- Trei camere fixe

Roboții industriali Adept

Roboții sunt comandați de SmartController, pe care rulează sistemul de operare de timp real V^+ . Programele utilizator pot fi scrise în limbajul V^+ .

Posibilități de interfațare cu alte echipamente:

- Comunicație TCP/IP (Ethernet)
- Digital I/O
- Comunicație RS232, RS485, DeviceNet

Roboții SCARA - Adept Cobra

SCARA = *Selective Compliant Assembly (Articulated) Robot Arm.*

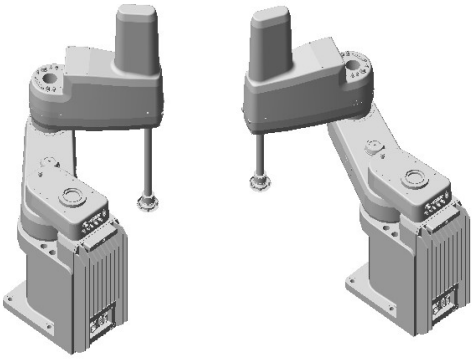

Figura 1.2: *Robot SCARA în configurație LEFTY și RIGHTY*

Configurații posibile

- LEFTY / RIGHTY
- SINGLE / MULTIPLE

LEFTY și RIGHTY se referă la configurația articulației 2.

SINGLE limitează mișcarea ultimei articulații la intervalul ±180 °, pentru următoarea mișcare. MULTIPLE permite mișcarea ultimei articulații pe tot domeniul posibil (±360 °).

Roboții articulați vertical - Adept Viper

Roboți cu 6 grade de libertate.

Configurații posibile

- LEFTY / RIGHTY
- ABOVE / BELOW
- FLIP / NOFLIP
- SINGLE / MULTIPLE

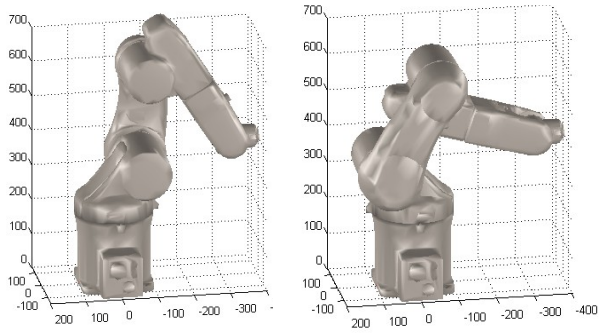

Figura 1.3: ***LEFTY***/*ABOVE/NOFLIP* *și* ***RIGHTY***/*ABOVE/NOFLIP*

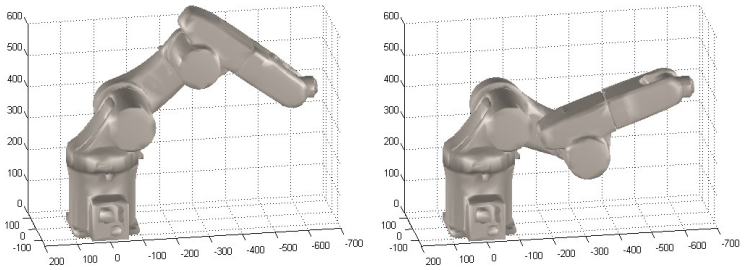

Figura 1.4: *LEFTY/****ABOVE***/*NOFLIP* *și LEFTY/****BELOW***/*NOFLIP*

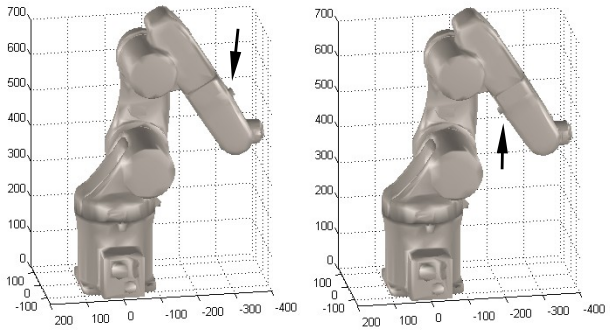

Figura 1.5: *LEFTY/ABOVE/****FLIP*** *și LEFTY/ABOVE/****NOFLIP***

Vedere artificială 2D

Indecision is the key to flexibility

AdeptVision - achiziția și prelucrarea imaginii se realizează pe controller-ul robot, iar configurarea şi operaţiile de vedere se fac din linia de comandă.

AdeptSight - achiziția și prelucrarea imaginii se desfăşoară pe PC. Controllerul robot se conectează la serverul AdeptSight prin TCP/IP și poate iniția operațiile de vedere și obține rezultatul acestora prin instrucțiuni V^+ . Operațiile de vedere se definesc folosind o interfață grafică.

Camera fixă: poziția acesteia față de robot nu se modifică.

