Branko Karan
Institut "Mihajlo Pupin"
Beograd, Volgina 15

PROGRAMSKI JEZIK ZA UPRAVLJANJE INDUSTRIJSKIM ROBOTIMA

A PROGRAMMING LANGUAGE FOR COMMANDING THE INDUSTRIAL ROBOTS

SADRŽAJ - Rad sadrži opis jezika RL namenjenog za programiranje neredundantnih robota sa do šest stepeni slobode. U definisanju strukture jezika posebna pažnja vodila se o mogućnostima implementacije na miniračunarima srednje klase. RL je projektovan kao jezik visokog nivoa koji podrazumeva programiranje robota u spoljašnjim koordinatama i jednostavno izražavanje prostornih odnosa, definisanje radnih položaja korišćenjem samog robota i komunikaciju robota sa okolnom opremom. U jezik je uključen koncept radnih zadataka i privatnih varijabli, čime se programeru RL paketa pruža mogućnost zaštite sistema od neispravnog korišćenja.

ABSTRACT - The paper contains a description of the language RL which is intended for programming of nonredundant robots with up to six degrees of freedom. During the definition of the language structure, a special attention was given to possibilities of its implementation on medium class minicomputers. The RL is designed as a high level language which supports robot programming in external coordinates and simple expression of space relationships, definition of working positions using the robot itself and communications between the robot and its surrounding equipment. A concept of operation tasks and private variables is included in the language, enabling the programmer to protect the system from an improper use.

1. UVOD

Većina postojećih sistema za upravljanje industrijskim robotima predvidja programiranje robota putem prenosivog upravljačkog sklopa i/ili komandnog pulta, tj. navodjenjem vrha robota kroz željenu putanju, memorisanjem tekućih položaja i unošenjem specijalnih komandi pomoću funkcionalne tastature sklopa za obučavanje (komande za definisanje načina ostvarivanja putanje pri radu robota i komande za izvršavanje karakterističnih operacija kao što su upravljanje hvataljkom, čekanje na spoljašnji signal, generisanje izlaznih signala itd.). S druge strane, sve izraženija potreba za rešavanjem složenijih zadataka, kao što je na primer pakovanje li sklapanje delova, zahtevala je uvodjenje efikasnijih sredstava za programiranje, što je dovelo do razvoja specijalizovanih tekstualnih programskih jezika za programiranje robota.

Prvi eksperimentalni tekstualni robotski jezik, WAVE, razvijen je još 1973. godine u laboratoriji za veštačku inteligenciju Univerziteta u Stanfordu [1]. Medjutim, trebalo je da prodje još punih 6 godina do pojave prvog komercijalnog robotskog jezika: bio je to sistem VAL [2], koji je do danas ostao najpopularniji robotski jezik. Od 1979. godine do danas, niz velikih uglavnom američkih proizvodjača industrijskih robota izneo je na tržište sisteme za upravljanje robotima koji podržavaju tekstualno programiranje. Karakteristike ovih sistema kao što su mogućnosti off-line programiranja, mogućnosti definisanja kretanja robota u spoljašnjim koordinatama i inkorporisane prednosti programskih jezika opšte namene svakako su bili značajni faktori koji su doveli do rapidnog povećanja broja robotizovanih radnih mesta u industriji. S druge strane, brz razvoj u ovoj oblasti doveo je do situacije u kojoj još uvek ne postoje šire usvojeni standardi robotskih jezika.

Gotovo bez izuzetka, postojeći komercijalni sistemi predvidjeni su za korišćenje sa odredjenim tipom robota ili familijom sličnih robota. Naprimer, sistem VAL namenjen je za programiranje robota porodice PUMA i robota serije 2000 i 4000 proizvodjača Unimation, sistem AML [3] omogućava programiranje robota IBM System/1 itd. Malobrojni sistemi koji se mogu primeniti na različite tipove robota (na primer [4]) poseduju brojne nedostatke koji se prvenstveno očitavaju u nemogućnosti kompenzacije dinamičkih efekata i složenom postupku pripreme sistema za rad s odredjenim robotom. Ovo je ujedno bio glavni razlog zbog koga se u Institutu "Mihajlo Pupin" pristupilo razvoju robotskog kontrolera opšte namene UMS-1. Kontroler je predvidjen za dinamičko upravljanje proizvoljnim tipovima neredundantnih robota sa do 6 stepeni slobode i razvijena je interaktivna procedura za prilagodjavanje kontrolera datom tipu robota. Detaljniji opis osnovnih koncepata primenjenih u kontroleru UMS-1 izložen je na drugom mestu [5], dok je u ovom radu dat neformalni opis jezika RL namenjenog za programiranje robota upravljanog ovom kontrolerom.

