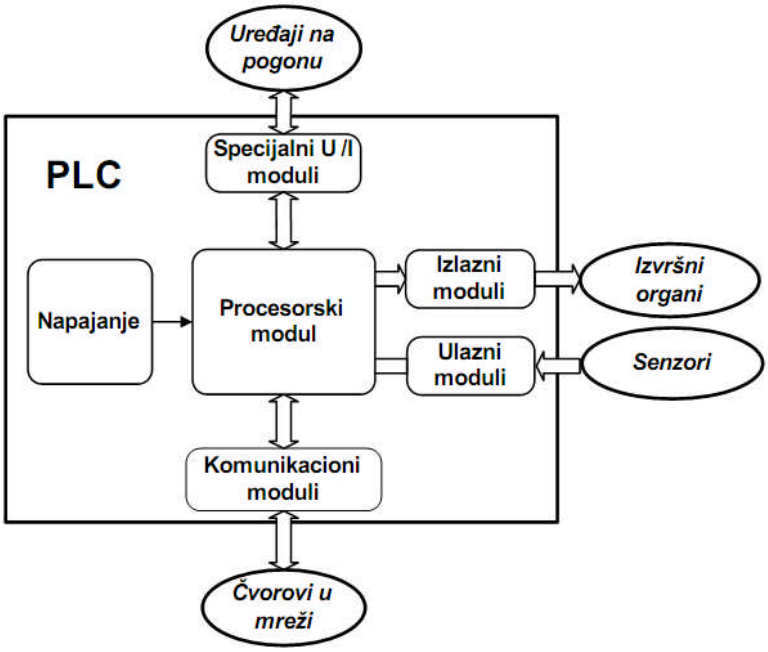
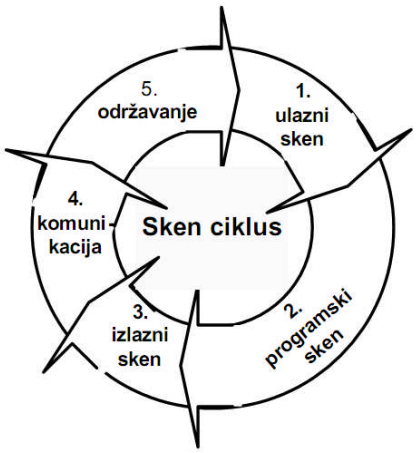
ISPITNA PITANJA ZA PREDMET PLC

1. Osnovna struktura PLC-a.

Prvobitno PLC je zamišljen kao specijalizovani računarski uređaj koji se može programirati tako da obavi istu funkciju kao i niz logičkih ili sekvencijalnih elemenata koji se nalaze u nekom relejnom uređaju ili automatu. Postepeno, obim i vrsta operacija koju može da obavi PLC proširena je uključivanjem složenijih funkcija potrebnih za direktno digitalno upravljanje nekim sistemom. PLC se može prikazazi na sledeći način: **Procesorski modul** sadrži *centralnu jedinicu* i *memorju*. U okviru ovog modula smeštaju se i program i podaci i odatle se upravlja radom celog sistema. Naziv **izlazni** i **ulazni moduli** se odnosi na *digitalne ulaze i izlaze* preko kojih se primaju binarni signali sa senzora, odnosno zadaju binarni signali pojedinim aktuatorima, dok **specijalni U/I moduli** obuhvataju *analogne U/I* kao i module posebne namene kao što su *brzi brojac, pozicioni servo sistem, PID regulator* itd. **Komunikacioni moduli** obezbeduju spregu sa komunikacionom opremom preko koje se razmenjuju podaci sa drugim računarskim uređajima u mreži i/ili operatoriskim uređajima preko kojih se PLC programira i nadzire njegov rad. PLC se sastoji iz *šasije (rack)* koja ima određeni broj *slotova* u koji se stavljaju pojedini *moduli*. Prvi dva slota u šasiji zauzimaju uređaj za napajanje i procesorski modul, dok je raspored modula u preostalim slotovima proizvoljan. U zavisnosti od broja modula, PLC može imati i više od jedne šasije. Svaka šasija ima sopstveno napajanje, dok se procesorski modul nalazi samo u prvoj šasiji. Programabilini logicki kontroleri iz familije ***Allen Bradley SLC 500 Modular Controllers*** mogu imati najviše **tri** šasije sa najviše **30 slotova**. Pri tome, postoje šasije sa 4, 7, 10 i 13 slotova.





2. Sken ciklus PLC-a.

Operativni sistem PLC-a projektovan je tako da, u toku rada sistema, automatski obezbedi ciklično ponavljanje navedenih aktivnosti (*Sken ciklus*). Sken ciklus započinje sa ***ulaznim***

skenom u okviru koga PLC ocitava sadržaj ulaznih linija (registara ulaznih modula). Očitani podaci se prenose u određeno područje memorije – ***slika ulaza***. Zatim se aktivira ***programski sken*** u okviru koga procesor izvršava programske naredbe kojima su definisane odgovarajuće aritmetičko-logicke funkcije. Podaci (operandi) koji se koriste u programskim naredbama uzimaju se iz memorije i to iz područja označenog kao *slika ulaza* (ako su operandi ulazni podaci) ili iz podrucja gde se smeštaju interne promenljive. Rezultati obrade se smeštaju u posebno podrucje memorije – ***slika izlaza***. Po zvršetku programskog skena, operativni sistem PLC-a aktivira ***izlazni sken*** u okviru koga se podaci iz *slike izlaza* prenose na izlazne linije (registre izlaznih modula). Cetvrti deo sken ciklusa – ***komunikacija*** - namenjen je realizaciji razmene podataka sa uređajima koji su povezani sa PLC-om. Nakon toga, operativni sistem dovodi PLC u fazu ***održavanja*** u okviru koje se ažuriraju interni časovnici i registr obavlja upravljanje memorijom kao i niz drugih poslova vezanih za održavanje sistema, o kojima korisnik i ne mora da bude informisan. U zavisnosti od tipa procesora ulazni i izlazni sken ciklus izvršavaju se u vremenu reda mili sekundi (od 0.25 ms do 2,56 ms). Trajanje programskog skena, svakako zavisi od veličine programa.Osnovni sken ciklus može biti modifikovan pomoću zahteva za prekid ili nekih drugih specijalnih programskih naredbi.

3. Osnovne karakteristike CPU-a.

Procesorski modul sadrži centralnu procesorsku jedinicu (*CPU*) i memoriju. Centralna jedinica obuhvata aritmetičko-logičku jedinicu (*ALU*), registre i upravljačku jedinicu. U funkcionalnom smislu centralna jedinica se bitno ne razlikuje od centralne jedinice bilo kog mikroračunara opšte namene. Osnovna razlika se ogleda u skupu naredbi koji je odabran tako da se zadovolje osnovni zahtevi u pogledu korišcenja PLC-a. Osnovne karakteristike procesorskog modula izražavaju se preko sledećih elemenata. *Memorija(RAM)* - je okarakterisana svojom veličinom, mogućnošću proširenja i konfigurisanja za smeštanja programa ili podataka. *U/I tačke* - su okarakterisane najvećim brojem lokalnih U/I adresa koje podržava procesor u toku ulaznog i izlaznog skena, kao i mogućnošcu proširenja preko *udaljenih U/I*. (Pod udaljenim U/I podrazumeva se posebna šasija koja sadrži U/I module koji razmenjuju podatke sa PLC-om). *Komunikacione opcije* - odnose se na raznovrsnost uređaja za spregu (komunikaciong interfejsa) koji podržavaju različite topologije mreža i različite komunikacione protokole. *Opcije trajnog pamćenja* - odnose se na raspoloživost različitih tipova memorijskih EPROM modula koji obezbeđuju trajno pamćenje podataka. *Performansa* - se specificira preko *vremena programskog skeniranja* potrebnog za 1Kbajt programa, preko *vremena* potrebnog *za ulazni i izlazni sken*, kao i *vremena izvršavanja jedne bit naredbe*. *Programiranje* - se specificira u odnosu na broj različitih mašinskih naredbi, kao i na vrstu raspoloživih programskih jezika.

4. Komunikacione karakteristike procesora.

Ulaznim interfejsima se ostvaruje veza između PLC-procesora i spoljnih uređaja koji se koriste za merenje fizičkih i električnih veličina okoline. Ulazni uređaji najčešće imaju ugrađeno prekidače, senzore, pojačavače i drugu elektroniku kondicionerskog tipa. Ovi uređaji se često nazivaju *field* uređaji (instaliraju se na licu mesta procesa) što ukazuje da oni nisu deo (ne pripadaju) PLC-ovog hardvera. Komunikacioni moduli ostvaruju vezu izmedju PLC-procesora i drugih računarsko-kontrolisanih mašina i uređaja koji dele iste podatke i zahteve za daljinskim upravljanjem sa PLC-sistemom.

5. Organizacija RAM memorije. s

Operativni sistem kontrolera, koji realizuje sken cikluse, upravlja i zauzećem RAM memorije, koja je organizovana na poseban način. U principu, RAM meorija se deli na *program files (programske datoteke)* i *data files (datoteke podataka)*. Skup programa i datoteka podataka koje su formirane za jednu aplikaciju čini *processor file (procesorsku datoteku)*. Ona sadrži sve naredbe, podatke i specifikaciju modula koji su relevantni za datu aplikaciju, odnosno korisnički program. Procesorska datoteka čini jednu celinu koja se može prenositi sa jednog procesorskog modula na drugi, sa odnosno na EPROM memorijski modul, kao i operatorski terminal. To zapravo znači da se jedna aplikacija može razviti na jednom sistemu i zatim u celini preneti i koristiti na drugom sistemu.

Programske datoteke - sadrže, informacije o samom kontroleru, glavne korisničke programe i potprograme. Svaka aplikacija mora da ima sledeće tri programske datoteke:

System Program – sistemski program (file 0) - sadrži različite informacije o samom sistemu kao što su tip proceosra, konfiguracija U/I modula, ime procesorske datoteke, lozinku i niz drugih relevantnih podataka.

Reserved – datoteka rezervisna za potrebe operativnog sistema (file 1)

Main Ladder Program – glavni leder program (file 2) – program koji formira sam korisnik i u okviru koga se definiše niz operacija koje SLC treba da izvede.