Camera mobilă poate fi montată pe:

- efectorul terminal al robotului
- unul din segmentele (link-urile) robotului
  - Link 2 - pentru robotul Cobra
  - Link 3 - pentru robotul Viper

Iluminare

- lumina ambiantă: variază pe parcursul zilei, și are dezavantajul că pe imagine apar umbrele obiectului, ceea ce face dificilă recunoașterea și localizarea exactă.
- iluminarea de dedesubt (backlighting): este facilitată recunoașterea conturului pieselor; imaginile au contrast ridicat, fără umbre, iar diferențierea între piese și fundal se poate realiza prin binarizarea imaginii. Analiza suprafeței pieselor este dificilă cu acest tip de iluminare.
- lumină structurată: permite extragerea trăsăturilor 3D din imaginile 2D prin analiza unor șabloane de lumini și umbre proiectate pe obiectul de interes.

Vedere artificială 3D

Postul de lucru nr. 4 dispune de un scanner 3D, montat pe gripper-ul robotului, care proiectează un fascicul de raze laser coplanare pe obiectul analizat, făcând posibilă extragerea conturului piesei în planul laser prin

analiza imaginilor de la două camere video. Robotul poate baleia scanner-ul laser în jurul unui obiect existent, permițând reconstrucția modelului 3D al acestuia.

Recunoașterea 3D se face prin procedeul de triangulație, iar tehnica de iluminare laser este un caz particular de lumină structurată.

Datele obținute de la scanner-ul 3D reprezintă un *nor de puncte*, iar prin postprocesare se poate obține un model matematic al suprafeței (triangular mesh, NURBS etc). Modelul 3D astfel obținut poate fi utilizat în programe CAD/CAM sau în software de simulare și animație.

Scanner-ul laser poate fi folosit și ca senzor de distanță, de exemplu identificarea numărului de piese dintr-o stivă, calibrarea unui sistem de referință pentru un plan înclinat, recunoașterea obiectelor 3D în vederea manipulării, alinierea precisă a robotului față de repere a căror poziție este cunoscută doar aproximativ etc.

Alimentare cu piese

Alimentarea cu piese a posturilor de lucru 1-4 se realizează cu ajutorul celor două dispozitive AnyFeeder și al robotului Cobra s800. Operatorul pune piesele vrac în *Bulk Container*, iar AnyFeeder-ul le vibrează, permițând robotului să recunoască piesele folosind vederea și să le preia de pe *Feed Surface*.

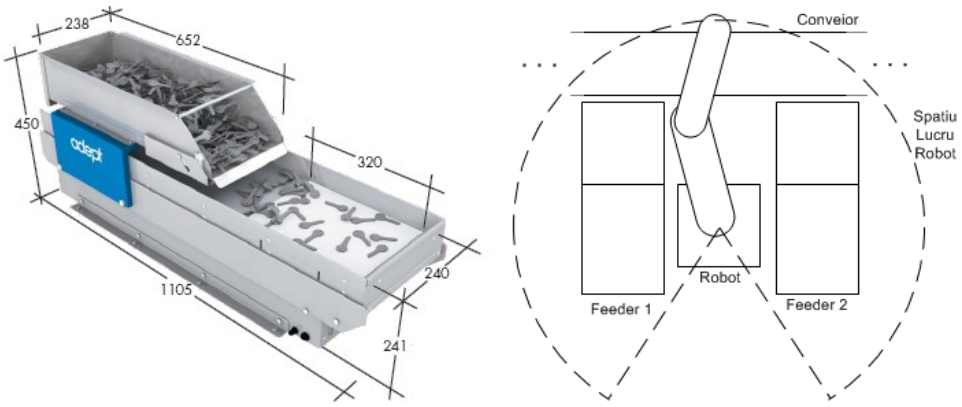

Figura 1.6: *Adept AnyFeeder (a) și structura postului de alimentare (b)*

Comenzi AnyFeeder:

- *Dispense / Heavy Dispense* - aduce piese din Bulk Container pe Feed Surface;
- *Flip* - întoarce piesele de pe Feed Surface pe partea cealaltă;

- *Feed Forward / Backward* - deplasează piesele de pe Feed Surface mai în față sau mai în spate;
- *Flip + Feed Forward / Backward* - combină mișcările Flip și Feed Forward / Backward
- *Purge* elimină piesele aflate pe Feed Surface
- *Stop* oprește mișcarea curentă (dacă există) și aduce Feed Surface în poziție orizontală.

Piesele aflate pe suprafața de alimentare (Feed Surface) sunt iluminate de dedesubt (Backlighting).

Prelucrare prin așchiere

> *When you get to the point where you really understand your CNC, it is obsolete.*

Acest proces permite realizarea pieselor mecanice pornind de la semifabricate, prin îndepărtarea materialului. Celula de fabricație dispune de două mașini de prelucrare prin așchiere, care pot efectua operații de frezare, găurire și filetare.