2. OPIS JEZIKA

U izradi projekta programskog jezika RL težilo se da se ostvare:

- prenosivost robotskih programa, tj. nezavisnost od odredjenog tipa robota; ovaj uslov je veoma značajan kod izrade složenijih robotskih programa, jer obezbedjuje da se jednom uloženi rad u pisanje programa ne obezvredi pri promeni mehaničke opreme;

- mogućnost definisanja načina sinhronizacije rada robota sa spoljašnjom sredinom, obzirom da većina robotskih aplikacija postavlja zahtev za intenziv-

nom komunikacijom robota, CNC mašina i pomoćnih uredjaja;

- jednostavnost programiranja radnih zadataka;

- mogućnost interpretativnog rada, obzirom da se u fazi testiranja i pripreme robotskih programa često postavlja zahtev za neposrednim izvršavanjem individualnih robotskih komandi;

- mogućnost implementacije na računarskom sistemu relativno skromnih karakteristika;

- mogućnost rada sa sistemom bez intenzivnog korišćenja spoljne memorije, čime se smanjuje verovatnoća otkaza sistema u fabričkim uslovima;

- robusnost sistema, tj. mogućnost zaštite od neautorizovanog korišćenja i slučajnih grešaka programera i/ili operatera.

Pri formulisanju sintakse niza postojećih robotskih programskih jezika često se kao jedan od glavnih projektnih ciljeva navodila mogućnost jednostavnog korišćenja sistema, čak i od strane neiskusnih korisnika nenaviknutih na rad s računarom i bez poznavanja osnovnih koncepta programiranja. S druge strane, ovakav prilaz obično implicira potrebu za veoma jakom tutorijalnom podrškom što neminovno dovodi do znatnih teškoća u implementaciji takvih sistema na malim računarima. Osim toga, opravdano je očekivati da je većina potencijalnih korisnika sistema za programiranje robota familijarna s nekim od programskih jezika opšte namene, kao što su PASCAL, BASIC, FORTRAN itd. Zbog toga je u specifikaciji programskog jezika RL usvojen nešto drukčiji pristup: postavljen je cilj da se korisnicima koji imaju izvesno iskustvo u programiranju omogući da iskoriste maksimum iz raspoložive opreme. U želji da se istovremeno izbegne potreba za učenjem novog programskog jezika, kao osnova za jezik RL iskorišćeni su elementi sintakse jezika PASCAL. Ovakav izbor napravljen je na osnovu pretpostavke da su osnovni elementi PASCAL-a u dovoljnoj meri poznati većini mašinskih inženjera i organizatora automatske obrade, koji bi trebalo da budu glavni projektanti automatizovanih radnih mesta i istovremeno glavni korisnici sistema za programiranje u industrijskoj robotici. Osim toga, značajan razlog za izbor PASCAL-a kao osnove za robotski jezik bio je i taj što PASCAL ohrabruje silazni pristup u izradi programa: ovakve osobine programskog jezika mogu u znatnoj meri da olakšaju timski rad u razvoju robotskih programa i da skrate vreme razvoja, testiranja i izmene programa.