Subroutine Ladder Program - potprogrami (file 3 - 255) – korisnički potprogrami koji se aktiviraju u skladu sa naredbama za njihovo pozivanje koje se nalaze u glavnom programu.

Datoteke podataka - sadrže podatke koji se obrađuju pomoću naredbi leder programa. Pri tome se pod pojmom *podaci* podrazumevaju konvertovane (numericke) vrednosti signala koji se preko ulazno/izlaznih modula unose u konrtoler, ili se iz kontrolera prenose na izlazne uredaje, kao i interne promenljive koje se koriste kao operandi u različitim operacijama. Datoteke podataka organizovane su u skladu sa tipom promenljivih koje sadrže. To zapravo znači da jedna datoteka sadrži samo jedan tip (vrstu) podataka. Jedna procesorska datoteka može da ima najviše 256 datoteka podataka.

6. Organizacija datoteke podataka (tipovi promenljivih i datoteka).

Osnovna karakteristika datoteke podataka je njen *tip*. Kao što je vec istaknuto tip datoteke, zapravo ukazuje na vrstu promenljivih koje se u njoj pamte. Jedna datoteka se označava pomoću *rednog broja*, koji jednoznačno odreduje mesto te datoteke u nizu datoteka podataka koje se nalaze u jednoj procesorskoj datoteci i *slova* kojim se identifikuje tip datoteke. Prvih 9 datoteka imaju unapred definisan tip koji ne može da se menja. Tipove preostalih datoteke korisnik sam odabira i definiše u skladu sa aplikacijom koju razvija.

File 0 – *Tip* **O** - *output (izlaz)* – sadrži *sliku izlaza*; sadržaj datoteke se prenosi na izlazne linije za vreme izlaznog skena.

File 1 – *Tip* **I** - i*nput* (*ulaz*) - sadrži *sliku ulaza*; u ovu datoteku se za vreme ulaznog skena smeštaju vrednosti sa ulaznih linija.

File 2 – *Tip* **S** - *status* - sadrži podatke vezane za rad kontrolera. Pregled znacenja pojedinih bitova u ovoj datoteci dat je u posebnom dodatku (Dodatak S).

File 3 – *Tip* **B** - *bit* – sadrži interne promenljive *bit* tipa.

File 4 – *Tip* **T** - *timer* (*casovnik*) - sadrži podatke koji se koriste za interne *časovnike*.

File 5 – *Tip* **C** - *counter* (*brojac*) - sadrži podatke koji se koriste za interne *brojače*.

File 6 – *Tip* **R** - *control* (*upravljanje*) – sadrži dužinu, položaj pokazivača i bitove statusa za određene naredbe kao što su naredbe za pomeranje sadržaja registara i sekvenci.
File 7 – *Tip* **N** - *integer* (*celobrojna*) – sadrži podatke celobrojnog tipa.
File 8 – *Tip* **F** - *floating point* (realna) - sadrži podatke predstavljene u tehnici pokretnog zareza kao 32-bit brojeve u opsegu (±1.1754944e-38 to ±3.40282347e+38) - *ovo važi samo za SLC 5/03 i SLC5/04*.
File 9 do file 255 – *Tip* **definiše korisnik** - k*orisnicke datoteke* – ove datoteke definiše korisnik kao datoteke tipa **B, T, C, N**.

7. Kreiranje datoteke i zauzeće memorije.

Jedna datoteka podataka zauzima memorijski prostor koji obuhvata niz susednih reči. Broj reči koje zauzima jedna datoteka određen je najvećom adresom elementa te datoteke koji se koristi u programskim naredbama. Prvo navođenje broja datoteke inicijalizuje njeno kreiranje. Pri tome tip datoteke koji je naveden u adresi određuje broj reči koje se pridružuju jednom elementu, dok adresa elementa odreduje niz konsekutivnih elemenata za koje se u memoriji rezerviše prostor. Taj niz počinje od nule, a završava se sa adresom elementa koja je navedena u naredbi. Ako se kasnije pojavi veća adresa elementa iste datoteke onda se prethodno rezervisani prostor proširi tako da uključi i tu adresu. Dozvoljeno je kreiranje najviše 256 datoteka podataka. Samo se po sebi razume da će broj kreiranih datoteka zavisiti od promenljivih koje korisnik definiše u programu. Pri tome sve kreirane datoteke ređaju se u nizu jedna iza druge. U formiranoj aplikaciji, datoteke podataka zauzimaju jedan neprekidan memorijski prostor. Redni brojevi ovih datoteka poređani su u rastućem nizu, ali ne moraju da cine kontinuallni niz. Operativni sistem kontrolera dozvoljava da se datoteke podataka kreiraju i direktno, a ne preko naredbi u kojima se navode adrese operanada. U tu svrhu koristi se posebna procedura *memory map function*, koja omogućuje da se rezerviše odgovarajući prostor i u njega direktno upišu podaci. Ista procedura omogućava i da se *obriše* grupa elemenata u nekoj definisanoj datoteci ili cela datoteka, naravno samo uz uslov da se ne koriste u programu.

8. Vrste adresiranja (indirektno, indeksno, adresiranje datoteka).

Indirektno adresiranje se realizuje tako što se navodi adresa promenljive čiji sadržaj predstavlja adresu promenljive (operanda) nad kojom se vrši neka operacija. U skladu sa time promenom sadržaja navedene promenljive menja adresa operanda. Adresa promenljive uključuje *broj datoteke* (*f*), *broj elementa* (*e*), *broj reci* (*w*) i *broj bita* (*b*). Svaki od ovih podataka može biti indirektno adresiran navođenjem adrese promenljive koja sadrži traženi podatak. Imajući u vidu činjenicu da se pri indirektnom adresiranju, isto kao i pri indeksnom, adresa promenljive određuje tek za vreme izvođenja programa, potrebno je i ovde voditi računa o zauzimanju meorije i mogućem prekoračenju granica.
Indeksnim adresiranjem se mesto u memoriji na kome se nalazi jedna promenljiva definiše *relativno* u odnosu na položaj neke druge promenljive. Pomeraj (ofset) promenljive sadržan je u 25. reči datoteke 2 (*status file*) tako da je adresa reči koja sadži pomeraj **S:24**. Ovaj način adresiranja je izuzetno pogodan ako je potrebno da se manipuliše sa nizom uzastopnih promenljivih. Indeksno adresiranje promenljive ostvaruje se navođenjem simbola **#** neposredno ispred adrese promenljive koja predstavlja bazu u odnosu na koju se izračunava indeksna adresa. Pomeraj može biti pozitivan ili negativan i u zavisnosti od tipa promenljive predstavlja broj reči

ili elemenata koje treba dodati baznoj adresi. Pri indeksnom adresiranju potrebno je obratiti pažnju na prethodni sadržaj datoteke kao i o prekoračenju granica datoteke.
Adresiranje datoteka - Pri realizaciji nekih algoritama, postoji potreba da se istovremeno ili na neki uređen način operiše sa nizom podataka koji je smešten u nekoj datoteci, ili sa datotekom u celini. U tu svrhu koristi se takođe indeksno adresiranje, ali se programski sistem sam stara o adekvatnoj promeni sadržaja indeksnog registra. Ukoliko se radi sa nizom podataka u datoteci onda se taj niz specificira tako što se definiše *bazna adresa #fn:e* gde je prvi ili nulti elemenat niza, što zavisi od konretne programske naredbe. Istovremeno se na odgovarajući način definiše i *dužina niza*. Rad sa podacima iz niza se odvija uz korišćenje indeksnog registra, s tim što, za razliku od naredbi u kojima se koristi indeksna adresa, korisnik ne mora da vodi računa o sadržaju indeksnog registra. Važno je jedino da se ima na umu da sve naredbe za rad sa datotekama menjaju sadržaj indeksnog registra.

9. Diskretni U/I moduli.

Od kontrolera se očekuje da obezbedi konverziju digitalnog (binarnog) signala koji dolazi sa senzora u numeričku vrednost 0 ili 1 i da taj podatak smesti kao jedan *bit* na odogovarajuće mesto u memoriji. Ova činjenica omogućila je projektovanje i izradu tipiziranih U/I kola koja su u stanju da obrađuju gotovo sve signale koji se sreću kod industrijske merne opreme i izvršnih organa. Pored toga, nekoliko U/I kola su grupisana zajedno i čine *diskretni U/I modul*, čija veza sa kontrolerom se ostvaruje jednostvanim ubacivanjem u odgovarajuci slot na šasiji. Na prednjoj ploči U/I modula nalazi se određeni broj pinova (*terminal points*) za koje se vezuju izlazi sa mernih instrumenata, odnosno ulazi u izvršne organe. Svaki pin je zapravo ulazna ili izlazna tačka odgovarjućeg kola za spregu sa kontrolerom. U skladu sa time svaki pin se identifikuje svojim *tipom* (ulaz ili izlaz) i *brojem* koji odredjuje položaj U/I kola u okviru modula, i koji zapravo predstavlja *adresu* pina. Pored U/I pinova, na prednjoj ploči modula nalaze se i pinovi koji su interno povezani sa napajanjem (DC ili AC), sa zajedničkom (nultom) tačkom i sa zemljom. Za vreme rada U/I modula, stanje svakog pina se prikazuje na odogovarajućem LED indikatoru. Postoje tri tipa U/I modula: *ulazni*, *izlazni* i *kombinovani ulazno/izlazn modul.* Oni se izrađuju sa različitim gustinama pinova (4, 8, 16 i 32 pina po modulu) i mogu se sprezati sa AC, DC i TTL naponskim nivoima. Maksimalni broj modula koji se može direktno povezati sa jednim kontrolerom, zavisi od veličine šasije i broja slobodnih slotova. Budući da svaki slot ima svoju adresu unutar šasije, to znači da je samim stavljanjem modula u slot određena i njegova adresa.