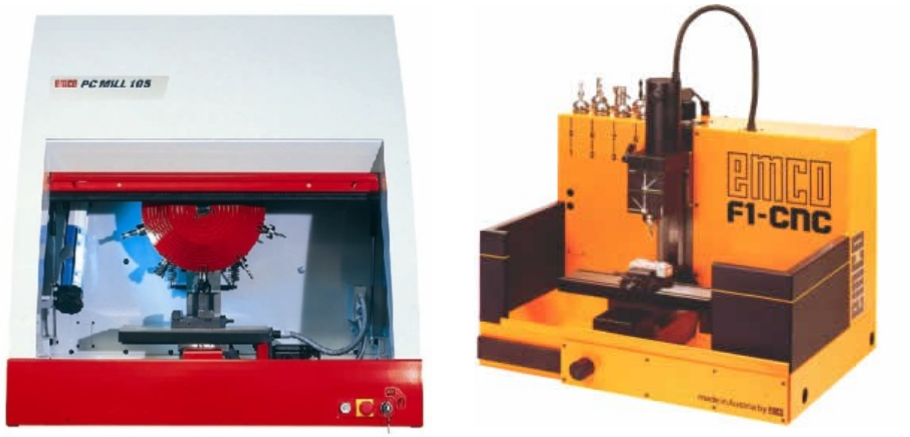

Figura 1.7: *EMCO ConceptMill 105 și Emco F1 CNC*

Mașinile CNC se programează folosind limbajul DIN 66025, cunoscut sub numele de G-Code. Acest cod poate fi scris manual pentru prelucrările

foarte simple, dar în cele mai multe cazuri el este generat automat de un program CAM (Computer Aided Manufacturing) pornind de la un fișier CAD (Computer Aided Design). Fișierul CAD poate fi o schiță 2D sau model 3D.

Mașina EMCO F1 CNC este un model educațional fabricat la începutul anilor '90 și se poate deplasa pe cele 3 axe carteziene. Este destinată prelucrării metalelor, însă poate prelucra și materiale plastice. Caracteristici principale:

- 3 axe carteziene
- menghină automată, acționată electric
- schimbător manual de scule
- ușă manuală, cu senzor
- interfață cu robotul

Mașina EMCO ConceptMill 105 CNC este un model educațional recent, cu următoarele caracteristici:

- 3 axe carteziene
- o axă rotativă
- menghină automată, acționată pneumatic
- magazie de scule cu 10 poziții, cu schimbător automat, pneumatic
- ușă automată (deschiderea și închiderea este acționată pneumatic)
- interfață cu robotul

Robotul care deservește mașina poate realiza alimentarea cu semifabricate și descărcarea pieselor prelucrate și poate controla operațiile efectuate de CNC (închiderea/deschiderea ușii, fixarea piesei în menghină, încărcarea și execuția programelor). Astfel devine posibilă automatizarea unei producții, aceasta putând rula fără a fi necesară intervenția operatorului uman.

Controlul celulei

Posturile de lucru pot lucra fie independent, fie sub supervizarea automatului programabil care deservește celula.

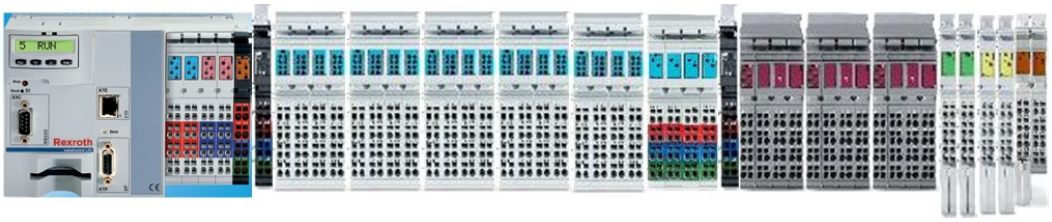

Figura 1.8: *Automatul programabil*

Conexiunile dintre automat și celula de fabricație:

- Stațiile de lucru PC (Ethernet);
- Controller-ele robot (Ethernet + Digital I/O);
- Senzori pentru detecția portpaletelor (Digital Input);
- Senzori pentru identificarea portpaletelor (ProfiBus);
- Elemente mecanice: motoare conveior, opritoare (Digital I/O);
- Coloanele de lumini (Digital Output).

Elemente de interfață cu utilizatorul

Pornirea robotului

Se execută următorii pași, în ordine:

1. Se alimentează modulul High Power (220V sau 380V)
2. Se alimentează controller-ul robot (24V)
3. Se așteaptă până când controller-ul robotului bootează
4. Se pornește aplicația AdeptWindows PC și se conectează la robot prin Ethernet
5. Se activează High Power
   - (varianta a) Se introduce de la consolă comanda `.enable power`
     sau
     (varianta b) Se acționează butonul de pornire de pe MCP (I)
   - Se confirmă acțiunea prin apăsarea butonului de pe Front Panel.
6. Doar pentru roboții COBRA: se execută rutina de calibrare a articulațiilor.

Oprirea robotului

1. Se dezactivează High Power (`.disable power`)
2. Se deconectează aplicațiile PC conectate la robot (AdeptWindows, AdeptDesktop, AdeptSight etc)
3. Se salvează programele din memoria robotului pe disc
4. Se oprește alimentarea controller-ului (24V)
5. Se oprește alimentarea High Power (220V / 380V)

Calibrarea articulațiilor

Pentru a putea ști în ce poziție se află robotul, fiecare articulație dispune de un encoder. Acest encoder este de tip incremental (de obicei în cuadratură). În momentul în care o articulație se deplasează, encoderul indică sensul și distanța parcursă folosind impulsuri. Controller-ul robot numără impulsurile și poate calcula poziția fiecărei articulații adunând numărul de impulsuri (scalat corespunzător) la o poziție de referință.