U dizajniranju jezika RL usvojen je stav da robotski jezik treba da olakša sve glavne faze programiranja i eksploatacije robota, medju kojima su:

- definisanje zadataka robota; definisanje osnovnih elemenata radnog zadatka može se izvršiti bilo ad-hoc, bilo korišćenjem samog robotskog jezika; korišćenje drugog načina pretpostavlja mogućnost korišćenja korisničkih procedura koje se u inicijalnim fazama samo grubo opisuju, dok se detaljniji opis potrebnih izračunavanja i kretanja robota ostavlja za kasnije faze izrade programa; ovakav pristup može znatno da ubrza proces pisanja robotskih programa i da potpomogne dobijanje čitljivijih programa iz kojih jednostavnije mogu da se uklanjaju greške i koji mogu brže da se testiraju, podešavaju i proširuju;

- izrada robotskog programa koji treba da opiše ne samo rad robota tokom izvršavanja zadatka, nego i procedure za obučavanje robota, testiranje i podešavanje veličina u robotskom programu itd.;

- obučavanje robota, tj. utvrdjivanje i memorisanje željenih pozicija i orijentacija vrha robota tokom izvršavanja zadatka, kao i utvrdjivanje i memorisanje prostornih položaja i dimenzija radnih objekata;

- testiranje robotskih rpograma, koje može eventualno da uključi potrebu za ponovnim obučavanjem radnih položaja;

- eksploatacija, tokom koje takodje može da se pojavi potreba za povremenom izmenom pojedinih parametara robotskog programa.

Delovi robotskog programa koji odgovaraju pojedinim fazama, posebno fazama obučavanja i izvršavanja radnog zadatka, mogu se posmatrati kao posebni programski zadaci koji koriste zajedničku bazu podataka. Baza podataka sadrži sve podatke koji treba da se permanentno čuvaju u memoriji računara, kao što je željena trajektorija vrha robota, pozicije i orijentacije radnih objekata itd. Ovakva baza podataka implicitno je uključena u svim sistemima za programiranje robota. Medjutim u razvoju jezika RL stalo se na stanovište da se provera grešaka u programiranju i održavanje robotskih programa može izvršiti znatno lakše ako je baza podataka eksplicitno deklarisana od strane programera i ako su svi zadaci opisani u istoj programskoj jedinici koja je nazvana paket. Paket (PACKAGE) se sastoji iz deklaracije baze podataka (BASE), opisa inicijalnog računanja koja treba da se izvrši po pozivu paketa i opisa nula ili više zadataka (TASK). Opšta struktura paketa je:

```
PACKAGE naziv_paketa;
    BASE opis_baze_podataka;
    BEGIN
        opis_inicijalnog_računanja
    END;
```

```
TASK naziv___zadatka___1;
    BEGIN
        opis___zadatka___1
    END;
TASK naziv___zadatka___2;
    BEGIN
        opis___zadatka___2
    END;
.
.
.
TASK naziv___zadatka___n;
    BEGIN
        opis___zadatka___n;
    END.
```

Izvršavanje zadataka zahteva se od strane korisnika jednostavnim navodjenjem naziva zadatka, tako da se zadatak može posmatrati kao korisničko proširenje sistemskih rutina namenjenih za dodeljivanje vrednosti programskim varijablama, komunikaciju izmedju korisnika i sistema, kretanje robota, rad efektora, sinhronizaciju s spoljašnjom okolinom i zadavanje kontrolnih parametara koji regulišu način rada celog sistema. Izvršavanje zadataka moguće je tek kada je izvršena inicijalizacija paketa u kome su dati paketi definisani.

Jezik RL podržava korišćenje logičkih (BOOLEAN), celobrojnih (INTEGER) i realnih (REAL) tipova podataka, podataka koji sadrže nizove printabilnih znakova (STRING) i podataka tipa vektor (VECTOR) i telo (BODY). Logički podaci mogu da dobiju vrednosti iz skupa (TRUE, FALSE). Opseg dozvoljenih vrednosti za celobrojne i realne podatke zavisi od konkretne implementacije: na mikroprocesoru Intel 8086 celobrojni podaci mogu da budu u opsegu - 32768 do +32767 dok realni podaci mogu da imaju apsolutne vrednosti u opsegu 10^{-70} do 10^{+70} uključujući nulu, sa približno sedam značajnih decimalnih cifara. Niz znakova predstavlja sekvencu od nula ili više (do 255) printabilnih ISO simbola. Vektor je definisan kao uredjena trojka realnih podataka (x, y, z) koji reprezentuju koordinate vektora u referentnom koordinatnom sistemu, dok je telo definisano kao uredjena šestorka realnih podataka (x, y, φ, θ, ψ) koji reprezentuju Euler-ove koordinate sistema vezanog za telo u odnosu na referentni koordinatni sistem.