10. Sprezanje U/I modula sa PLC-om.

Vrednost binarnog signala koji dolazi na ulazni pin nekog U/I modula očitava se za vreme *ulaznog dela sken ciklusa*. U zavisnosti od toga da li očitana vrednost predstavlja *logičku nulu* ili *jedinicu* formira se odgovarajuća vrednost *bita* (0 ili 1) i upisuje na mesto u *datoteci 0* (*Input image file*) koje odgovara adresi ulaznog pina. Isto tako, vrednost bita koji treba da se prenese na izlazni pin kontrolera kao binarni signal, nalazi se u *datoteci 1* (*output image file*). Za vreme *izlaznog dela sken ciklusa* ova vrednost se očitava, konvertuje u odgovarajući signal i prenosi na izlazni pin čija adresa odgovara mestu u datoteci na kome se nalazi posmatrani bit. Svakom modulu koji nema više od 16 pinova pridružuje se po jedna *16-bitna reč* u datoteci 0 odnosno 1. Ako modul ima manje od 16 pinova, onda se ne koriste svi bitovi u pridruženoj reči. Ako modul im 32 pina, njemu se pridružuju dve susedne 16-bitne reči. Pri tome, koja reč će biti pridružena

modulu zavisi od slota u kome se modul nalazi.

11. Digitalni senzori i digitalni aktuatori (izvršni organi).

Digitalni signal je signal čija amplituda može imati jednu od konačnog broja različitih vrednosti. Posebna podvrsta digitalnog signala je binarni (diskretni) signal čija amplituda ima jednu od dve moguće vrednosti koje se kodiraju kao binarna nula i binarna jedinica (0 i 1). Ove vrednosti, u zavisnosti od određene aplikacije imaju značenje uključen/isključen, istinit/neistinit, itd. Osnovna komponenta diskretnog senzora i aktuatora je kontakt. U principu, kontakt može biti realizovan kao bilo koja vrsta prekidača, ili neka drugi elektro-mehanički, elektro-optički, pneumatski ili hidraulični uređaj koji ima dva stanja: *zatvoren* (*closed*) – provodi struju (odnosno propušta odgovarajuću fizičku veličinu) i *otvoren (open)* – ne provodi struju (odnosno ne propušta odgovarajuću fizičku veličinu). Senzori i kontakti koji se koriste kao ulazni procesni uređaji mogu biti otvarani i zatvarani kao rezultat nekog mehaničkog dejstva, čoveka, prisustvom ili odsustvom nekog objekta, promenom temperature, itd. Osnovni izlazni element je rele, koje se realizuje kao tranzistorsko rele (za mala opterećenja), elektromehaničko rele (za srednja opterećenja) i kao kontaktor (za velika opterećenja). Pored relea, često se koristi i solenoid – još jedan elektromehanički aktuator, čjim radom se upravlja pomoću elektromagnetne sile proizvedene u namotaju. Postoje dva tipa solenoidnih aktuatora: *Jednosmerni solenoid* koji ima samo jedan izvod za napajanje, tako da struja ima uvek isti smer, što znaci da se i jezgro pod dejstvom magnetne sile može pomerati samo u jednom smeru. U odsustvu napajanja solenoida, mehanicka opruga vraca jezgro u početni položaj. *Dvosmerni solenoid* – koji ima dva izvoda za napajanje, tako da smer struje, odnosno odgovarajuće magnetne sile zavisi od toga na koji izvod je priključeno napajanje. U skladu sa time i jezgro se kreće u jednom od dva moguća smera. Ukoliko se napajanje dovede na oba izvoda, jezgro se neće pomerati, Isto tako, ukoliko ni na jednom kraju nema napajanja, jezgro će ostati u zatečenom položaju.

12. Povezivanje digitalnih uređaja za odgovarajući modul PLC-a.

Prilikom povezivanja digitalnih uređaja važno je da se vodi računa o tome kako je uređaj projektovan, odnosno kakav treba da bude smer električnog signala. U tom smislu razlikuju se dve vrste uređaja:

- uređaji koji su izvor signala (source devices) – povezuju se na pozitivni pol izvora napajanja
- uređaji koji su primaoci signala (sinking device) – povezuju se na zajedničku tačku izvora napajanja.

Da bi se obezbedila kompatibilnost digitalnih uređaja i PLC-a za koji se oni vezuju, digitalni moduli se takođe proizvode u dve kategorije:

- *digitalna U/I kola koja su izvor signala* za uređaje koji su projektovani kao primaoci.
- *digitalna U/I kola koja su primaoci signala* za uređaje koji su projektovani kao izvor.

Potrebno je da se istakne da digitalni moduli koji predstavljaju izvor signala moraju u sebi da imaju i izvor napajanja. U tom slučaju, postojanje još jednog spoljnog izvora, je opciono. Za razliku od njih digitalni moduli koji primaju signale nemaju izvor napajanja. To znači da u kolu preko koga se vezuje digitalni uređaj mora da postoji spoljni izvor napajanja.

13. Analogni U/I moduli.

Analogni ulazni moduli su kola za spregu između kontinualnih (analognih) signala koji dolaze od mernih instrumenata i digitalnih (numeričkih) vrednosti kojima su ovi signali prikazani u PLC-u. Analogni izlazni moduli obezbeđuju spregu između numeričkih vrednosti u PLC-u i analognih signala koji predstavljaju ove vrednosti i koji služe za upravljanje izvršnim organima. U principu jedan modul se spreže sa više spoljnih uređaja, pričemu se svaka sprega posmatra kao jedan ulazni ili izlazni kanal. Moduli se međusobno razlikuju po broju i vrsti kanala. Neki moduli su samo ulazni ili samo izlazni, a neki su kombinovani, što zanči da imaju i izlazne i ulazne linije. Svaki modul se smešta u jedan slot na šasiji PLC-a. Otuda se, sa gledišta adresiranja, on tretira isto kao I digitalni modul, s tim što je značenje pinova i broj bitova koji odgovaraju jednom pinu drugačiji. Drugim rečima podaci koji se preko modula unose u računar nalaziće se u određenim lokacijama datoteke ulaza (I), a podaci koji se iznose iz računara, nalaziće se u datoteci izlaza (O).

14. Analogni ulazni kanali.

Nezavisno od toga koliko se ulaznih kanala nalazi na jednom modulu, modul, po pravilu, ima samo jedan A/D konvertor. U toku ulaznog sken ciklusa, uz pomoć multipleksera, odabira se jedan po jedan ulazni kanal na modulu, izvrši se konverzija odgovarajućeg signala i on se smešta u odgovarajuću reč u memorijskom području koje odgovara datoteci ulaza. Imajući u vidu da je merni signal uvek zašumljen, signal koji dolazi preko analognog kanala se posle konverzije propušta kroz digitalni filter koji ima za cilj da odbaci komponente visokih učestanosti koji potiču od šuma. Tip i vrsta ovog ugrađenog filtra zavisi od proizvođača. Kod nekih tipova modula korisniku može sam da podešava parametre filtra. Pored toga, neki analogni moduli pružaju mogućnost korisniku da dobije i informacije o prekoraćenju opsega ili o drugim aspektima rada modula. Anolgni ulazni kanali su, po pravilu, prilagođeni za standardizovane vrste signala (strujne ili naponske) i to u opsezima koji se najcešće sreću kod različitih analognih davaa. Najcešće je modul tako podešen da se postavljanjem internih prekidača može definisati da li će se kanal koristiti za strujni ili naponski signal. Napomenimo još, da najveci broj analognih modula interno skenira ulazne kanale i vrši A/D konverziju ulaznih signala daleko cešce nego što to zahteva sken ciklus PLC-a. Sve konvertovane vrednosti nalaze se u skupu internih registara PLC-a i bivaju zamenjene sanovim vrednostima posle sledece konverzije. Sadržaj tih registara, se me_utim, prebacuje u odgovarajuce podrucje datoteke ulaza samo u toku ulaznog sken ciklusa. i da imaju i ulazne i izlazne linije.

15. Analogni izlazni kanali.

Za razliku od analognih ulaza, svakom analognom izlaznom kanalu pridružen je poseban D/A konvertor. Pomoću njega se celobrojna vrednost koja se nalazi na odgovarajućem mestu u datoteci izlaza pretvara u strujni ili naponski signal. U principu, moduli se razlikuju po rezoluciji, ali se najčešće sreću konvertori čija je rezolucija 12 do 14 bitova. Kod posmatrane klase SLC kontrolera koriste se 14-bitni D/A konvertori. Otuda se svakom ulaznom kanalu pridružuje po jedna memorijska reč. Kao i analogni ulazi, i analogni izlazi se prave za standardizovane naponske i strujne signale. Pri tome, za razliku od ulaznih modula, ovde je svaki kanal unapred formiran za prenošenje ili naponskih ili strujnih signala.

16. Formiranje aplikacije (konfiguracija).

Formiranje jedne aplikacije započinje uvek specifikacijom samog kontrolera na kome će se data

aplikacija realizovati. To znači da korisnik mora da pruži informaciju o vrsti i tipu procesorskog modula koji će se koristiti, o ulazno izlaznim modulima koji će se postaviti u šasiji i o tipu računarske mreže u koju će taj kontroler biti vezan. Svaki proizvođač PLC-a razvija i posebni softverski alat koji omogućava da se na izuzetno jednostavan način definiše struktura PLC-a i formira odgovarajući program. Ovaj alat, koji kod najvećeg broja proizvođača koristi grafički interfejs (GUI), omogućava da se PLC emulira na standardnom PC računaru. To zapravo znači da se, sa gledišta korsinika, PC računar na kome je softverski alat instaliran ponaša kao PLC. Kada je program razvijen on se, posebnom tehnikom, prebaci na PLC (download programa). Ukoliko PLC ostane i dalje u vezi sa PC računarom, onda se isti softverski alat može koristiti da se pomoću PC-a prati izvršavanje formiranog algoritma i obavljaju eventualne korekcije.

17. Uvod u leder programiranje.

PLC je projektovan kao namenski mikroračunarski sistem za upravljanje i nadzor rada nekog procesa, i da u skladu sa tim ima poseban operativni sistem koji obezbeđuje periodično ponavljanje sken ciklusa, onda je logično očekivati da je za njegovo programiranje razvijen i poseban programski jezik. PLC je početno razvijen sa idejom da zameni relejne sisteme. To znači da se očekivalo da on realizuje odgovarajuću vremensku sekvencu logičkih operacija. Kada je reč o projektovanju relejnih sistema onda je zapravo potrebno da se reši problem grafičkog predstavljanja vremenske sekvence logičkih operacija. Klasični logički dijagrami ne pružaju mogućnost za prikazivanje različitih ulazno/izlaznih promenljivih kao funkcija vremena. Sa druge strane, vremenski dijagrami ne omogućavaju da se prikaže logika koja uslovljava te odnose. U cilju spajanja obe vrste prikazivanja, za projektovanje relejnih sistema razvijeni su leder (lestvicasti) dijagrami. Projektovanje PLC-ova je, dakle, podrazumevalo da se za njih mora razviti i odgovarajući programski jezik zasnovan na leder dijagrmima – *leder programski jezik*. Jedna programska linija leder jezika sastoji se iz niza grafičkih simbola (programskih naredbi) koji predstavljaju različite logičke elemente i druge komponente kao što su časovnici i brojači, koji su poređani duž horizontalne linije – *rang* (*rung*) – koja je na oba kraja spojena sa dvema vertikalnim linijama. Prema tome, leder dijagram ima izgled *lestvica*, odakle potiče i njegov naziv (*ladder – lestvice*).