În momentul pornirii robotului, controller-ul nu cunoaște poziția inițială a articulațiilor. Prin procedura de calibrare a articulațiilor, numită și *calibrare electromecanică*, se determină această poziție de referință, folosind pe lângă encoder, limitatori fizici (mecanici, optici, magnetici etc.) la capătul cursei sau din loc în loc în interiorul cursei. Procedura de calibrare poate include de asemenea verificarea funcționării limitatoarelor, verificarea conexiunilor electrice la motoare, precum și determinarea parametrilor dinamici ai robotului (momente de inerție, frecări etc.) pentru acordarea buclelor de reglare.

Mașinile cu comandă numerică folosite în laborator sunt acționate de motoare pas cu pas, și lucrează în buclă deschisă, fără a folosi encodere. Mașina EMCO F1 nu dispune de rutină de calibrare, acest lucru fiind lăsat în sarcina utilizatorului, care trebuie să stabilească manual originea sistemului de coordonate. Mașina EMCO ConceptMill 105 include un procedeu de calibrare, numit *referențiere* (comanda Reference Machine), prin care se determină poziția celor 4 axe și poziția mecanismului de schimbare a sculei folosind câte un limitator fizic la unul din capetele fiecărei axe pentru a stabili originea.

Limitatorii electromecanici sunt folosiți și pentru protecție, în cazul depășirii cursei maxime admise. După calibrare se folosesc și limite software, cu același rol. Atingerea limitatorilor fizici poate indica o funcționare defectuoasă a software-ului de control, rezultatul unei calibrări incorecte sau existența unor suprasolicitări mecanice.

Procedura de calibrare a roboților

Roboții Viper execută automat procedura de calibrare la pornirea sistemului.

Pentru roboții Cobra s600 și s800, utilizatorul trebuie să ruleze manual rutina de calibrare după activarea High Power. În timpul calibrării, robotul nu se mișcă deloc.

Pentru robotul Cobra 600TT, rutina de calibrare este rulată de asemenea de către utilizator după activarea High Power. În timpul calibrării, robotul se va mișca; de aceea, înaintea calibrării, utilizatorul trebuie să se asigure că robotul nu se va lovi de echipamentele învecinate (în special de robotul Cobra s600).

Calibrarea din consola AdeptWindows:

```
.enable power
.calibrate
```

Calibrarea din MCP: CMD → Calib.

Elemente de siguranță

Roboții industriali sunt pe cât de utili, pe atât de periculoși. Spre deosebire de programele obișnuite care rulează pe PC, în cazul roboților industriali o greșeală de programare sau de operare poate avea urmări foarte grave, cum ar fi:

- accidentarea operatorului sau a persoanelor din jur
- avarierea robotului, distrugerea pieselor sau a echipamentelor din apropiere
- avarierea cablurilor electrice, putând duce la scurtcircuit

Roboții sunt echipați cu mecanisme de protecție, însă acestea sunt efective atâta timp cât operatorul le folosește corect. O clipă de neatenție vă poate costa viața!

Din aceste motive, orice program robot se va rula mai întâi cu viteză redusă, operatorul fiind pregătit să acționeze butonul de avarie în cazul în care robotul este pe cale să producă o coliziune. Rularea programului la viteze mai ridicate se va face numai după ce operatorul s-a asigurat că programul nu conține erori, iar creșterea vitezei se va face gradual. Atenție: dacă programul se comportă în regulă la o rulare (cu viteză redusă), acest lucru **nu** înseamnă că este lipsit de erori!

Roboții sunt prevăzuți cu următoarele mecanisme de siguranță:

- Ciupercă de avarie (E-Stop), atât pe MCP, cât și pe panoul frontal

- Întreruperea automată a alimentării motoarelor în cazul detectării unei suprasarcini (nu ajută dacă robotul se deplasează cu viteză mare)
- Posibilitatea limitării vitezei maxime de lucru
- Confirmare la activarea High Power
- Mişcarea robotului din programele V^+ se poate face doar în modul Comp selectat din MCP

Utilizarea MCP

MCP-ul (Manual Control Pendant) permite realizarea următoarelor funcții:

- Controlul robotului prin activarea/dezactivarea alimentării; comanda manuală a robotului;
- Învățarea locațiilor (punctelor) pentru robot;
- Afișarea poziției curente a robotului (World sau Joint), a punctelor învățate, vizualizarea semnalelor de I/O și a mesajelor sistem;
- Pornirea și oprirea aplicațiilor;
- Afișarea și editarea variabilelor.