Podaci koje podržava jezik RL mogu da budu konstante ili varijable. Referisanje na varijable vrši se pomoću identifikatora koji se mogu konstruisati iz

proizboljnog niza alfanumeričkih znakova, uključujući znak podvlačenja "_" i koji počinje slovom. Poput PASCAL-a, RL zahteva da se svi identifikatori koji se koriste kao varijable ili imenovane konstante eksplicitno deklarišu. Na primer:

```
CONST   PORUKA = 'KRAJ';
        PRILAZ = (0.0, 20.0, 0.0, 0.0, 0.0, 0.0)
VAR     TRANSL : VECTOR;
        GREŠKA: BOOLEAN;
        PRIHVAT, PREDMET, BLOK: BODY;
```

Osim elementarnih tipova podataka (INTEGER, REAL itd.), u jeziku RL moguće je i deklarisanje masiva; deklarisanje masiva vrši se korišćenjem ključne reči ARRAY, navodjenjem donje i gornje granice masiva i navodjenjem tipa elemenata masiva. Na primer:

```
VAR     OTVOR: ARRAY [1..4] OF BODY;
```

Jezik RL podržava rekurzivno definisanje masiva, čime se omogućava korišćenje višedimenzionalnih varijabli.

Dodeljivanje vrednosti varijablama definiše se operatorom dodeljivanja ":=". Takodje, dodeljivanje vrednosti može se izvršiti preko tastature korisničkog terminala pri izvršavanju komande ACCEPT koja ima oblik:

```
ACCEPT varijabla
```

Specijalna komanda predvidjena je za dodeljivanje vrednosti varijablama tipa BODY:

```
LEARN   objekat
```

Komanda LEARN omogućuje da se sadržaj varijable objekat tipa telo izjednači sa tekućom pozicijom i orijentacijom vrha robota (obično nakon intervencije operatora kojom se putem komandne palice ili prenosivog upravljačkog uredjaja vrh robota dovede u željeni položaj).

Referisanje na pojedine komponente vektora i tela može se izvršiti korišćenjem ključnih reči X, Y, Z, PHI, THETA, PSI i POS. POS je definisan kao uredjena trojka (x, y, z) koja reprezentuje poziciju početka koordinatnog sistema vezanog za telo u odnosu na referentni koordinatni sistem. Naprimer:

```
(* postavljanje x-komponente varijable TRANSL na 20 [mm] *)
TRANSL.X: = 20.0;
(* transliranje sadržaja varijable BLOK za TRANSL *)
BLOK.POS: = BLOK.POS + TRANSL;
```

U interpretator RL paketa uključen je skup bibliotečkih procedura za izračunavanje logičkih, celobrojnih i realnih izraza. Jezik uključuje logičke (NOT, AND, OR), relacione (=, <, >, <=, >=, <>) i aritmetičke operatore (+, -, *, /, **) za izvršavanje skalarnih operacija. Operacije nad vektorima mogu se ostvariti pomoću operatora vektorskog sabiranja i oduzimanja (+, -) i operatora množenja vektora skalarom (*).

Medjusobni odnosi tela mogu se izraziti korišćenjem operatora komponovanja "*" i bibliotečke funkcije INV. Primena binarnog operatora komponovanja nad operandima tipa telo daje poziciju i orijentaciju tela u odnosu na referentni sistem, pri čemu drugi operand reprezentuje poziciju i orijentaciju tela u odnosu na koordinatni sistem čija je pozicija i orijentacija u odnosu na referentni sistem reprezentovana prvim operandom. Funkcija INV daje poziciju i orijentaciju referentnog sistema u odnosu na sistem vezan za telo čija je pozicija i orijentacija reprezentovana argumentom funkcije. Na primer, ako varijabla PRIHVAT reprezentuje poziciju i orijentaciju vrha hvataljke robota u momentu prihvatanja nekog objekta i ako je pozicija i orijentacija donje osnove radnog objekta data varijablom OSNOVA, onda se položaj za prihvatanje objekta OBJEKAT u odnosu na donju osnovu može izračunati kao:

OBJEKAT: = INV (OSNOVA) * PRIHVAT

Kao i u drugim programskim jezicima, standardni prioritet operatora može se izmeniti korišćenjem zagrada.