Svaki rang leder dijagrama sastoji se iz *dva dela*. Na levoj strani ranga nalazi se *uslov* izražen u formi kontaktne (prekidačke) logike, dok se na desnoj strani ranga nalazi *akcija* koja treba da se izvrši ukoliko je *uslov ispunjen* (*true*). Uslov u leder dijagramu ima vrednost ON ili OFF, zavisno od statusa bita koji mu je dedeljen. Ukoliko se priključi taster na klemu koji mu odgovara moguće je menjati stanje bita iz stanja logičke jedinice u stanje logičke nule i obrnuto. Stanje ligičke jedinice najčešće se označava kao ’’ON ’’ a stanje logičke nule kao ’’OFF’’. Koji bi u bukvalnom prevodu značile ’’uključeno’’ i ’’isključeno’’. Desni deo leder dijagrama je instrukcija koja se izvršava u slučaju da je levi uslov ispunjen. Postoji više vrsta instrukcija koje bi se najlakše mogle podeliti na jednostavne i složene. Primer jednostavne instrukcije je aktiviranje nekog bita u memorijskoj lokaciji.

Uz svaki grafički simbol na leder dijagramu naznačava se i adresa promenljive koja predstavlja operand. Pri ispitivanju istinitosti smatra se da se nad svim simbolima u jednoj liniji (redna, serijska veza) obavlja logicka “I” operacija. U jednom rangu dozvoljena su i granjanja (paralelene veze). Pri ispitivanju istinitosti uslova paralelene veze se tretira kao logicko “ili” operacija.

18. Bit naredbe.

Bit naredbe su, kao što samo ime kaže naredbe čiji su operandi *bitovi*. Gledano potpuno opšte za vreme programskog skena u okviru bit naredbi ispituje se stanje pojedinog bita, ili se njegova vrednost postavlja na 1 (*set*) ili na 0 (*reset*).

Bit naredbe za definisanje uslova

Ove naredbe se postavljaju na levoj strani ranga i definišu uslov koji se odnosi na stanje bita čija je adresa definisana u naredbi. Kao rezultat izvođenja naredba dobija istinosnu vrednost *true* (*istinit*) ili *false* (*neistinit*) .

- *XIC - Examine if closed* (ispitivanje da li je kontakt zatvoren) - odnosi na *normalno otvoren prekidač* (ima vrednost 1 kada je prekidač pritisnut)
- *XIO - Examine if open* (ispitivanje da li je kontakt otvoren) - odnosi na *normalno zatvoren prekidač* (ima vrednost 1 kada prekidač nije pritisnut).

Bit naredbe za postavljanje vrednosti izlaza

Ovim naredbama se bitu čija je adresa navedena u naredbi dodeljuje vrednost 1 ili 0. Ove nardebe nalaze na desnoj strani ranga, što znači da će se one izvršiti samo ako je iskaz na levoj strani ranga istinit.

- *OTE - Output energize (pobuđivanje izlaza)* - ovom naredbom se vrednost bita čija je adresa "a" može promeniti samo jedanput za vreme sken ciklusa. Ova vrednost ostaće neizmenjena sve do sledećeg sken ciklusa, kada će se pri skeniranju odgovarajućeg ranga ponovo ispitati uslov i izvesti odgovarajuća akcija.
- *OTL - Output latch (pamćenje izlaza)* se adresovani bit može isključivo postaviti na 1. Kod *OTL* naredbe vrednost bita se postavlja (lečuje) na 1 u prvom skenu u kome je *uslov* istinit. Nakon toga ova naredba postaje neosetljiva na istinosnu vrednost *uslova*. To znači da će vrednost bita ostati neizmenjena bez obzira na to kako se menja vrednost *uslova*.
- *OTU - Output unlatch (resetovanje izlaza)* naredbom se adresovani bit može isključivo postaviti na 0. Pri tome, vrednost bita se postavlja (lečuje) na 0 u prvom skenu u kome je *uslov* ispunjen. Nakon toga ova naredba postaje neosetljiva na vrednost *uslova*.

Potrebno je da se istakne da se *OTL* i *OUT* naredba koriste uvek u paru, pri čemu se u obe naredbe adresira isti bit.

Bit triger naredba

- *OSR - One-shot rising (uzlazna ivica)* - ova naredba omogućava da se obezbedi izvođenje neke akcije samo jedanput. Potrebno je da se istakne da je ovo specificna naredba koja istovremeno pripada i kategoriji uslova i kategoriji akcije. Kada se u toku sken ciklusa detektuje da je uslov promenio svoju vrednost sa neistinit na istinit (uzlazna ivica) onda OSR naredba takođe dobija vrednost istinit (što ovu naredbu svrstava u kategoriju naredbi uslova). Istovremeno se i bitu cija je adresa pridružena toj naredbi dodeljuje vrednost 1 (po cemu se ova naredba svrstava i u kategoriju akcija). Obe ove vrednosti ostaju nepromenjene do sledeceg sken ciklusa.

19. Programski sken i vremenski dijagram.

U toku programskog skena procesor izvršava pojedinačne naredbe, obrađujući rang po rang od početka pa do kraja programa. U okviru obrade jednog ranga, procesor ispituje stanja bitova u datotekama podataka, određuje vrednost pojedinačnih naredbi uslova, izvršava logičke I i ILI

operacije nad tim vrednostima, u skladu sa načinom na koji je formiran desni deo ranga, i kao rezultat tih operacija određuje istinosnu vrednost *uslova*. Ukoliko je ova vrednost *istinit*, procesor će izvršiti naredbe koje se nalaze na levoj strani ranga i koje predstavljaju *akciju*. Ulazna linija vezana je za nulti pin ulaznog dela kombinovanog U/I modula, smeštenog u slotu 1 PLC-a. Na osnovu stanja ulazne linije generiše ssignal na izlaznoj liniji koja je vezana za nulti pin izlaznog dela istog U/I modula. Predpostavljeno je da ceo program ima više ulaznih signala i više izlaznih signala čije očitavanje, odnosno generisanje zahteva određeni period vremena za ulazni i izlazni skeninterval. Isto tako, predpostavljeno je i da se program sastoji od više rangova, čija obrada zahteva neki period vremena (programski sken interval). Stanje signala na ulaznoj liniji se može promeniti u bilo kom trenutku. Međutim, vrednost bita pridružena toj liniji biće promenjena tek u toku prvog ulaznog sken ciklusa koji nastupa posle promene stanja ulazne linije. Za vreme programskog skana vrednost XIC naredbe se određuje na osnovu stanja odgovarajućeg bita, a ne njemu odgovarajućeg ulaznog signala. Otuda je u prvom sken ciklusu vrednost ove naredbe 0 iako je ulazni signal već u stanju logičke jedinice.

20. Naredbe za merenje vremena i prebrojavanje događaja-casovnik i brojac.

Prebrojavanje događaja obavlja ***brojač*** (***counter***), koji nakon registrovanja unapred zadanog broja događaja generiše odgovarajući signal. Merenje vremena ostvaruje se pomoću ***časovnika*** (***timer***). U suštini časovnik izražava vreme kao multipl određenog osnovnog intervala (*vremenska baza*). To zapravo znači da časovnik radi kao brojač protoka osnovnih intervala i da nakon isteka određenog, unapred zadanog intervala vremena, generiše odgovarajući signal. Merenje protoka vremena i prebrojavanje događaja u okviru kontrolera može se realizovati hardverski pomoću odgovarajućih računarskih komponenti (modula) ili softverski (programski). Hardverska realizacija podrazumeva da kontroler ima posebni modul koji ostvaruje funkciju časovnika i brojača. Korisnik odgovarajućim naredbama definiše parametre modula i u toku izvršavanja programa kontroliše njegov rad. U slučaju softverske realizacije, ovu funkciju ostvaruje posebni programski modul, koji korisnik, po potrebi, uključuje u svoj program i odgovarajućim naredbama upravlja njegovim radom.
Razlika se ogleda samo u funkcionalnom smislu. Ukoliko su časovnik i brojač hardverski realizovani oni svoju funkciju obavljaju autonomno, što znači da ne koriste procesor za svoj rad dok softverski realizovani časovnik i brojač za izvođenje svojih funkcija zahtevaju izvesno procesorsko vreme. Hardverska realizacija, bar u principu, omogućava brže ponavljanje događaja od softverske.
Pri korišćenju časovnika i brojača neophodno je da se definišu sledeći parametri:

- *Vremenska baza (time base)* određuje dužinu *osnovnog intervala vremena*. Vremenska baza je definisana kao 0.01 sec.
- *Zadana vrednost (preset value - PRE)* je vrednost kojom se definiše željeni broj osnovnog intervala vremena (čime se određuje ukupno vreme koje časovnik treba da izmeri), odnosno ukupni broj događaja koje brojač treba da registruje pre nego što se generiče signal koji označava da su časovnik ili brojač završili rad.
- *Akumulirana vrednost (accumulated value - ACC)* predstavlja broj osnovnih vremenskih intervala koje je časovnik izbrojao, odnosno broj događaja koje brojač registrovao u nekom trenutku. Kada akumulirana vrednost postane veća ili jednaka od zadane vrednosti časovnik, odnosno brojač, završavaju svoj rad.