Comanda manuală a robotului

Moduri de lucru:

- *Comp*: robotul este controlat de programul utilizator
- *World*: Efectorul terminal al robotului este deplasat în spațiul cartezian World (din baza robotului) $(X/Y/Z)$ sau rotit $(RX/RY/RZ)$
- *Tool*: Deplasarea se realizează în sistemul de coordonate atașat efectorului terminal (Tool)
- *Joint*: Se controlează poziția individuală a fiecărei articulații
- *Free*: Permite poziționarea manuală a articulațiilor (doar pt. Cobra). Robotul poate fi controlat (mişcat) de programul utilizator numai atunci când modul de lucru setat din MCP este *Comp*. În celelalte moduri, execuţia unei instrucţiuni de mişcare va genera un mesaj de eroare.

Viteza de deplasare se ajustează folosind bara Speed Pot. Pentru poziționarea precisă a robotului se poate activa comutatorul Slow.

Pe ecran se pot urmări poziția carteziană a robotului (Disp → World Location) sau poziția fiecărei articulații (Disp → Joint Values).

Pentru mai multe detalii privind operarea dispozitivului MCP se poate consulta manualul *Adept T1 Pendant User's Guide* [6].

Sistemul de operare V^+

– *I heard about this thing called 'Linux'.*
– Oh, I use Linux.
– *What is it?*
– An operating system.
– *Like Firefox?*

Pe controller-ul robot rulează sistemul de operare V^+ . Utilizatorul poate controla sistemul folosind una din cele două aplicații:

- AdeptWindows PC
- AdeptDesktop [5]

Notă: AdeptDesktop nu este disponibil pe toți roboții din laborator.

AdeptWindows PC oferă acces la consola sistemului, o interfață în mod text similară cu cea folosită de Linux. În mediul V^+ , consola se numește *monitor*. Utilizatorul poate interacționa cu sistemul de operare utilizând *comenzi monitor*.

AdeptDesktop este un mediu de dezvoltare a programelor robot de tip *IDE* (Integrated Development Environment). Lucrul cu V^+ în AdeptDesktop se poate realiza fie prin interfața grafică, fie prin introducerea de comenzi la consola monitor.

Comenzile monitor nu sunt case-sensitive și pot fi prescurtate.

În configurația standard, sistemul V^+ poate rula până la 7 task-uri utilizator, numerotate de la 0 la 6. La un moment dat, un robot poate fi controlat de un singur task. În mod implicit, task-ul 0 este programul care controlează robotul (pe care se pot executa instrucțiuni de mișcare).

Sistemul V^+ are o structură de fișiere organizate în directoare, similară cu cea folosită în sistemele de operare uzuale. Programele V^+ pot fi salvate în fișiere cu extensia `.v2` sau `.pg`.

Lucrul cu fișiere

Lucrul cu fișiere în V^+ se poate face fie cu utilitarul File Manager din AdeptDesktop, fie folosind comenzi monitor asemănătoare cu cele din DOS sau Unix.

Discurile disponibile în laborator sunt:

- `DISK>D:\` - discul intern al fiecărui robot (CompactFlash)
- `NFS>C:\` - disc de rețea aflat pe server, partajat între toți roboții

Pentru aplicațiile de laborator se va folosi directorul `NFS>C:\scr\<grupa>\`.

```
.cd                    ; Afișează directorul curent
.cd ..                 ; Deplasare în directorul părinte
.cd DISK>D:\           ; Deplasare în rădăcina discului intern D:\
.cd NFS>C:\            ; Deplasare în rădăcina discului de rețea C:\
.cd \lab1\hello        ; Deplasare în directorul \lab1\hello
.fdir                  ; Afișarea fișierelor din directorul curent
.ls                    ; idem
.flist fisier.v2       ; Afișarea conținutului unui fișier
.fdir/c test                          ; Creare director test
.frename newname.v2 = oldname.v2      ; Redenumire fișier
.fcopy dest.v2 = src.v2               ; Copiere fișier
```

Editarea programelor

Pentru editarea programelor avem câteva variante:

- AdeptWindows Offline Editor
- Editorul din AdeptDesktop
- Editorul SEE din consola monitor V^+
- Orice alt editor ASCII (Notepad)

Variantele recomandate sunt AdeptDesktop (dacă este disponibil) sau SEE. Acestea rulează doar dacă PC-ul este conectat la controller-ul robot.

Dacă se dorește lucrul offline, se poate utiliza un editor ASCII, sau editorul offline AdeptWindows (nerecomandat).

Editorul SEE se activează folosind comanda monitor `see`:

```
.see numeprogram
```

Editorul SEE pornește în modul Command; pentru a scrie cod, trebuie trecut în modul Insert. Așadar, apăsați tasta INSERT.

Editorul suportă Copy/Paste la nivel de linii. Mai exact, o linie întreagă este copiată cu tasta F9. Operația Paste se realizează cu F10. Pentru a copia mai multe linii în memoria tampon (*copy buffer*), apăsați de mai multe ori F9 si de mai multe ori F10.

Nu se poate realiza Copy/Paste dintr-un program în altul. Mai mult, dacă există linii în *copy buffer*, nu puteți închide editorul. Acesta va afișa mesajul *Cannot exit while lines attached*. Pentru a ieși va trebui să apăsați CTRL-K de mai multe ori, până se golește buffer-ul.