Vrednosti varijabli, konstanti i izraza mogu se prikazati na korisničkom terminalu korišćenjem komande DISPLAY čiji je format:

DISPLAY izraz_1, izraz_2, . . ., izraz_n

Najvažnija komanda jezika je komanda za specificiranje kretanja robota. Izvršavanje komande kretanja oblika:

MOVE objekat TO položaj

dovodi do koordinisanog kretanja zglobova robota sve dok pozicija i orijentacija objekta vezanog za efektor robota ne postane jednaka poziciji i orijentaciji dobijenoj izračunavanjem specificiranog izraza položaj. Na primer, ako je EFEKTOR varijabla tipa telo koja reprezentuje tekuću poziciju i orijentaciju efektora u odnosu na referentni sistem, izvršavanje komande:

MOVE OBJEKAT TO OSNOVA * BLOK * OTVOR [1]

rezultuje u koordinisanom kretanju sve dok se ne zadovolji:

EFEKTOR = OSNOVA * BLOK * OTVOR |I|* INV(OBJEKAT)

Specifikacija objekta može se izostaviti, i u tom slučaju izvršavanje komande MOVE dovodi do pomeranja vrha efektora robota u poziciju i orijentaciju reprezentatovane izrazom položaj.

Sinhronizacija sa spoljašnjom opremom kao što su konvejeri, senzorski uređaji itd., ostvaruje se korišćenjem 32 ulazna i 32 izlazna kanala, posredstvom pridruženih varijabli. Varijabla se pridružuje kanalu korišćenjem iskaza ATTACH čiji je oblik:

ATTACH identifikator: tip TO tip_kanala broj_kanala;

gde identifikator označava skalarnu varijablu, tip reprezentuje tip varijable koji može da bude celobrojni ili logički, tip_kanala reprezentuje ulazni ili izlazni kanal (INPUT ili OUTPUT), dok je broj_kanala celobrojna vrednost u opsegu 1 do 32. Ovakvo eksplicitno pridruživanje uključeno je u jezik RL s ciljem da se korisniku omogući bolja čitljivost programa u kojima se koriste ulazne i izlazne linije s različitim značenjem. S druge strane, na ovaj način pružaju se prevodiocu RL programa dodatne informacije korisne za proveru semantičkih grešaka (na primer, zabranjeno je dodeljivanje vrednosti varijablama koje su pridružene ulaznim kanalima).

Korišćenje varijabli pridruženih kanalima moguće je u svim izrazima u kojima je dozvoljeno korišćenje podataka istog tipa. Generisanje signala za izlazne kanale vrši se kao rezultat izvršavanja operatora dodeljivanja vrednosti odgovarajućim pridruženim varijablama. Takodje, moguće je eksplicitno čekanje na neki spoljašnji dogadjaj korišćenjem komande WAIT:

WAIT dogadjaj

gde je dogadjaj bilo koji logički izraz koji uključuje bar jednu varijablu pridruženu nekom ulaznom kanalu. Naprimer:

```
ATTACH  NIVO: INTEGER TO INPUT 2;
        SPREMAN: BOOLEAN TO OUTPUT 10;
.
.
.
WAIT NIVO < 100;
SPREMAN := TRUE;
```

Specifikacija komande kretanja može da sadrži niz kontrolnih parametara kao i testiranje izraza koji uključuju varijable pridružene ulaznim kanalima. Opšti

oblik komande MOVE u jeziku RL je:

```
MOVE objekat TO krajnji___položaj
             VIA položaj___1, položaj___2, . . ., položaj___n
             DEPART d___izraz
             APPRO a___izraz
             WITH kontrola___1, kontrola___2, kontrola___m
             UNTIL dogadjaj
```