21. Realizacija casovnika.

S obzirom da je časovnik realizovan softverski, parametri koji definišu njegov rad moraju biti smešteni u memoriji kontrolera. Za pamćenje podataka o časovnicima koristi se datoteka podataka broj 4 (*timer file – T*). U ovoj datoteci može se definisati najviše 256 različitih časovnika. Ukoliko je potrebno da se koristi veći broj časovnika, korisnik može definisati i dodatne datoteke (*korisnički definisane datoteke*) čiji su brojevi od 9 do 255. Svakom časovniku pridružuju se po jedan *element* u odgovarajućoj datoteci. *Osnovni element* ovih datoteka sastoji se od tri 16-bitne reči:

- *Reč 0* je kontrolna reč koja sadrži tri bita koja ukazuju na stanje časovnika, kao i bitove za interno upravljanje radom časovnika
- *Reč 1* sadrži *zadanu vrednost* (PRE)
- *Reč 2* sadrži akumuliranu vrednost (ACC)

22. Naredbe casovnika.

Timer on-delay (TON)

Ova naredba se unosi direktno u ledder program i njome se definiše prva vrsta časovnika. TON naredba započinje rad časovnika (prebrojavanje osnovnih vremenskih intervala) za vreme onog programskog sken ciklusa u kome *uslov* u rangu u kome se naredba nalazi prvi put postaje *istinit.* U svakom sledećem sken ciklusu, sve dok je *uslov istinit* časovnik vrši ažuriranje akumulirane vrednosti (ACC) u skladu sa proteklim vremenom između dva ciklusa. Kada akumulirana vrednost dostigne zadanu vrednost, časovnik prekida svoj rad i postavlja DN bit na 1. Pri tome, ako u nekom sken ciklusu *uslov* postane *neistinit*, časovnik prekida svoj rad i akumulirana vrednost se postavlja na 0, bez obzira da li je časovnik pre toga izmerio zahtevano vreme ili ne.

Timer off-delay (TOF)

Ovom naredbom se definiše druga vrsta časovnika. TOF započinje rad časovnika za vreme onog programskog sken ciklusa u kome uslov u rangu u kome se naredba nalazi prvi put postaje neistinit. U svakom sledećem sken ciklusu, sve dok je uslov neistinit časovnik vrši ažuriranje akumulirane vrednosti u skladu sa proteklim vremenom između dva sken ciklusa. Kada akumulirana vrednost dostigne zadatu vrednost, časovnik prekida svoj rad i postavlja svoj izlaz na 1. Pri tome, ako u nekom sken ciklusu uslov postane istinit, časovnik prekida svoj rad i akumulirana vrednost se postavlja na 0 bez obzira da li je časovnik pre toga izmero zahtevano vreme ili ne.

Retentive Timer (RTO)

Ovom naredbom se definiše treća vrsta časovnika. RTO naredba se razlikuje od TON naredbe samo po tome što se akumulirana vrednost ne resetuje, već zadržava i onda kada uslov postane neistinit. Drugim rečima, ovaj časovnik počinje sa radom kada uslov postane istinit, i nastavlja sa radom povećavajući akumuliranu vrednost sve dok je uslov istinit. Kada uslov postane neistinit, časovnik prekida svoj rad, ali se akumulirana vrednost pri tome ne menja. To znaci da ce kada uslov ponovo postane istinit, časovnik nastavi sa radom i prethodno izmerenom vremenu dodavati nove vrednosti. Na taj način ovaj časovnik omogućuje da se kumulativno mere intervali vremena u kojima je uslov bio istinit.

Reset naredba (RES)

Reset naredba je naredba akcije i koristi se za resetovanje časovnika. Kada je uslov istinit ova naredba se izvršava tako što se u časovniku čija je adresa navedena u RES naredbi, resetuju na

nulu svi bitovi kao i akumulirana vrednost. S obzirom na način rada očigledno je da se RES naredba ne sme koristiti za TOF tip časovnika.

23. Nacin rada casovnika.

Sve dok časovnik radi u svakom sken ciklusu povećava se akumulirana vrednost. Pri tome, iznos za koji će se povećati ACC vrednost zavisi od dužine trajanja sken ciklusa. Naime, kada se prilikom obrade ranga ustanovi da su se stekli uslovi da časovnik počne sa radom onda se istovremeno startuje jedan interni časovnik, koji se ažurira preko prekida (interapta) na svakih 0,01 sec.
Broj registrovanih vremenskih intervala se smešta u interni 8-bitni registar (bitovi 0-7 u prvoj reči). Ukoliko je u pitanju časovnik čija je vremenska baza 0,01 sec, onda se u sledećem programskom skenu, kada se naiđe na dati rang, vrednost internog registra, koja zapravo predstavlja interval vremena koji je protekao između dva sukcesivna sken-a, dodaje akumuliranoj vrednosti. Nakon toga se interni rgistar resetuje na 0 i počinje ponovo da meri vreme do sledećeg skena. Budući da je maksimalna vrednost koju može da ima interni registar oko 2,5 sec (255x0,01), može se očekivati da će tajmer raditi ispravno samo ako sken ciklus ne traje duže od 2,5 sekundi.
Ukoliko se tajmer koristi u programu čiji sken ciklus traje duže, onda je neophodno da se ista naredba za časovnik postavi na više mesta u programu čime
će se obezbediti da se rangovi koji sadrže taj časovnik obrađuju sa učestanošću koja nije veća od 2,5 sekundi.

24. Realizacija brojaca.

Budući da je brojač, isto kao i časovnik, realizovan softverski, parametri koji definišu njegov rad moraju biti smešteni u memoriji kontrolera. Za pamćenje podataka o brojačima koristi se datoteka podataka broj 5 (*counter file – C*). U ovoj datoteci može se definisati najviše 256 različitih brojača. Ukoliko je potrebno da se koristi veći broj brojača, korisnik može definisati i dodatne datoteke (*korisnički definisane datoteke*) čiji su brojevi od 9 do 255.
Svakom brojaču pridružuju se po jedan *element* u odgovarajućoj datoteci. *Osnovni element* ovih datoteka sastoji se od tri 16-bitne reči:

- *Reč 0* je kontrolna reč koja sadrži 6 bitova koji ukazuju na stanje brojača.
- *Reč 1* sadrži *zadanu vrednost* (PRE)
- *Reč 2* sadrži akumuliranu vrednost (ACC)

Postoje dva osnovna tipa brojača *brojač unapred* (*CTU – count up*) i *brojač unazad* (*CTD – count down*) i oba koriste istu datoteku (Sl. 5). Isto kao i kod časovnika i brojaču i pojedinim bitovima mogu se umesto adrese dodeliti simbolička imena.

25. Naredbe brojača.

Naredbe za oba tipa brojača su naredbe *akcije*, što znači da se smeštaju u desni deo ranga. Oba brojača broje promenu vrednosti *uslova sa neistinit na isitinit* (uzlazna ivica). Pri svim ostalim vrednostima *uslova*, oni zadržavaju prebrojani iznos i čekaju sledeći prelaz. Drugim rečima, brojači se niti puštaju u rad, niti zaustavljaju. Oni neprekidno rade i beleže (broje) svaki prelaz *istinit/neistinit*. Dostizanje zadane vrednosti se signalizira postavljavanjem odgovarajućeg bita – *done bit* (DN) – na 1, ali se brojanje i dalje nastavlja. Prebrojani iznos se može izbristai jedino posebnom *RES* naredbom.

Count up (CTU)

Bitovi stanja brojača menjaju se u toku programskog sken ciklusa na sledeći nacin: *OV - Count up overflow bit* se postavlja na 1 kada akumulirana vrednost (ACC) prelazi sa 32767 na –32768 (u binarnoj aritmetici drugog komplementa, sa 16-bitnom reci 32767+1= –32768), i nastavlja brojanje unapredi. *DN - done bit* se postavlja na 1 kada je ACC = PRE; *CU - Count up enable bit* se postavlja na 1 kada je uslov istinit, a resetuje na 0 kada je *uslov neistinit* ili kada se aktivira odgovarajuća *RES* naredba.

Count down (CTD)

Bitovi stanja brojača menjaju se u toku programskog sken ciklusa na sledeći način: *UN- Count down underflow bit* se postavlja na jedan kada akumulirana vrednost (ACC) prelazi sa – 32768 na 32767 (u binarnoj aritmetici drugog komplementa, sa 16 bitnom rec i nastavlja da broji unazad od te vrednosti. *DN - done bit* se postavlja na 1 kada je ACC = PRE; *CU - Count up enable bit* se postavlja na 1 kada je uslov istinit, a resetuje na 0 kada je uslov neistinit ili kada se aktivira odgovarajuca RES naredba.

Reset naredba (RES)

RES naredba je naredba akcije i koristi se za resetovanje brojaca. Kada je uslov istinit ova naredba se izvršava tako što se u brojacu cija je adresa (ili simbolicko ime) a navedeno u RES naredbi, postavljaju na nulu svi indikatorski bitovi, kao i akumulirana vrednost (ACC).

26. Naredbe za operacije nad podacima (operandi, operacije).

Operandi

Kao što je već rečeno, promenljive se u memoriji kontrolera pamte kao *numerički podaci* ili *alfanumerički podaci – stringovi*. Numerički podaci se pri tome mogu pamtiti kao *celoborjne* vrednosti (*integers*) ili *decimalni brojevi prikazani u tehnici pokretnog zareza* (*floating point*). Različiti tipovi numeričkih podataka smeštaju se u *datoteke podataka* odgovarajućeg tipa. Operandi mogu biti promenljive iz bilo koje datoteke. Pored promenljivih, operandi u pojedinim operacijama mogu biti i *programske konstante* – nepromenljive veličine koje se definišu eksplicitnim navođenjem vrednosti u okviru naredbe. Pri tome, nije dozvoljenno da oba operanda budu programske konstante. Samo se po sebi razume da se programska konstanta ne može korsititi kao rezultat.

Operacije

Operacija koja treba da se izvrši nad operandima definiše se u okviru naredbe. Najveći broj ovih naredbi pojavljaju se kao *naredbe akcije*. Drugim rečima, sam proces izračunavanja predstavlja jednu *akciju*, čije izvršavanje može biti uslovljeno istinosnom vrednošću nekog *uslova* koji se nalazi u levom delu ranga. Izuzetak su jedino *naredbe za poređenje*, koje opet, po svojoj prirodi, proveraju da li je neka *relacija* između operanada ispunjena ili nije odnosno da li njena vrednost *istinita* ili *neistinita*. Shodno tome, takve naredbe moraju biti *naredbe uslova*, tako da je rezultat njihovg izvođenja istinosna vrednost naredbe.