Pentru a ieși din editorul SEE se va folosi tasta F4. Modificările vor fi salvate automat în memoria RAM. Pentru salvarea programului pe disc folosiți comenzile `store*`.

Gestionarea programelor

Programele V^+ pot fi organizate în *module*.

Comenzi monitor pentru gestionarea programelor:

```
.dir                          ; lista programelor încărcate în RAM
                                (nu de pe disc)
.rename newprog = oldprog     ; redenumire în memoria RAM
.copy newprog = oldprog       ; copiere în memoria RAM
.deletep prog                 ; șterge programul prog din memorie
.testp prog
.status                       ; afișează starea task-urilor
```

Salvarea și încărcarea programelor

Un program aflat în memorie poate fi salvat pe disc cu una din comenzile:

```
.store fisier          ; salvează tot (programe+variabile) în fisier.v2
.store fisier = prog   ; salvează programul prog împreună cu subrutinele
                         și variabilele utilizate de acesta, în fisier.v2
.store fisier = p1, p2
.storep fisier.v2      ; salvează toate programele, fără variabile, în
                         fisier.v2 (extensia implicită este .pg)
.storep fis.v2 = p     ; salvează doar programul p împreună
                         cu subrutinele apelate de acesta, dar fără variabile
```

Variabilele pot fi salvate în fișiere separate, cu comenzile `storel` (variabile de tip locație și puncte de precizie), `storer` (numere reale), `stores` (variabile de tip string).

```
.storel fisier         ; salvează variabilele locație în fisier.lc
.storel fis = prog     ; salvează în fis.lc variabilele locație utilizate
                         de programul prog
```

Încărcarea programelor și a variabilelor de pe disc se face cu comanda `.load numefisier` . Programele care există deja în memorie nu sunt suprascrise, însă variabilele existente sunt suprascrise.

Gestionarea variabilelor

```
.listl
.listr
.lists
.deleter realvar
.deletes $stringvar
.deletel loc.pick
.deletel #loc.safe
.deletel locations[]
.deleter timestamps[]
```

Execuția și depanarea programelor

Debugging: Removing the needles from the haystack.

```
.exec prog                  .bpt
.exec prog(pick,place)      .proceed
.abort 0                    .retry
.kill  0                    .sstep
.debug prog                 .xstep
.watch                      .prime
```

Comenzi diverse

`.zero`

Informații detaliate pot fi găsite în manualele V^+ *Operating System User's Guide* [3] și V^+ *Operating System Reference Guide* [4].

Întrebări și exerciții

Întrebări

- Pe ce task se va executa programul `test` lansat cu comanda monitor `.exec test` ? Ce comandă trebuie tastată pentru a rula acest program pe task-ul 5 ?
- Care este echivalentul comenzii DOS/Linux `mkdir` în V^+ ?
- Care este echivalentul comenzii DOS/Linux `dir` ?
- Se poate șterge un program de pe disc folosind comanda `deletep` ?

Exercițiu

În memoria sistemului V^+ se află programul `test`, care a fost încărcat cu comanda monitor `.load nfs>c:\scr\test.v2`. Ultima comandă monitor executată este `.exec test`, iar programul `test` și-a încheiat execuția.

Programul `test.v2` a fost modificat într-un editor extern (Notepad) și se dorește încărcarea și execuția noii versiuni în memoria V^+ . Ce comenzi vor fi introduse la consolă pentru a realiza acest lucru?

Indicație: se va consulta modul de utilizare a comenzii `deletep` în manualul V^+ Operating System Reference Guide.

Limbajul de programare V^+

Management: We don't really understand the problem, so let's give it to the programmers!

Programe V^+

Un program Hello World în V^+ arată astfel:

```
.PROGRAM hello()
    TYPE "Hello, World!"
.END
```

Un program puțin mai complex:

```
.PROGRAM numara(n)              ; program cu parametru
    LOCAL i
    TYPE "Numar pana la ", n    ; afisare mesaj la consola
    FOR i = 1 TO n
        TYPE i
    END
.END
```

Se pot observa: antetul `.PROGRAM` *nume.program*(*lista parametri*), declarațiile de variabile (`LOCAL i`), corpul programului (secvența de instrucțiuni), comentariile, precum și sfârșitul programului marcat prin `.END` .

Observație: Editorul SEE introduce automat `.END`, dar nu îl afișează explicit. Terminatorul `.END` trebuie însă introdus în editoarele externe (AdeptWindows Offline Editor, Notepad etc).

Variabile

V^+ operează cu următoarele tipuri de date:

- Numere reale și întregi
- șiruri de caractere (string)
- Variabile locație (transformări și puncte de precizie)
- Vectori (arrays)

Variabilele pot fi:

- `GLOBAL`: persistente și vizibile în toate programele
- `LOCAL`: persistente și vizibile în programul curent; toate instanțele programului curent folosesc în comun variabilele de tip LOCAL.
- `AUTO`: vizibile doar în programul curent; fiecare instanță a programului are o copie proprie a variabilei de tip AUTO.