U ovom obliku komande izrazi položaj___1, položaj___2, . . . , položaj___n daju pozicije i orijentacije medjupoložaja kroz koje objekat treba da prodje tokom kretanja u krajnji___položaj bez zaustavljanja robota. Izraz d___izraz definiše relativnu poziciju i orijentaciju objekta u odnosu na polazni položaj do koga brzina vrha robota treba da se poveća do maksimalne željene brzine; analogno, a___položaj definiše relativnu poziciju i orijentaciju objekta u odnosu na krajnji___položaj počev od koje brzina vrha robota treba da počne da se smanjuje. Ako je POLAZNI___POLOŽAJ varijabla tipa telo koja reprezentuje početnu poziciju i orijentaciju vrha robota, tokom izvršavanja komande MOVE treba da se zadovolje sledeće relacije:

```
EFEKTOR = POLAZNI___POLOŽAJ
EFEKTOR = POLAZNI___POLOŽAJ * d___izraz * INV(objekat)
EFEKTOR = položaj___1 * INV(objekat)
EFEKTOR = položaj___2 * INV(objekat)
.
.
.
EFEKTOR = položaj___n * INV(objekat)
EFEKTOR = krajnji___položaj * a___izraz * INV(objekat)
EFEKTOR = krajnji___položaj * INV(objekat)
```

Opcionalna stavka WITH omogućuje korisniku da kontroliše način kretanja. Kontrole kontrola___1, kontrola___2, . . ., kontrola___m imaju oblik:

```
ključna___reč = vrednost
```

gde ključna___reč specificira parametar kontrolera (PASSMODE, SPEED, MAXTIME, TOLERANCE, EXECMODE itd.), dok vrednost reprezentuje željenu vrednost parametra, koja može da bude ključna reč, celobrojna ili realna vrednost. Na primer:

```
MOVE OBJEKAT TO OSNOVA WITH PASSMODE = JOINT, MAXTIME = 3.0
```

specificira da kretanje treba da izvrši uz linearnu promenu unutrašnjih koordinata robota (standardni modus kretanja, alternativni modusi su STRIGHT, uz linearnu

promenu Euler-ovih koordinata i PARABOLIC, uz kvadratnu interpolaciju) i za vreme manje ili jednako 3 sekunde.

Opcionalna stavka UNTIL specificira logički izraz dogadjaj koji uključuje varijable pridružene ulaznim kanalima; izraz se izračunava tokom kretanja robota i kretanje se zaustavlja čim vrednost izraza dogadjaj postane istinit.

Alternativni način za definisanje kretanja je specificiranje željenih pomeraja u unutrašnjim koordinatama robota. Korišćenjem komande DRIVE:

```
DRIVE  broj___zgloba___1 BY pomeraj___1
       broj___zgloba___2 BY pomeraj___2
       .
       .
       .
       broj___zgloba___n BY pomeraj___n
       UNTIL dogadjaj
```

pri čemu celobrojni izrazi broj___zgloba___1, broj___zgloba___2, . . ., broj___zgloba___n označavaju indeksa zglobova robota (počev od osnove ka vrhu), a realni izrazi pomeraj___1, pomeraj___2, . . ., pomeraj___n odgovarajuće pomeraje; Opcionalna stavka UNTIL ima isto značenje kao kod komande MOVE. Strogo govoreći, uvodjenjem mogućnosti za definisanje kretanja robota u unutrašnjim koordinatama smanjuje se portabilnost robotskih programa; ipak, komanda DRIVE ostavljena je u jeziku RL obzirom da u nekim slučajevima programiranja u unutrašnjim koordinatama može da rezultuje u efikasnijem radu robota.

Za rad s hvataljkom robota predvidjene su komande:

```
OPEN UNTIL dogadjaj
CLOSE UNTIL dogadjaj
```

kojima se zahteva otvaranje/zatvaranje hvataljke; opcionalni sufiks UNTIL dogadjaj omogućuje izračunavanje specificiranog logičkog izraza i prekidanje otvaranja/zatvaranja hvataljke, isto kao kod komandi MOVE i DRIVE. Radom ostalih efektora (kao kod bojenja itd.) upravlja se dodeljivanjem vrednosti varijeblama pridruženim odgovarajućim izlaznim kanalima.

Kontrolni parametri koji su zajednički za sekvencu komandi MOVE mogu se takodje postaviti preko komande:

```
SET kontrola___1, kontrola___2, . . ., kontrola___n
```

pri čemu lista kontrola ima isto značenje kao kod komande MOVE.