27. Naredbe za poređenje.

Naredbe za poređenje su naredbe *uslova*. U okviru ovih naredbi proverava se stinosna vrednost relacije između dva operanda. Kao rezultat provere naredba dobija vrednost *istinit* ili *neistinit*.

EQU - Equal (jednako)

NEQ - Not equal (nejednako)

LES - Less than (manje)

LEQ - Less than or equal (manje ili jednako)
GRT - Greater than (veće)
GEQ - Greater than or equal (veće ili jednako)

Pored navedenih naredbe među nardbama za poređenje postoje i sledeđe dve naredbe:

- *MEQ - masked comparison for equal (ispitivanje jednakosti pojedinih bitova)* - Ova naredba služi za poređenje delova pojedinih reči.
- *LIM – Limit test (ispitivanje granica)* - naredbom se proverava da li se vrednost operanda *Test* nalazi unutar datih granica. Ako je *donja granica* manja od *gornje granice*, vrednost naredbe je *istinita* ako operand pripada segmentu koji određuju granice.

28. Matematičke naredbe.

Kako im i samo ime kaže, *matematičke naredbe* služe za realizaciju različitih operacija nad operandima. Ove naredbe su naredbe *akcije* i u najvećem broju slučajeva imaju *dva operanda*. Izvršavanjem naredbe obavlja se zahtevana matematička operacija nad operandima i dobija rezultat čija se vrednost pamti. Operandi mogu biti programske poromenljive ili konstante, s tim što oba operanda ne mogu biti konstante.

Postavljanje indikatorskih bitova

Tok izvođenje zahtevane operacije u smislu prekoraćenja dozvoljenog opsega brojeva, prenosa, pokušaja deljenja sa nulom itd, može se pratiti preko vrednosti *indikatorski* bitova u *datoteci 2 (Status)*, koje se automatski postavljaju kad se naredba izvršava. Pri tome se postavljaju sledeći bitovi.

- *S:0/0 – Carry bit (C)* - ovaj bit se postavlja na 1 kada pri obavljanju operacije dolazi do prenosa bita iz najviše, 15. lokacije, u protivnom *bit C* ima vrednost 0.
- *S:0/1 – Overflow bit (V)* - ovaj bit se postavlja na vrednost 1 onda kada je rezultat matematičke operacije premašio dozvoljeni opseg brojeva, ili ako je operacijom zahtevano deljenje sa nulom, u protivnom vrednost *bita V* je 0.
- *S:0/2 – Zero bit (Z)* - ovaj bit se postavlja na vrednost 1 ako je rezultat zahtevane operacije jednak nuli, u protivnom *Z bit* ima vrednost 0.
- *S:0/3 – Sign bit (S)* - ovaj bit se postavlja na vrednost 1 ako je rezultat matematičke operacije negativan, (odnosno ako bit 15 rezultata ima vrednost 1), u protivnom vrednost *bita S* je nula.
- *S:5/0 – Overflow trap bit* - ovaj bit se postavlja na 1 onda kada je overflow bit *V* bit postavljen na 1. Pri tome, ako se vrednost bita S:5/0 ne resetuje na nulu, pre završetka programskog sken ciklusa, operativni sistem će signalizirati da je došlo do *popravljive greške* (kod greške 0020).

Izvođenje operacija celobrojnog množenja i deljenja – matematički registar S:13 i S:14

Da bi se omogućio ispravan rad i u ovom slučaju, koriste se dve reči *datoteke 2 (Status)* i to na **S:13** i **S:14** koje predstavljaju *32-bitni matematički registar*. Naime, prilikom izvođenja množenja, rezultat se dobija u matematičkom registru i to tako što reč na adresi S:13 sadrži prvih (manje značajnih) 16 bitova rezultata, dok S:14 sadrži preostalih (više značajnih) 16 bitova rezultata. Pri tome, se donji, manje značajni, deo rezultata istovremeno smešta i na adresu koja je u naredbi navedana kao promenljiva u kojoj se čuva rezultat.

32-bitno sabiranje i oduzimanje – bit S:2/14

Procesori SLC 5/02 (počev od serije C) i SLC 5/03 i SLC5/04 omogućavaju da se realizuje sabiranje i oduzimanje 32-bitnih podataka. Ovo se ostvaruje postavljanjem bita **S:2/14** na vrednost 1.

29. Pregled matematickih naredbi.

U odnosu na broj operanada i tip operacije koja se izvršava, matematičke naredbe se mogu podeliti u nekoliko grupa.

Aritmetičke i logičke binarne operacije

ADD – Add (sabiranje) , SUB - Subtract (oduzimanje) , MUL - Multiply (množenje) , DIV - Divide (deljenje) , XPY - X to the power of Y , AND -And (logičko "I") , OR - Or (logičko "ILI") , XOR - Exclusive OR (ekskluzivno "ili").

Unarne operacije

NEG - Negate (negacija) , NOT - Not (komplement) , DDV - Double divide(deljenje 32-bitnog celog broja iz mat. reg. sa 16-bitnim operandom) , SQR - Square Root (kvadratni koren) , ABS - Absolute (apsolutna vrednost) , SIN – Sine (sinus) , COS –Cosine (kosinus) , TAN – Tangent (tanges) , ASN - Arc Sine (arkus sinus) , ACS - Arc Cosine (arkus kosinus) , ATN - Arc Tangent (arkus tanges) , LN - Natural log (prirodni logaritam) , LOG - Log to the base 10 (dekadni logaritam).

Složene matematičke naredbe

CPT – Compute (izračunavanje aritmetičkog izraza) , SCP – Scale with parameters (parametarsko skaliranje podatka) , SCL - Scale data (skaliranje podatka) , RMP – Ramp instruction (generisanje signala)

30. Naredbe za manipulaciju sa numeričkim podacima.

Naredbe za manipulaciju sa podacima služe za definisanje vrednosti promenljivih ili za određene izmene u formi prezentacije podataka. U tom smislu one se ne razlikuju bitno od matematičkih naredbi. Naime, nema nikakve sumnje da se matematičkim naredbama takođe vrši određena manipulacija sa podacima. Izdvajanjem ovih naredbi u posebnu grupu se zapravo želi naglasiti specifičnost oblika same naredbe i obrade podataka koja se njima vrši.

Naredbe za postavljanje vrednosti

CLR – Clear (postavi na nulu) , MOV – Move (postavljanje vrednosti promenljive) , MVM – Masked move (postavljanje vrednosti pojedinih bitova)

Naredbe za konverziju

DEG - Radians to degrees (radiani u stepene) , RAD - Degrees to radians (stepenei u radiane), TOD - To BCD (u BCD kod) , FRD - From BCD (iz BCD koda) , DCD - Decode 4 to 1 of 16 (dekodiranje 4 – 16) , ENC - Encode 1 of 16 to 4 (kodiranje 16 – 4)

Naredba za premeštanje bajtova

SWP – Swap (zamena bajtova)

31. Transformacija analognih ulaznih signala.

Analogni signali koji dolaze sa mernih instrumenata su standardizovani električni signali koji reprezentuju neke fizičke veličine. Ovi signali se prilikom A/D konverzije pretvaraju u celobrojne vrednosti, koje se zatim mogu dalje pretvoriti u odgovarajuće fizičke veličine. Drugim rečima signal prolazi kroz niz transformacija koje sve zavise od granica signala u svakoj fazi transformacije, te se mogu definisati na potpuno opšti način.

32. Transformacija analognih izlaznih signala.

Kao i kod ulaznog signala očigledno je da je i pri formiranju izlaznog signala neophodno da se izvrši jedan broj transformacija. U suštini analogni izlazni signal je najčešće neka fizička veličina koja pretstavlja upravljački signal u nekom sistemu. Ova veličina je, po pravilu, linearno srazmerna nekom električnom (strujnom ili naponskom) signalu koji pretstavlja pobudni signal za odgovarajući izvršni organ.

33. Naredbe za rad sa datotekama podataka (kreiranje datoteke, translacija bitova).

Naredbe za kreiranje datoteke

- COP – Copy file (kopiranje datoteke)
- FLL – Fil file (punjenje datoteke)

Ove naredbe se izvršavaju ukoliko je *uslov istinit* i to tako što se elementima datoteke *dest* pridružuju određene vrednosti. Pri tome se pri izvršavanju *COP* naredbe ove vrednosti uzimaju iz neke druge datoteke, označene kao *source.* Prilikom izvršavanja *FLL* naredbe jedan isti podatak, koji se nalazi na adresi *source* se prenosi u sve elemente odredišne datoteke i to počev od bazne adrese *#fn:e* zaključno sa adresom *#fn:*(*e*+*length*-1).

Naredbe za translaciju bitova

- BSL – Bit shift left (translacija bitova u levo)
- BSR – Bit shift right (translacija bitova u desno)

Naredbe za translaciju bitova su naredbe akcije. Međutim, one se izvode samo kada se *uslov* menja sa *neistinit* na *istinit*. Prilikom izvršavanja naredbe niz od *length* bitova koji je smešten u datoteci *fn*, počev od nultog bita u elementu *e*, se translatorno pomera za jedno mesto u levo (BSL) odnosno u desno (BSR), pri čemu se na upražnjeno mesto smešta bit čija je adresa *fn:w/b* dok se prvi (BSR), odnosno poslednji (BSL) bit niza prebacuje u indikatorski bit UL . Posle izvođenja ove naredbe *indeksni registar S:24* se postavlja na nulu.

34. Naredbe za sekvencijalnu obradu podataka.

Jedan od izuzetno čestih zadataka pri upravljanju procesima je *sekvencijalno upravljanje*. Ovim upravljanjem se izvršnim organima na procesu zadaje niz naredbi binarnog tipa (uključi/isključi, napred/nazad, kreni/stani i sl.) koje se smenjuju u vremenu, pri čemu svaka aktivnost traje određeni, unapred definisani interval vremena, ili dok se ne detektuje nastanak nekog događaja. Bitno obeležje ovog načina upravljanja je da je sekvenca unapred potpuno određena i da se niz aktivnosti može definisati kao sukcesivan niz binarno kodiranih reči, kod kojih se svaki bit odnosi na pojedini izvršni organ, koji je vezan za kontroler preko odgovarajućeg digitalnog izlaznog modula. Kako se proces odvija, tako se na izlazni modul prenosi reč po reč iz upravljačke sekvence. Budući da prelazak sa jedne aktivnosti na drugu može da bude uslovljen

stanjem u pojedinim delovima procesa, to znači da je neophodno da se, pod određenim uslovima, očitavaju stanja indikatora na procesu i porede sa unapred definisanim stanjima. U zavisnosti od rezultata poređenja, odlučuje se da li je došlo vreme za sledeću aktivnost. Kada je odgovor potvrdan, onda je izvesno da proces ulazi u sledeću fazu, te da se nadalje stanje mora porediti sa drugim nizom vrednosti koji ukazuje na završetak sledeće faze.