Una din diferențele majore dintre LOCAL și AUTO apare atunci când un program are mai multe instanțe (fie rulează același program pe 2 task-uri în paralel, fie programul este recursiv). Atunci când un program modifică o variabilă LOCALă, toate celelalte instanțe ale programului vor vedea modificarea. Modificările asupra unei variabile AUTO nu sunt vizibile în celelalte instanțe ale programului. Cu alte cuvinte, echivalentul unei variabile locale din C/C++ este `AUTO`.

Variabilele persistente (GLOBAL și LOCAL) sunt menținute și după terminarea execuției programului, și își păstrează valoarea la următoarea execuție (dacă nu sunt reinițializate).

Variabilele nedeclarate sunt implicit globale. Se recomandă declararea variabilelor folosite ca `AUTO` sau `LOCAL`, cu excepția cazului în care acestea sunt folosite pentru comunicația între task-uri, sau reprezintă puncte robot învățate de la consolă. De asemenea, se recomandă declararea explicită a variabilelor GLOBALe.

Variabile scalare (numerice)

Variabilele de tip numeric pot fi întregi sau reale. Numerele reale pot avea precizie simplă (implicit) sau dublă. V^+ decide singur dacă o variabilă este număr întreg (pe 32 biți) sau număr real în precizie simplă. Variabilele în precizie dublă se declară explicit.

```
AUTO n
AUTO e
AUTO DOUBLE ang
n = 5
e = 2.718
ang = ATAN2(5, 100)
```

Caractere și string-uri

Variabilele de tip string sunt prefixate cu caracterul "$". Variabilele caracter sunt tratate ca variabile întregi, iar atribuirea se face cu ajutorul caracterului apostrof.

```
AUTO $msg
AUTO chr
$msg = "Variabila de tip string"
chr = 'X
```

Funcții utile:

- `LEN($msg)` - lungimea unui string
- `POS(haystack, needle, start)` - caută un subșir
- `VAL($x)` - conversie string → numeric
- `$MID($str, first.char, num.chars)` - extragere subșir
- `ASC($str, index)` - extragere caracter
- `$CHR(cod.ascii)` - conversie caracter - string
- `$ENCODE("X = ", x)` - concatenare

Variabile locație

Variabilele de tip locație sunt de 2 tipuri:

- *Puncte de precizie*: descriu poziția robotului prin valorile individuale ale articulațiilor. Sunt prefixate cu caracterul "#", iar atribuirea se face cu SET.
- *Transformări*: descriu locația în spațiu a unui corp solid (poziție + orientare). Nu au prefix; atribuirea acestora se face de asemenea cu SET. Conțin 6 elemente: X, Y, Z, Yaw, Pitch, Roll.

```
AUTO loc.test
AUTO #loc.safe
SET loc.test = TRANS(100,100,50,0,180,45)
SET #loc.safe = #PPOINT(0,-90,90,0,0,0)
```

Vectori (arrays)

Variabilele de tip vector pot conține numere, locații sau stringuri. Vectorii `GLOBAL` și `LOCAL` sunt alocați dinamic (se extind automat cât e nevoie). Vectorii `LOCAL` au dimensiune fixă.

```
AUTO joints[6]
~~AUTO joints[]~~
LOCAL timestamps[]
~~LOCAL timestamps[1000]~~
GLOBAL path.to.follow[]
GLOBAL $user.messages[]
```

Structuri de control

```
IF conditie THEN
    actiuni
END
```

```
IF a == b THEN
    actiuni
ELSE
    actiuni
END
```

```
FOR i = 1 TO 10
    actiuni
END
```

```
FOR i = 10 TO 1 STEP -1
    actiuni
END
```

```
WHILE a <> b DO
    actiuni
END
```

```
DO   ; nerecomandat
    actiuni
UNTIL conditie
```

```
CASE value OF
    VALUE 1:
        actiuni
    VALUE 2, 3:
        actiuni
    ANY    ; Fără :
        actiuni
END
```

```
CASE TRUE OF
    VALUE (x > 0)
        TYPE "Pozitiv"
    VALUE (x < 0)
        TYPE "Negativ"
    VALUE (x == 0)
        TYPE "Zero"
END
```

Buclele se pot întrerupe forțat cu instrucțiunea `EXIT` (echivalent cu `break` în C/C++). Atenție, `BREAK` în V^+ este instrucțiune de mișcare!

Subrutine

Orice program poate fi considerat o subrutină.

```
CALL subrutina()
CALL subrutina(param1, param2)
```

În mod implicit, parametrii sunt transmiși prin *referință*!. Din acest motiv, modificările efectuate în subrutină asupra parametrilor sunt vizibile în programul care a apelat subrutina.

Avantaj: parametrii sunt atât de intrare, cât și de ieșire.

Dezavantaj: programul poate deveni dificil de înțeles si depanat.

Transmiterea parametrilor prin *valoare* se realizează folosind un truc, mai exact, se construiesc artificial expresii care nu modifică valoarea variabilelor:

```
CALL subrutina((a), loc:NULL, $msg+"")
```

Aici, `a` este o variabilă reală, `loc` este o variabilă de tip transformare, iar `$msg` este un șir de caractere.