Elementarne komande jezika RL (tj. komande dodeljivanja vrednosti, komande čekanja, komande za prikazivanje sadržaja programskih varijabli, komande kretanja, komande za rad s hvataljkom i komande za postavljanje kontrolnih parametara) ne predstavljaju samo elemente za formiranje programskih paketa; izvršavanje ovih komandi može se takodje zahtevati direktnim unošenjem željene komande preko korisničkog terminala, čime se može pojednostaviti priprema robota za izvršavanje odredjenih zadataka.

Redosledom izvršavanja komandi u programskim paketima može se upravljati pomoću uobičajenih BEGIN ... END, IF ... THEN ... ELSE, WHILE ... DO i REPEAT ... ... UNTIL konstrukcija. Komande se mogu grupisati u procedure ili funkcijske potprograme koji se mogu slobodno koristiti kao korisničke komande ili elementi izraza; njihovo korišćenje ograničeno je na paket u kome su deklarisani i nije dozvoljeno izvršavanje korisničkih potprograma direktnim unošenjem naziva potprograma preko tastature korisničkog terminala.

Doseg varijabli i imenovanih konstanti su potprogrami i programski zadaci u kojima su varijable/konstante deklarisane. Varijable i konstante deklarisane u delu za inicijalizaciju paketa, bazne varijable i varijable pridružene ulaznim/izlaznim kanalima globalne su za ceo paket.

Pri zahtevima za neposredno izvršavanje RL komandi postoji stalna opasnost da njihovo izvršenje dovede do neželjenih promena sadržaja programskih varijabli i time do neočekivanih rezultata rada RL zadataka. Da bi se onemogućile ovakve neželjene izmene i da bi se istovremeno omogućila potpunija kontrola operatera nad načinom izvršavanja robotskih zadataka, u jezik RL uvedena je mogućnost deklarisanja pojedinih globalnih varijabli kao javnih ili privatnih. Naprimer, deklarisanje jedne bazne varijable može da se izvrši konstrukcijom:

BASE identifikator: opcija tip

pri čemu opcija može da bude PUBLIC ili PRIVATE (sve varijable za koje nije specificirana ova opcija tretiraju se kao privatne). Na taj način omogućeno je programeru RL paketa da eksplicitno naznači koje globalne varijable operater može da menja tokom eksploatacije paketa.

3. ZAKLJUČAK

Opisani jezik za programiranje robota karakterišu jednostavnost i prirodnost programskih iskaza. U jezik su uvedeni elementi koji omogućavaju programiranje u spoljašnjim koordinatama i iskazivanje prostornih odnosa radnih objekata (komande kretanja i operacije nad podacima tipa telo), definisanje radnih položaja korišćenjem samog robota (komanda LEARN), komunikaciju robota sa spoljašnjom okolinom

(pridruživanje varijabli spoljašnjim kanalima, komanda čekanja itd.) i zaštitu sistema od nepravilnog korišćenja (koncept zadataka i privatne varijable). Programer RL paketa ima mogućnost korišćenja niza konstrukcija karakterističnih za programske jezike visokog nivoa; istovremeno, jezik sadrži relativno mali broj redundantnih konstrukcija i sintaksa jezika nije preobimna, čime se znatno olakšava njegova implementacija na mini računarima srednje klase.

4. LITERATURA

[1] Paul R.P., "WAVE: A Model-Based Language for Manipulator Control", The Industrial Robot, Vol. 4. No. 1, 1977.

[2] Unimation Inc., "User's Guide to VAL", Version 12, Danbury, USA, 1980.

[3] Taylor R.H., Summers P.D., Mayer J.M., "AML: A Manufacturing Language", Robotics Research, Vol. 1. No. 3, pp. 19-41 , 1982.

[4] HORYU Controller Engineering Co., "TAIRIKU Model-82H Controller Operations Manual", Kanagawa, Japan, 1983.

[5] Vukobratović M., Kirćanski N., Stokić D., Kirćanski M., Karan B., "General Purpose Controller for Industrial Manipulators", Proc. of the Second Yugoslav-Soviet Symposium on Applied Robotics, pp. 1-15, Belgrade, 1984.