35. Naredbe za sekvencijalni rad sa datotekama.

U okviru ovih naredbi bar jedan od operanada je datoteka u kojoj se nalazi niz podataka. Pri tome se dozvoljava rad samo sa onim datotekama čiji elementi su dužine jedne reči. Adrese pojedinih podataka određuju se pomoću *bazne adrese* koja se definiše u naredbi i *pointera* koji predstavlja upravljački paramater, čija se vrednost menja u toku ponovljenih izvršavanja naredbe. Pri tome se adresa operanda dobija kao zbir *bazne adrese* i vrednosti *pointera*. U naredbama se definiše početna vrednost pointera kao i ukupna dužina niza.

- *SQL – Sequencer Load (sekvencijalno punjenje datoteke)* - svaki put kada se uslov menja sa *neistinit* na *instinit*, ova naredba se izvršava tako što se vrednost pointera poveća za 1 i podatak koji je određen kao *source* prenese u datoteku *file* na onu adresu na koju pokazuje pointer.
- *SQO – Sequencer output (sekvencijalno upravljanje)* - svaki put kada se uslov menja sa *neistinit* na *instinit*, ova naredba se izvršava tako što se vrednost pointera (*position*) poveća za 1 i uzme ona reč iz datoteke *file* (*#fn:w*) na koju pokazuje pointer.
- *SQC – Sequencer compare (sekvencijalno poređenje)* - svaki put kada se uslov menja sa *neistinit* na *instinit*, SQC naredba se izvršava tako što se vrednost pointera (*position*) poveća za 1 i uzme ona reč iz datoteke *#fn:w* na koju pokazuje pointer.

36. Naredbe za formiranje steak (punjenje, pražnjenje steaka, itd.).

U računarskoj terminologiji *stek* označava niz podataka koji se sekvencijalno puni i prazni. Naime, stek se formira u nekom području memorije i to tako što se definiše početna adresa steka i pointer steka se postavi na tu početnu adresu. Svaki put kada se podatak unese u stek, vrednost pointera poraste za 1, tako da on uvek ukazuje na sledeću slobodnu lokaciju. U pogledu uzimanja podataka iz steka postoje dva principa:

- LIFO stek (Last in last out) – je stek kod koga se podatak uzima sa "vrha" steka, odnosno kod koga se kao *prvo* uzima podatak koji je *poslednji* smešten u stek. Vrednost pointera se pri tome smanjuje za 1, ali se položaj preostalih podataka ne menja.
- *FIFO stek (First in first out)* – je stek kod koga se podatak uzima sa "dna" steka, odnosno kao *prvo* uzima podataka koji je *prvi* stavljen na *stek*, pri čemu se svi preostali podaci transliraju za jedno mesto na dole prema dnu steka, dok se memorijsko mesto koje je zauzimao podatak na vrhu steka postavlja na 0. Istovremeno se i vrednost pointera smanjuje za 1

Punjenje steka

- *LFL – LIFO load (punjenje LIFO steka)*
- *FFL – FIFO load (punjenje FIFO steka)*

Ove naredbe se izvršavaju svaki put kada se *uslov* menja sa *neistinit* na *istinit*, i to tako što se podatak čija je adresa navedena kao *source*, prenese na položaj u steku koji je određen pointerom. Nakon toga se vrednost pointera poveća za 1.

Pražnjenje steka

- *LFU – LIFO unload (pražnjenje LIFO steka)*
- *FFU – FIFO unload (pražnjenje FIFO steka)*

Ove naredbe se izvršavaju svaki put kada se *uslov* menja sa *neistinit* na *istinit*, i to tako što se vrednost pointera smanji za 1 i uzme podataka sa vrha steka (LFU), odnosno sa dna steka (FFU) i prenese na adresu koja je navedena kao *dest*.

37. Naredbe za upravljanje izvršavanjem programa.

U principu kada PLC počne sa radom (uđe u tzv *Run mode*) on započinje sken ciklus koji se satoji iz ulaznog sken ciklusa, programskog sken ciklusa, izlaznog sken ciklusa, komunikacionog ciklusa i održavanja. Pri tome se u okviru programskog sken ciklusa obrađuje rang po rang u redosledu u kome su oni napisani. Različite aplikacije mogu zahtevati da se redosled izvođenja leder programa, pod određenim uslovima, promeni. Isto tako može biti potrebno da se program privremeno prekine, da se pojedine aktivnosti suspenduju, da se stanja nekih veličina resetuju ili da se usled nastanka nekih događaja preduzimaju i neke druge aktivnosti. Svi ovi efekti mogu se ostavriti posebnim naredbama za upravljanje izvršavanjem programa.

Naredba za skok

Leder program se izvršava u okviru programskog sken ciklusa i to tako što se obradjuje rang po rang u redosledu u kome su oni napisani u programu. U samom procesoru postoji jedan registar, koji igra ulogu *pokazivača* (*pointera*), koji sadrži memorijsku adresu sledećeg ranga u programu koji treba da se obradi. U svakom rangu ispituje se istinitost uslova i ako je on istinit izvršavaju se naredbe akcije.

- *LBL - naredba* - sve dotle dok se naredbe programa izvršavaju u redosledu u kome su napisane nema nikakve potrebe da se pojedini rangovi posebno označe. Međutim, ukoliko se od programa očekuje da omogući skok na neku nardebu, onda je neophodno da se omogući da se jedna određena naredba identifikuje na nedvosmislen način.
- *JMP – Jump (skok)* - ova naredba je naredba akcije, što znači da se izvršava ukoliko je uslov istinit. Naredba se izvodi tako što se menja vrednost pointera tako da on ukazuje na memorisjku adresu na kojoj se nalazi rang čija je labela naznačena u JMP naredbi.

38. Potprogrami.

Veoma često u okviru neke aplikacije javlja se potreba da se jedna ista sekvenca naredbi ponovi više puta na različitim mestima u programu. Da bi se to izbeglo, dati niz rangova formira se samo jedanput kao *podprogram (subroutine)*, koji se poziva na izvršavanje na više mesta u leder programu.

Poziv potprograma

- *JSR – Jump to Subroutine (skok na podprogram)* – ova naredba je naredba akcije. Ukoliko je uslov istinit JSR naredba prouzrokuje prekid u normalnom izvršavanju leder programa i ostvaruje skok na podprogram čije je simboličko ime (broj programske datoteke) naveden kao adresa u JSR naredbi. Izvršavanje programa nastavlja se od prvog ranga podprograma.
- SBR – Subroutine (podprogram) –je zasebna celina leder programa koja se mora formirati u okviru posebne programske datoteke. Broj te datoteke (3 – 255) predstavlja istovremeno i simbolicko ime potprograma. SBR naredba se koristi da bi se naznačilo da programska datoteka predstavlja podprogram.

Završetak potprograma

- RET – return (povratak)
- END – End (kraj)

Poslednji rang svakog potrpograma, kao i glavnog programa, sadrži samo jednu naredbu akcije – END naredbu. Pri tome, u delu za uslov nema nikakve naredbe, što znači da se ovaj rang izvršava u svakom programskom sken ciklusu. Izvršavanje END naredbe u potprogramu ima za posledicu da se promeni vrednost pointera sledećeg ranga i to tako da on ukazuje na prvi rang koji se nalazi neposredno iza JSR naredbe kojom je ostvaren skok na ovaj potprogram.

39. Promena toka sken ciklusa.

Ova grupa naredbi koristi se u fazi testiranja programa ili za ubrzavanje sken ciklusa u nekimslučajevima.

- *TND – Temporary end* - ukoliko je uslov istinitova naredba prekida izvršavanje programskog sken ciklusa. U tom slučaju odmah započinje izlazni sken ciklus, posle koga se nastavlja ciklus komunikacija i održavanja. Sledeći programski sken ciklus započinje od prve programske naredbe.
- *SUS – Suspend naredba* - ukoliko je uslov istinit ova naredba prouzrokuje suspenziju rada procesora (*suspend idle mode PLC-a)* i ukidanje pobude na svim izlaznim linijama.
- *MCR – Master control reset* - par naredbi MCR i END MCR definiše zonu unutar leder programa koja se može izvršavati na specifičan način. Ukoliko je uslov u MCR naredbi *neistinit* onda se sve naredbe koje nalaze u MCR zoni izvršavaju regularno.

40. Ažuriranje ulaznih i izlaznih podataka.

Rezultati obrade podataka se ne prenose direktno na izlazne linije već se upisuju u datoteku izlaza iz koje će, u toku izlaznog sken ciklusa, biti prenete na izlazne linije. To zapravo znači da postoji neko kašnjenje između ocitavanja podataka i njihove obrade, kao i između definisanja izlaznih signala i njihovog prenošenja na izvršne organe. Kašnjenje nekih signala može prouzrokovati ozbiljnije poremećaje u efikasnosti upravljačkog algoritma. Da bi se to izbeglo omogućeno je da se u toku samog programskog sken ciklusa zahteva očitavanje trenutne vrednosti signala na nekoj od ulaznih linija, ili trenutno prenošenje izračunate akcije na izlaznu liniju. Ovo se postiže posebnim naredbama kojima se privremeno prekida programski sken ciklus i izvršava deo ulaznog odnosno izlaznog sken ciklusa.