Nu se pot defini funcții utilizator (care întorc o valoare). Se pot folosi doar subrutine cu parametri de ieșire.

Lucrul cu consola

- `TYPE "Au fost recunoscute ", x, " piese."`
  → afișare mesaj la consolă
- `PROMPT "Introduceti numele piesei:  ", $nume.piesa`
  → citirea unui sir de caractere
- `PROMPT "Cate piese sunt?  ", nr.piese`
  → citirea unei variabile de tip numeric

Instrucțiuni de mișcare

```
MOVE loc
MOVES loc
MOVET loc, hand.opening
MOVEST loc, hand.opening
BREAK
APPRO loc, d
APPROS loc, d
DEPART d
DEPARTS d
```

Instrucțiunile `APPRO/S` poziționează efectorul terminal la distanța d "deasupra" punctului `loc`. Instrucțiuneile `DEPART/S` se deplasează cu distanța d "înapoi" sau "deasupra" poziției curente. Deplasarea se face în sensul negativ al axei Z din sistemul de coordonate atașat efectorului terminal (Tool).

Instrucțiunile cu sufixul `S` realizează o mișcare liniară în spațiul operațional (cartezian), iar instrucțiunile fără acest sufix realizează o mișcare interpolată liniar în spațiul articulațiilor. În timpul unei mișcări liniare în spațiul operațional, configurația robotului (LEFTY/RIGHTY, ABOVE/BELOW, FLIP/NOFLIP) nu poate fi schimbată.

Instrucțiunea BREAK așteaptă până când mișcarea curentă este terminată.

Controlul vitezei și accelerației:
`SPEED 30` ; doar pentru următoarea mișcare
`SPEED 50 ALWAYS` ; valabil pentru toate instrucțiunile de mișcare ce urmează, cu excepția celor precedate de `SPEED spd`
`ACCEL 50, 10` ; controlul accelerației și decelerației

Specificarea configurației robotului

```
Cobra si Viper:          Numai Viper:
LEFTY                    ABOVE
RIGHTY                   BELOW
SINGLE   [ALWAYS]        FLIP
MULTIPLE [ALWAYS]        NOFLIP
```

Controlul gripper-ului

```
PARAMETER hand.time = 0.5
OPENI
CLOSEI
OPEN
CLOSE
MOVET loc, TRUE
```

Switch-uri și parametri

```
ENABLE POWER
DISABLE UPPER
SWITCH DRY.RUN = FALSE
PARAMETER HAND.TIME = 0.5
```

Switch-uri de interes:

- `POWER` - activează High Power (alimentarea brațului robot)
- `DRY.RUN` - pentru testarea programelor fără a mișca robotul
- `TRACE` - se afișează fiecare linie de program executată (pentru depanare)
- `CP` - traiectorie continuă (prin interpolare similară cu B-Spline)
- `UPPER` - dacă este dezactivat, comparaţia între string-uri este *case sensitive*

Parametri de interes:

- `HAND.TIME` - timpul de așteptare la închiderea/deschiderea gripper-ului
- `VTIMEOUT` - pentru operațiile de vedere AdeptSight

Semnale

- Semnale de intrare: de la 1001 la 1012, de la 1033 la 1512.
- Semnale de ieșire: de la 1 la 8, de la 33 la 512.
- Semnale software interne (intrare/ieșire): de la 2001 la 2512.

```
SIGNAL 5            ; activare semnal
SIGNAL -5           ; dezactivare semnal
SIGNAL -4,5,2010    ; activarea semnalelor 5 și 2010 și dezactivarea semnalului 4
SIG(1001)           ; testare (citire) semnal 1001
SIG(-1002)          ; citire semnal 1002 în logică negativă
WAIT SIG(1001)      ; se așteaptă până când semnalul 1001 devine activ
```

Temporizare

Așteptare de 5 secunde:

```
WAIT.EVENT , 5
```

Așteptare până la îndeplinirea unei condiții:

```
WAIT SIG(1001) OR TIMER(1) > 5
```

Multitasking

```
EXECUTE task.num prog.name(param1, param2)
TAS  ;   test and set
STATUS("prog.name")
ABORT task.num
CYCLE.END task.num
KILL task.num
```

Informații detaliate pot fi găsite în manualele V^+ *Language User's Guide* [1] și V^+ *Language Reference Guide* [2].

De reținut

- *Toate variabilele se declară la începutul programului!*
- *Parametrii subrutinelor sunt transmiși prin referință!*
- *Nu se folosește instrucțiunea* `GOTO`*!*

Bibliografie

[1] *Adept V^+ Language User's Guide*

[2] *Adept V^+ Language Reference Guide*

[3] *Adept V^+ Operating System User's Guide*

[4] *Adept V^+ Operating System Reference Guide*

[5] *Adept DeskTop Online User's Guide*

[6] *Adept T1 Pendant User's Guide*

[7] *Adept AnyFeeder User's Guide*

[8] *AdeptSight Tutorials*

[9] *AdeptSight Online Help*

[10] *AdeptSight Calibration Wizards*