- *IIM – Immediate input with mask* - ukolikoje uslov istinit, *IIM* naredba će prouzrokovati očitavanje (sken) svih ulaznih linija koje se nalaze u modulu u datom slotu. Vrednosti bitova koji odgovaraju bitovima koji su u masci postavljeni na 1 biće smešteno u sliku ulaza (ulaznu datotreku) i korišćena u svim sledećim sken naredbama
- *IOM – Immediate output with mask* - pri isitnitom uslovu *IOM* naredba će prouzrokovati da se vrednosti bitova koji se nalaze u slici izlaza (izlaznoj datoteci), a koji odgovaraju bitovima koji su u masci postavljeni na 1, trenutno prenesu na izlazne linije modula u datom slotu.
- *REF – I/O refresh* - ukoliko je uslov istinit *REF* naredba prouzrokovaće prekid programskog sken ciklusa i obavljanje izlaznog sken ciklusa, ciklusa komunikacije i održavanja, kao i ulaznog sken ciklusa. Posle toga se programski sken ciklus nastavlja od mesta gde je prekinut.

41. Sistem prekida (Interrupt).

Prekid je mehanizam pomoću koga se neki program privremeno prekida da bi se omogućilo izvršavanje nekog drugog posebnog dela programa, koji se označava kao servisni potprogram. Kada se servisni potprogram završi, nastavlja se izvršavanje programa koji je bio prekinut. Svrha ovih signala je "obaveštavanje" procesora o nastanku nekih spoljašnjih događaja. U zavisnosti od vrste događaja procesor menja redosled izvođenja operacija ili reaguje na neki drugi unapred predviđen način. Ovi signali imaju izuzetan znacaj za rad sistema u realnom vremenu i koriste se kao:

- *časovnik realnog vremena* - gde spoljašnji hardverski uređaj generiše signal u ravnomernim vremenskim intervalima;
- *događaji (alarmi)* - gde se nastanak neke nepredviđene situacije (događaja) na procesu može identifikovati tako što će odgovarajući senzori generisati digitalne signale;
- *ručno upravljanje*, gde se korišćenjem prekida može omogućiti da se preuzme ručno upravljanje procesom u slučaju regularnog remonta ili opravki na sistemu;
- *indikacije hardverskog otkaza* - gde se informacija o otkazu spoljašnjeg hardvera ili podsistema za spregu može dobiti preko signala prekida;
- *pomoć pri traženju grešaka u programu* - gde se prekid često koristi za prekidanje rada programa na određenim mestima u fazi provere njegove ispravnosti;
- *nestanak napajanja* - gde se u računar uključuje kolo koje veoma brzo detektuje gubitak napajanja u sistemu i obezbeđuje upozorenje nekoliko milisekundi pre nego što sistem prestane da radi.

Realizacija sistema prekida obuhvata detekciju signala prekida, suspenziju trenutne aktivnosti, pronalaženje ure_aja koji je tražio prekid, opsluživanje prekida i nastavak suspendovane aktivnosti.

Tipovi prekida i hijerarhijski nivoi

Ukoliko se istovremeno javi više od jednog signala prekida, procesor ih opslužuje u redosledu kojim su ovde dati. Isto tako prekid nižeg hijerarhijskog nivoa ne može odpočeti da se opslužuje ukoliko se u tom trenutku opslužuje neki prekid višeg hijerarhijskog nivoa.

Detekcija signala prekida i period latentnosti – Nailazak signala prekida, po pravilu, ne može baš trenutno da prekine rad procesora. To znaci da ce do pocetka opsluživanja signala prekida proteci izvesno vreme. Taj vremenski interval oznacava se kao interval latentnosti i njegova dužina zavisi od operacije koja se trenutno izvodi i od zahteva koje specificira korisnik.

Opsluživanje prekida – Kada PLC registruje prekid i završi zapocetu operaciju, odnosno do_e u stanje u kome se dozvoljava opsluživanje prekida, on sacuva vrednosti statusnih bitova, a zatim pozove odgovarajuci servisni potprogram.

Servisni potprogram – Servisni potrogram je posebna programska datoteka. U servisnom potprogramu ne smeju se koristiti TND, REF i SVC naredbe. Pored toga, servisni potprogram može pozivati najviše tri nivoa ucaurenih (jedan podprogram može pozvati drugi podprogram koji poziva treci i tako redom, pri cemu je dozvoljeno povezivanje do osam nivoa podprograma) potprograma. Ukoliko se želi trenutna spoljna reakcija na detektovani prekid moraju se koristi naredbe

42. User Fault Routine.

User Fault Routine je specifična vrsta prekida koja nastaje ukoliko se u toku izvođenja leder programa javi greška. Jedan broj ovih grešaka može biti prouzrokovan izvođenjem

matematickih operacija, neadekvatnim adresiranjem ulazno/izlaznih modula i tome slicno. Ukoliko se detektuje greška korisnik ima mogućnost da zahteva prekid i da pokuša da otkloni grešku.

Definisanje servisnog potprograma – servisni potprogram, po pravilu ispituje kod greške i ustanovljava da li je ona popravljiva ili nije. Ukoliko je greška popravljiva, ona se ispravlja. Ukoliko greška nije popravljiva šalje se poruka nekom susednom cvoru u mreži da PLC privremeno obustavlja rad (sve dok se ne ustanovi šta izaziva grešku i ne unesu se odgovarajuće korekcije).

Detekcija i način opsluživanja prekida – kada se u toku izvršavanja programa detektuje greška procesor generiše određen broj podataka u datoteci statusa. Ako je podatak jednak 0, to znači da korisnik nije želeo da se bavi ispitivanjem i eventualnim popravkom grešeka, te će procesor zaustaviti dalji rad. Ukoliko se međutim u toj reči nalazi bilo koji broj između 3 i 255, onda taj broj označava programsku datoteku koja sadrži servisni potprogram i procesor otpočinje izvođenje servisnog potrprograma.

Postavljanje internih indikatorskih bitova i reci - ukoliko dođe do greške procesor će postaviti sledeće bitove u datoteci statusa

- S:1/13 (*major error bit*)
- S:6 - kod greške
- S:20 - broj ranga u kome je greška nastala
- S:21 - broj programske datoteke u kome se taj rang nalazi.

43. DII – Discrete Input Interapt.

Ovaj nivo prekida koristi se za registrovanje nastanka određenih događaja i obezbeđivanje odgovarajuce reakcije na te događaje. Događaji se detektuju preko očitavanja stanja linija na nekom ulaznom modulu. Istovremeno se specificira i slot u kome se nalazi digitalni ulazi koji primaju ove bitove. Kada procesor detektuje pojavljivanje zadanog niza u odgovarajućem slotu, on registruje nastanak događaja. Pri tome, ako je sistem prekida definisan tako da registruje svaki događaj, procesor će automatski da pozove servisni potprogram. Ukoliko je, međutim sistem definisan tako da prebrojava događaje, onda će servisni potprogram biti pozvan tek kad se dostigne zadani broj nastanka događaja. Definisanje načina rada ovog prekida, kao i odgovarajuće reakcije obavlja se pomocu šest reči u statusnoj datoteci. Sam procesor, opslužujuci prekid postavlja još neke indikatorske bitove u statusnoj datoteci.

44. STI – Selectable Timed Interrupt.

STI je sistem prekida koji omogućava da se sken ciklus periodično prekida i da se pri svakom prekidu obradi odgovarajući niz naredbi koji se nalazi u servisnom potprogramu. Kada se servisni potprogram završi, sken ciklus se nastavlja od momenta u kome je bio prekinut.

Definisanje periode ponavljanja prekida

Sistemom prekida se upravlja pomoću internog časovnika. Periodu ponavljanja korisnik definiše sam upisivanjem odgovarajuće vrednosti u statusnoj datoterci.

- S:30 – *period* – multipl osnovnog takta. Period ponavljanja se dobija kao proizvod sdržaja reči S:30 i osnovnog takta. Ukoliko se kao multipl upiše 0, sistem prekida je onemogućen.
- *Takt* zavisi od tipa procesora
- SLC 5/02 – takt = 10ms, period : 10 – 2550 ms

- SLC 5/03 i više takt zavisi od vrednosti bita S:2/10
    - S:2/10 = 0 – takt = 10ms, period od 10 do 32760ms
    - S:2/10 = 1 – takt = 1ms, priod od 1 do 32767ms

Definisanje servisnog potprograma

- S:31 - broj programske datoteke koja sadrži servisni potprogram. Ukoliko je ovaj broj nula, prekid je onemogućen.

Detekcija i način opsluživanja prekida

U trenutku započinjanja programa interni časovnik otpočinje sa radom. Časovnik radi sa osnovnim taktom od 10μs. Kada časovnik izmeri vreme koje odgovara periodi odabiranja generiše se signal prekida i otpočinje izvršavanje servisnog potprograma. Izvršavanjem STI prekida se može dodatno upravljati pomocu tri posebne naredbe:

- STS – selectable timed start
- STE – Selectable timed enable
- STD – selectable timed disabled

Postavljanje internih indikatorskih bitova i reci

U toku rada, procesor postavlja niz inidkatorskih bitova vezanih za STI prekid cime se korisniku omogucava da prati nacin na koji se prekid opslužuje.

45. I/O prekidi.

Neki specijalni U/I moduli imaju sposobnost generisanja signala prekida. Ukoliko se ovaj signal detektuje, procesor prekida sken ciklus i započinje opsluživanje odgovarajućeg servisnog potprograma. Specijalni moduli koji se koriste uz sistema prekida treba da se postave u početne slotove.

Definisanje servisnog potprograma

Broj programske datoteke koja sadrži servisni potprogram mora se upisati u samom specijalnom modulu za vreme njegove konfiguracije. Prva naredba u prvom rangu servisnog potprograma je INT naredbom koja ukazuje da se radi o potprogramu koji opslužuje I/O prekid.

Detekcija i nacin opsluživanja prekida

Kada specijalni U/I modul generiše signal prekida onda procesor prekida svoj rad i poziva odgovarajući servisni potprogram na izvršavanje. Ukoliko se u trenutku zahteva za prekidom izvršava neki prekid višeg prioriteta onda će se sačekati da se svi prekidi višeg prioriteta završe pre nego što počne opsluživanje ovog prekida. Na isti način, ukoliko se u toku opsluživanja prekida pojavi zahtev za nekim prekidom višeg prioriteta (Fault, DII ili STI), opluživanje se prekida.

Za izvršavanjem I/O prekida se može dodatno upravljati pomoću dve posebne naredbe: •

- IIE – Interrupt enable
- IID – Interrupt disable

Postavljanje internih indikatorskih bitova i reci

U toku rada, procesor postavlja niz inidkatorskih bitova vezanih za I/O prekid čime se korisniku omogućava da prati način na koji se prekid opslužuje.